ON FUNCTION SPACES AND POLYNOMIAL-TIME COMPUTABILITY

AKITOSHI KAWAMURA

In Computable Analysis, elements of uncountable spaces, such as the real line $\mathbb{R}$, are represented by functions on strings and fed to Turing machines as oracles; or equivalently, they are represented by infinite strings and written on the tapes of Turing machines [Wei00, BHW08]. To obtain reasonable notions of computability and complexity, it is hence important to choose the "right" representation (encoding) for the spaces being considered.

Let's say we have already agreed upon representations γ and δ of spaces X and Y (that are admissible with the topologies of X and Y). How would we represent the space $\mathrm{C}[X \to Y]$ of continuous functions from X to Y? It is known that there is a natural representation $[\gamma \to \delta]$ of $\mathrm{C}[X \to Y]$ which is characterized by the property that it is the poorest representation that makes function evaluation computable [Wei00, Lemma 3.3.14].

Is there a representation with a similar property also at the level of polynomial-time computability (as introduced in [KF82] and extended in [KC96, KC12])? In this note we observe that there is such a nice representation for the space of continuous real-valued functions. Generalization to other spaces is left for future research.

1. Type-two polynomial-time computability

We consider computational problems as *multi-valued functions* from the set X of possible inputs to the set Y of possible outputs. An (X, Y)-*problem* F is formally a subset of $X \times Y$. The set of $x \in X$ such that there is $y \in Y$ with $(x, y) \in F$ is called the *domain of definition* or the *promise* of F and denoted $\operatorname{dom} F$. For $x \in \operatorname{dom} F$, we write $F[x]$ for the (nonempty) set of all such y. If $F[x]$ is a singleton, we write $F(x)$ for the unique element of $F[x]$. When this is the case for all $x \in \operatorname{dom} F$, we say that F is a *single-valued* problem, or a *partial function*. When $\operatorname{dom} F = X$, we say that F is *total*. A single-valued total problem is called a *function*. The intuitive interpretation is that F specifies a problem where, given any $x \in \operatorname{dom} F$, you are required to output *some* element of $F[x]$. Thus, the specification becomes stricter as $\operatorname{dom} F$ gets bigger or as $F[x]$ (for some $x \in \operatorname{dom} F$) gets smaller.

For a (Y, Z)-problem F and an (X, Y)-problem G, we define the (X, Z)-problem $F \circ G$ by saying that its promise is

$$\operatorname{dom}(F \circ G) = \{\, x \in \operatorname{dom} G : G[x] \subseteq \operatorname{dom} F \,\}, \tag{1}$$

and that, for any x in this promise,

$$(F \circ G)[x] = \bigcup_{y \in G[x]} F[y]. \tag{2}$$

As noted at the beginning, we will use (a certain class of) string functions to encode objects. We say that a (total) (Σ^*, Σ^*)-function φ is *length-monotone* if it preserves relative lengths of strings in the sense that $|\varphi(u)| \le |\varphi(v)|$ whenever $|u| \le |v|$. We write Σ^{**} for the set of all length-monotone functions. We restrict attention to Σ^{**} (rather than using all (Σ^*, Σ^*)-functions) to keep the notion of their *size* (to be defined shortly) simple.

This work is supported in part by the Grant-in-Aid for Scientific Research (Kakenhi) 23700009.

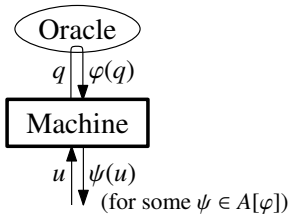

FIGURE 1. A machine solving a $(\Sigma^{**}, \Sigma^{**})$-problem A.

We use an oracle Turing machine (henceforth just "machine") to convert length-monotone functions to length-monotone functions (Figure 1).

Definition 1. A machine M *solves* a $(\Sigma^{**}, \Sigma^{**})$-problem A if for any $\varphi \in \operatorname{dom} A$, there is $\psi \in A[\varphi]$ such that M on oracle φ and any string u outputs $\psi(u)$ and halts.

Remark 2. 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.

Length-monotone functions map strings of equal length to strings of equal length. Therefore it makes sense to define the *size* $|\varphi|$ of a length-monotone function φ to be the (non-decreasing) $(\mathbb{N}, \mathbb{N})$-function $|\varphi|(|u|) = |\varphi(u)|$. Note that, despite whatever the term "size" might suggest, it is a function.

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 [KC96]. *Second-order polynomials* (in type-1 variable $\mathtt{L}$ and type-0 variable $\mathtt{n}$) are defined inductively as follows: a positive integer is a second-order polynomial; the variable $\mathtt{n}$ is also a second-order polynomial; if P and Q are second-order polynomials, then so are $P + Q$, $P \cdot Q$ and $\mathtt{L}(P)$. An example is

$$\mathtt{L}\big(\mathtt{L}(\mathtt{n} \cdot \mathtt{n})\big) + \mathtt{L}\big(\mathtt{L}(\mathtt{n}) \cdot \mathtt{L}(\mathtt{n})\big) + \mathtt{L}(\mathtt{n}) + 4. \tag{3}$$

A second-order polynomial P specifies a function, which we also denote by P, that takes an $(\mathbb{N}, \mathbb{N})$-function L to another $(\mathbb{N}, \mathbb{N})$-function $P(L)$ in the obvious way. For example, if P is the above second-order polynomial (3) and $L(n) = n^2$, then $P(L)$ is given by

$$P(L)(n) = \big((n \cdot n)^2\big)^2 + (n^2 \cdot n^2)^2 + n^2 + 4 = 2 \cdot n^8 + n^2 + 4. \tag{4}$$

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

Definition 3. A machine M runs in *polynomial time* if there is a second-order polynomial P such that, given any $\varphi \in \Sigma^{**}$ as oracle and any $u \in \Sigma^*$ as input, M halts within $P(|\varphi|)(|u|)$ steps.

We write **FComp** (resp. **FP**) for the class of $(\Sigma^{**}, \Sigma^{**})$-problems solved by a machine (resp. that runs in polynomial time). We can suitably define some other complexity classes related to nondeterminism or space complexity, as well as the notions of reduction and hardness [KC12].

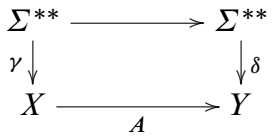

FIGURE 2. (γ, δ)-solving a problem A.

2. COMPUTATION ON REPRESENTED SPACES

A *representation* γ of a set X is formally a partial (Σ^{**}, X)-function which is surjective—that is, for each $x \in X$, there is at least one $\varphi \in \Sigma^{\mathbb{N}}$ with $\gamma(\varphi) = x$. We say that φ is a γ-*name* of x.

Definition 4. Let γ and δ be representations of sets X and Y, respectively. We say that a machine (γ, δ)-*solves* an (X, Y)-problem A if it solves the $(\Sigma^{**}, \Sigma^{**})$-problem $\delta^{-1} \circ A \circ \gamma$.

In other words, whenever the machine is given a γ-name of an element $x \in \operatorname{dom} A$, it must output some δ-name of some element of $A[x]$ (Figure 2). We write (γ, δ)-**FComp** (resp. (γ, δ)-**FP**) for the class of (X, Y)-problems solved by a machine (resp. that runs in polynomial time).

Example 5. We give a representation of the real numbers $\mathbb{R}$.

For each $n \in \mathbb{N}$, let $\mathbf{D}_n$ denote the set of strings of the form

$$sx/1\underbrace{00\ldots0}_{n}, \tag{5}$$

where $s \in \{+, -\}$ and $x \in \{0, 1\}^*$. Let $\mathbf{D} = \bigcup_{n \in \mathbb{N}} \mathbf{D}_n$. A string in $\mathbf{D}$ *encodes* a number in the obvious sense—namely, read (5) as a fraction whose numerator and denominator are integers in binary (with leading zeros allowed in the numerator). We write $[\![u]\!]$ for the number encoded by $u \in \mathbf{D}$. The numbers that can be encoded in this way are called *dyadic* rationals.

We define a representation $\rho_{\mathbb{R}}$ of $\mathbb{R}$ as follows: $\varphi \in \Sigma^{**}$ is a $\rho_{\mathbb{R}}$-name of $x \in \mathbb{R}$ if $\varphi(0^i) \in \mathbf{D}$ and $|[\![\varphi(0^i)]\!] - x| < 2^{-i}$ for each $i \in \mathbb{N}$. Thus we specify a real number as a list of dyadic rationals converging to it. We write $\rho_{\mathbb{I}}$ for the restriction of $\rho_{\mathbb{R}}$ to (names of) real numbers in the interval $\mathbb{I} = [0, 1]$. It turns out that $(\rho_{\mathbb{I}}, \rho_{\mathbb{R}})$-**FP** and $(\rho_{\mathbb{R}}, \rho_{\mathbb{R}})$-**FP** coincide with the well-studied classes [KF82] of polynomial-time computable real functions; see [KC12, Section 4.1].

The complexity notions for $(\Sigma^{**}, \Sigma^{**})$-problems defined in the previous section, combined with representations for various spaces in analysis, provide useful tools to discuss the complexity of numerical problems (see Kawamura and Cook [KC12]). For this purpose, it is important to choose a suitable representation for each application. Two representations can be compared in terms of whether one can be easily translated into the other:

Definition 6. Let γ and δ be two representations of the same set X. We write $\gamma \leq_{\mathbf{FComp}} \delta$ (resp. $\gamma \leq_{\mathbf{FP}} \delta$) if $\delta^{-1} \circ \gamma$ is in **FComp** (resp. **FP**).

Thus, $\gamma \leq_{\mathbf{FComp}} \delta$ (resp. $\gamma \leq_{\mathbf{FP}} \delta$) if there is a $(\Sigma^{**}, \Sigma^{**})$-problem F in **FComp** (resp. in **FP**) that *translates* γ to δ in the sense that $\delta \circ F$ realizes γ (Figure 3). This may be interpreted as γ being "richer" or "more informative" than δ. The preorders $\leq_{\mathbf{FComp}}$ and $\leq_{\mathbf{FP}}$ define equivalence classes of representations, with $\leq_{\mathbf{FP}}$-equivalence finer.

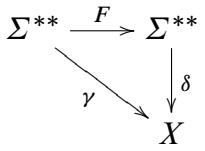

FIGURE 3. Function F translating γ to δ.

3. PRODUCTS AND FUNCTION SPACES

In this section, we discuss canonical ways to construct the representations of the product space and of the space of continuous functions. We will see that the "right" representation for the product is nicely characterized up to $\leq_{\mathbf{FP}}$-equivalence, but for the function space such a nice representation in general exists only up to $\leq_{\mathbf{FComp}}$.

3.1. **Products.** We define the pairing function for length-monotone functions as follows: for $\varphi, \psi \in \Sigma^{**}$, define $\langle \varphi, \psi \rangle \in \Sigma^{**}$ by setting $\langle \varphi, \psi \rangle(0u) = \varphi(u)10^{|\psi(u)|}$ and $\langle \varphi, \psi \rangle(1u) = \psi(u)10^{|\varphi(u)|}$ (we pad the strings to make $\langle \varphi, \psi \rangle$ length-monotone).

Definition 7. Let γ_0 and γ_1 be representations of X_0 and X_1, respectively. We define the representation $[\gamma_0, \gamma_1]$ of the Cartesian product $X_0 \times X_1$ by setting $[\gamma_0, \gamma_1](\langle \varphi_0, \varphi_1 \rangle) = (\gamma_0(\varphi_0), \gamma_1(\varphi_1))$ for each $\varphi_0, \varphi_1 \in \Sigma^{**}$.

The following theorem characterizes $[\gamma_0, \gamma_1]$ (up to $\leq_{\mathbf{FP}}$-equivalence) as the poorest representation that makes projections efficiently computable.

Theorem 8. *Let γ_0 and γ_1 be representations of X_0 and X_1, respectively. Let δ be any representation of $X_0 \times X_1$. Then the two projections are in (δ, γ_0)-**FP** and (δ, γ_1)-**FP**, respectively, if and only if $\delta \leq_{\mathbf{FP}} [\gamma_0, \gamma_1]$.*

3.2. **Function spaces.** Consider Σ^{**} as a topological space as follows: for each partial (Σ^*, Σ^*)-function p with dom p finite, we write $[p]$ to denote the set of all functions $\varphi \in \Sigma^{**}$ that agree with p on dom p; those $[p]$ form an open base of the topology on Σ^{**}.

When a set X and its representation γ is given, we define the induced topology on X by calling $U \subseteq X$ open when the inverse image $\gamma^{-1}[U]$ is open in Σ^{**}. Note that the representations $\rho_{\mathbb{R}}$ and $\rho_{\mathbb{I}}$ of in Example 5 give $\mathbb{R}$ and $\mathbb{I}$ the usual topology.

Let γ_1, γ_0 be representations of X_1 and X_0, respectively, and consider the space $\mathrm{C}[X_1 \to X_0]$ of (total) continuous (X_1, X_0)-functions (with respect to the induced topologies).

Definition 9. Let γ_0 and γ_1 be representations of X_0 and X_1, respectively. We define the representation $[\gamma_1 \to \gamma_0]$ of $\mathrm{C}[X_1 \to X_0]$ by setting $[\gamma_1 \to \gamma_0](\langle M, \varphi \rangle) = f$ if and only if f is (γ_1, γ_0)-solved by $M^{\langle \varphi, \cdot \rangle}$ (the machine M expecting two oracles, one of which is hardwired to $\varphi \in \Sigma^{**}$ and the other is still to be filled in with a γ_1-name). Here, M is a finite string (for a machine program), so $\langle M, \varphi \rangle$ means some reasonable encoding of M and φ by a length-monotone function (for example, we could regard M as a constant function with value M, and then couple it with φ using the pairing function defined in Section 3.1).

That $[\gamma_1 \to \gamma_0]$ is indeed a representation follows from the fact [Wei00, Lemma 2.3.11, Theorem 3.2.11] that every function in $\mathrm{C}[X_1 \to X_0]$ is computable relative to some oracle φ.

Define the $(\mathrm{C}[X_1 \to X_0] \times X_1, X_0)$-function *Apply* by $\mathit{Apply}(f, x) = f(x)$. The following lemma says that (the $\leq_{\mathbf{FComp}}$-equivalence class of) the representation $[\gamma_1 \to \gamma_0]$ is the poorest representation of $\mathrm{C}[X_1 \to X_0]$ that makes *Apply* computable.

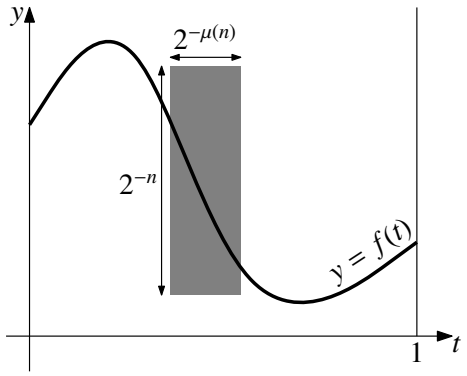

FIGURE 4. Modulus of continuity μ of a real function $f \in \mathrm{C}[\mathbb{I} \to \mathbb{R}]$.

Theorem 10 ([Wei00, Lemma 3.3.14])**.** *Let γ_0 and γ_1 be representations of X_0 and X_1, respectively. Let δ be any representation of $\mathrm{C}[X_1 \to X_0]$. Then Apply is in $([\delta, \gamma_1], \gamma_0)$-***FComp** *if and only if $\delta \leq_{\mathbf{FComp}} [\gamma_1 \to \gamma_0]$.*

Theorems 8 and 10 are useful because they induce representations on new spaces in a canonical way, allowing us to apply the theory of computation to many objects encountered in mathematics. However, unlike Theorem 8 about polynomial-time computability, Theorem 10 is stated with computability only. The representation $[\gamma_1 \to \gamma_0]$ does not in general make *Apply* polynomial-time computable. Now it is natural to ask:

Question 11. *Is there (under some additional assumptions if necessary) a representation of $\mathrm{C}[X_1 \to X_0]$ satisfying the property in Theorem 10 with* **FComp** *replaced by* **FP***?*

4. REPRESENTATIONS OF REAL FUNCTIONS

We close this note by pointing out that the answer to Question 11 is yes when X_1, X_0 are the spaces $\mathbb{I}$, $\mathbb{R}$ with the representations from Example 5.

We say that a non-decreasing $(\mathbb{N}, \mathbb{N})$-function μ is a *modulus of continuity* of a function $f \in \mathrm{C}[\mathbb{I} \to \mathbb{R}]$ if for all $n \in \mathbb{N}$ and $t, t' \in \mathbb{I}$ such that $|t - t'| \leq 2^{-\mu(n)}$, we have $|f(t) - f(t')| \leq 2^{-n}$ (Figure 4). Note that any $f \in \mathrm{C}[\mathbb{I} \to \mathbb{R}]$ is uniformly continuous and thus has a modulus of continuity. Given μ, we define $\overline{\mu} \in \Sigma^{**}$ by $\overline{\mu}(u) = 0^{\mu(|u|)}$.

Define the representation $\delta_\square$ of $\mathrm{C}[\mathbb{I} \to \mathbb{R}]$ as follows: we set $\delta_\square(\langle \overline{\mu}, \varphi \rangle) = f$ if and only if μ is a modulus of continuity of f and, for every $n \in \mathbb{N}$ and $u \in \mathbf{D}$ with $[\![u]\!] \in \mathbb{I}$, we have $v := \varphi(0^n, u) \in \mathbf{D}$ and $|f([\![u]\!]) - [\![v]\!]| < 2^{-n}$ (the string v may need to have leading 0s padded in order to make φ length-monotone). Thus, a $\delta_\square$-name of a real function is a pair consisting of its modulus of continuity and the approximate values on dyadic rationals.

It is easy to verify that $\delta_\square$ is indeed a representation. It was defined and used in [KC12, Section 4.3] to discuss complexity of operators on real functions. The following theorem justifies this choice by showing that $\delta_\square$ satisfies the polynomial-time version of Theorem 10.

Theorem 12. *Let δ be any representation of $\mathrm{C}[\mathbb{I} \to \mathbb{R}]$. Then Apply is in $([\delta, \rho_\mathbb{I}], \rho_\mathbb{R})$-***FP** *if and only if $\delta \leq_{\mathbf{FP}} \delta_\square$.*

Proof. For the "if" part, it suffices to prove that *Apply* is in $([\delta_\square, \rho_\mathbb{I}], \rho_\mathbb{R})$-**FP**. Let a $\delta_\square$-name $\langle \overline{\mu}, \varphi \rangle$ of $f \in \mathrm{C}[\mathbb{I} \to \mathbb{R}]$ and a $\rho_\mathbb{I}$-name θ of $t \in \mathbb{I}$ be given. We get a $\rho_\mathbb{R}$-name ξ of $f(t)$ by computing $\xi(0^n)$ (that is, a 2^{-n}-approximation of $f(t)$) as follows. First read $m := \mu(n+1)$ to see how precisely we need to know t in order to determine $f(t)$ within error bound 2^{-n-1}.

Then read $u := \theta(0^m)$ to get a 2^{-m}-approximation $[\![u]\!]$ of t. Because μ is a modulus of continuity, $|f([\![u]\!]) - f(t)| < 2^{-n-1}$. Finally, read $v := \varphi(0^{n+1}, u)$. Then $[\![v]\!]$ is a 2^{-n-1}-approximation of $f([\![u]\!])$, and hence a 2^{-n}-approximation of $f(t)$. All this can be done in second-order polynomial time in μ, $|\varphi|$ and $|\theta|$.

For the other direction, suppose that *Apply* is in $([\delta, \rho_{\mathbb{I}}], \rho_{\mathbb{R}})$-**FP**. That is, there is a second-order polynomial P such that given a δ-name ψ of a function $f \in \mathrm{C}[\mathbb{I} \to \mathbb{R}]$, a $\rho_{\mathbb{I}}$-name θ of a number $t \in \mathbb{I}$, and a unary string 0^n, it is possible to compute the value on 0^n of a $\rho_{\mathbb{R}}$-name of $f(t)$ in time $P(|\psi|, |\theta|)(n)$. We need to translate δ to $\delta_\Box$. Let a δ-name ψ of $f \in \mathrm{C}[\mathbb{I} \to \mathbb{R}]$ be given. To get a $\delta_\Box$-name $\langle\overline{\mu}, \varphi\rangle$ of f, we do as follows. Let $\mu = P(|\psi|, \lambda)$, where λ is a (usual) polynomial such that any number in $\mathbb{I}$ has a $\rho_{\mathbb{I}}$-name of size λ; for example, $\lambda(n) = 2n + 4$ will do, because every number in $\mathbb{I}$ has a 2^{-n}-approximation of the form (5) where x has length $n + 1$. Then μ is a modulus of continuity of f, because a 2^{-n}-approximation of $f(t)$, where t is given as a $\rho_{\mathbb{I}}$-name θ of size λ, can be computed within time bound $P(|\psi|, \lambda)(n)$ and hence without reference to θ's values on strings longer than this bound. The other part φ of the $\delta_\Box$-name should be such that $\varphi(0^n, u)$, where $u \in \mathbf{D}$, is a 2^{-n}-approximation of $f([\![u]\!])$. This can be computed from ψ, 0^n, u as follows. Let $\theta \in \Sigma^{**}$ be the constant function taking value u. Note that θ is a $\rho_{\mathbb{R}}$-name of $[\![u]\!]$. We then run the assumed algorithm for *Apply* on oracles ψ and θ and string 0^n. This takes time $P(|\psi|, |\theta|)(n)$, which is polynomial in $|\psi|$ and $|u|, n$. □

We remark that we can construct a similar representation of $\mathrm{C}[\mathbb{R} \to \mathbb{R}]$ (which we also denote by $\delta_\Box$ by abuse of notation) by using a modified notion of the modulus of continuity as follows. An $(\mathbb{N} \times \mathbb{N}, \mathbb{N})$-function μ is a modulus of continuity of $f \in \mathrm{C}[\mathbb{R} \to \mathbb{R}]$ if for all $m, n \in \mathbb{N}$ and $t, t' \in \mathbb{R}$ such that $|t|, |t'| \leq 2^m$ and $|t - t'| \leq 2^{-\mu(m,n)}$, we have $|f(t) - f(t')| \leq 2^{-n}$. The representation $\delta_\Box$ is again defined by saying that $\langle\overline{\mu}, \varphi\rangle$ is a $\delta_\Box$-name of $f \in \mathrm{C}[\mathbb{R} \to \mathbb{R}]$ if and only if μ is a modulus of continuity of f and, for every $n \in \mathbb{N}$ and $u \in \mathbf{D}$, we have $v := \varphi(0^n, u) \in \mathbf{D}$ and $|f([\![u]\!]) - [\![v]\!]| < 2^{-n}$. The following can be verified by slightly modifying the proof of Theorem 12.

Theorem 13. *Let δ be any representation of* $\mathrm{C}[\mathbb{R} \to \mathbb{R}]$. *Then Apply is in* $([\delta, \rho_{\mathbb{R}}], \rho_{\mathbb{R}})$-**FP** *if and only if* $\delta \leq_{\mathsf{FP}} \delta_\Box$.

Can this be generalized somehow to other spaces or to other complexity classes? The general Question 11 is left for further investigation.

References

[BHW08] V. Brattka, P. Hertling, and K. Weihrauch. A Tutorial on Computable Analysis. In *New Computational Paradigms: Changing Conceptions of What is Computable*, S. B. Cooper, B. Löwe, and A. Sorbi (Eds.), 425–491. Springer, 2008.

[KC96] B. M. Kapron and S. A. Cook. A new characterization of type-2 feasibility. *SIAM Journal on Computing*, 25(1):117–132, 1996.

[KC12] A. Kawamura and S. Cook. Complexity theory for operators in analysis. *ACM Transactions on Computation Theory*, 4(2), Article 5, 2012.

[KF82] K. Ko and H. Friedman. Computational complexity of real functions. *Theoretical Computer Science*, 20(3):323–352, 1982.

[Wei00] K. Weihrauch. *Computable Analysis: An Introduction*. Springer, 2000.

Department of Computer Science, University of Tokyo
E-mail address: kawamura@is.s.u-tokyo.ac.jp