Computational Complexity in Analysis and Geometry

Akitoshi Kawamura

March 2011

Abstract

Computable analysis studies problems involving real numbers, sets and functions from the viewpoint of computability. Elements of uncountable sets (such as real numbers) are represented through approximation and processed by Turing machines. However, application of this approach to computational complexity has been limited in generality. In this thesis, we present a refined framework that is suitable for discussing computational complexity. The key idea is to use (a certain class of) string functions as names representing these objects. These are more expressive than infinite sequences, which served as names in prior work that formulated complexity in more restricted settings. An important advantage of using string functions is that we can define their *size* in the way inspired by higher-type complexity theory. This enables us to talk about computation on string functions whose time or space is bounded polynomially in the input size, giving rise to more general analogues of the classes P, NP, and PSPACE. We also define NP- and PSPACE-completeness under suitable many-one reductions.

Because our framework separates machine computation and semantics, it can be applied to problems on sets of interest in analysis once we specify a suitable representation (encoding). As prototype applications, we consider the complexity of several problems whose inputs and outputs are real numbers, real sets, and real functions. The latter two cannot be represented succinctly using existing approaches based on infinite sequences, so ours is the first treatment of functions on them. As an interesting example, the task of numerical algorithms for solving the initial value problem of differential equations is naturally viewed as an operator taking real functions to real functions. Because there was no complexity theory for operators, previous results could only state how complex the solution can be. We now reformulate them and show that the operator itself is polynomial-space complete. We survey some of such complexity results involving real numbers and cast them in our framework.

Acknowledgements

(will be added in the final version submitted to the SGS)

Contents

1 Introduction

Computable analysis [Wei00, BHW08] studies problems involving real numbers, sets and functions from the viewpoint of computability. There are several equivalent formulations, but one powerful approach (employed e.g. in [Wei00]) is to

(a) consider Turing machines converting infinite sequences (elements of $\{0,1\}^{\mathbb{N}}$) into infinite sequences, thus defining the class of computable functions from $\{0,1\}^{\mathbb{N}}$ to $\{0,1\}^{\mathbb{N}}$, and

(b) then discuss computability of problems whose inputs and outputs are taken from uncountable sets (such as $\mathbb{R}$, the real numbers) by representing (encoding) their elements by infinite sequences.

Such computability theory is called the Type-Two Theory of Effectivity (TTE) and provides a natural extension of the ordinary (type-one) computability. Here, “type-two” refers to the level of objects of which we are considering computability: type-zero means finite objects, such as strings or natural numbers; type-one means functions over type-zero objects, e.g. functions from strings to strings, or infinite strings (viewed as functions from $\mathbb{N}$ to $\{0,1\}$); our concern is the type-two theory, which is about the computability of functions whose inputs are type-one objects.

Since this framework separates the machine (formulation of part (a) above) and its semantics (part (b)), we can apply it not only to the real numbers but also with great generality to other spaces arising naturally in mathematical analysis, once we agree on the representation of the considered space. And in many cases, the appropriate representation to use is determined in a more or less natural way from each particular application. Thus there is a coherent way to discuss computability of real functions, sets of real numbers, operators taking real functions as inputs, and so on. Various problems in mathematical analysis and physics have been studied using this framework [PR89, Wei00].

In contrast, application of this approach to computational *complexity* (with bounded time or space) has been limited in generality. For example, although there is a widely accepted notion of polynomial-time computable real functions $f\colon [0,1]\to\mathbb{R}$ on the compact interval that has been studied extensively [Ko98], the same approach does not give a nice class of real functions on $\mathbb{R}$. Most of the complexity results in computable analysis have been (with a few exceptions [Hoo90, Tak01, Wei03, ZM08]) essentially limited to the complexity of either real functions with compact domain, or of bounded subsets of $\mathbb{R}$. They do not address the complexity of, say, an operator F that takes real functions $f\colon [0,1]\to\mathbb{R}$

to another real function $F(f)$. There are many positive and negative results [Ko91] about such operators, but typically they are stated in the form

> if f is in the complexity class X, then $F(f)$ is in complexity class Y, and
> there is f in complexity class X such that $F(f)$ is hard for Z.

More direct statements would be the "constructive" or "effectivized" form

> the operator F is in class $\mathcal{Y}$, and
> the operator F is $\mathcal{Z}$-hard,

where $\mathcal{Y}$ and $\mathcal{Z}$ are the "higher-order versions" of Y and Z. At the level of computability, it is common to ask, as soon as we see an ineffective result, whether it can be effectivized. For complexity, it was hard to even ask this question because we do not know how to formulate $\mathcal{Y}$ and $\mathcal{Z}$.

An approach to address this problem was proposed recently in [KC10], where we refine the above part (a) (machine model) of the computability theory to the level of complexity by using (a certain class of) functions from strings to string instead of infinite sequences and defining their *size* in the way suggested by Kapron and Cook [KC96]. This enables us to measure the growth of running time (or space) in terms of the input size—exactly what we do in the usual (type-one) complexity theory and what we were not able to do with the infinite string approach. We obtain the complexity classes **P**, **NP**, **PSPACE** analogous to P, NP, PSPACE (and function classes **FP**, **FPSPACE**, etc.) by bounding the time or space by *second-order polynomials* in the input size. Analogues of reductions and hardness can also be formulated, and we have type-two versions of complete problems for NP and PSPACE. This theory will be presented in Chapter 2.

In Chapter 3, we introduce the theory of representations in order to bring part (b) into complexity consideration. Most of the development here is similar or analogous to the corresponding parts of the computability argument, but for complexity we need to choose the right representations for each space more carefully. We discuss suitable representations of important spaces (real numbers, real functions, etc.). For real numbers, the induced complexity notions turn out to be equivalent to what has been studied by Ko–Friedman [KF82] and Hoover [Hoo90]. For sets and functions, the approach of [KC10] seems to be the first to provide complexity notions in a general way, but still there are pieces of evidence suggesting that the representations used there (and in this thesis) are the natural choices. For example, we will show (Theorem 3.33) that the representation $\delta_{\square}$ that we use to encode real continuous functions carries just the right information that enables us to evaluate a function at a given real number.

This extension will play an important role in Chapter 4 where we apply our framework to several specific numerical problems in the real world, because many such problems are naturally formulated as operators taking sets or functions. For example (Section 4.4.1), consider the operator F that finds the solution $F(f)$ of the differential equation (with a condition called *Lipschitz continuity*) given by a function f. As mentioned above, the foregoing ineffective results [Ko83, Kaw10] only talked about *how complex the solution $F(f)$ can be when f is easy*; precisely, they say that if f is polynomial-time computable, $F(f)$ is polynomial-space computable and can be polynomial-space hard. But the practical concern for numerical analysis would be *how hard it is to compute F* (i.e., to compute $F(f)$ given f). The framework of [KC10] made it possible to state and prove such a result: F itself is an **FPSPACE**-complete operator. The technically hard parts of the proofs of effectivized results like this are already done in the proofs of the ineffective versions, and in most cases all we need to do is to check that they effectivize. The original ineffective versions are now corollaries of the effectivized statements.

Chapter 4 is a collection of complexity results (some by the author and some reformulated in our framework) about various problems encountered in basic calculus and geometry. Many such problems contain combinatorial structures that make the consideration of complexity interesting. For example, integration (as an operator on real functions) is complete for the counting complexity class **#P** (Section 4.3.2), and (as can be seen from the proof) this can be explained by the intuitive idea that integration is an operator that "adds up" the given function. On the other hand, the above result about the **FPSPACE**-completeness of solving differential equations [Kaw10] requires a rather involved argument. It is proved by a combinatorially nontrivial insight that a polynomial-space computation can be described by a computation tableau that looks somewhat similar to the dynamics given by the differential equation.

As mentioned above, now we are able to discuss whether each complexity result is effective or not. Some of the results in Chapter 4 hold in the effectivized form (e.g. both the above results about integration and Lipschitz continuous differential equations). Others hold only in the ineffecive form. For example, we will see (Section 4.4.3) that the more restricted operator that solves *analytic* differential equations maps polynomial-time computable functions to polynomial-time computable functions, but this is not because the operator itself is polynomial-time computable, but because the operator that takes the Taylor series of the given function to that of the solution is polynomial-time computable.

2 Complexity of Type-Two Problems

Throughout the thesis, Σ^* denotes the set of finite strings over the alphabet Σ. We will tacitly assume that $\Sigma = \{0, 1\}$ in some contexts, and in other contexts that Σ contains all symbols needed in the discussion. Justification of this sloppiness should be easy.

The standard computational complexity theory classifies problems whose inputs and outputs are strings in Σ^*. By encoding various objects of interest by strings, we can discuss the hardness of problems involving strings, integers, graphs, etc. But in this thesis, we want to deal with problems involving uncountably many objects such as real numbers, sets of real numbers, and real-valued functions, which cannot be encoded by strings but can be specified only through approximation. For this purpose, this chapter develops a complexity theory for problems whose inputs and outputs are *functions from Σ^* to Σ^**. The following chapters will use these functions to *represent* various mathematical objects.

The notion of computability that our model will define is essentially the same thing as what has long been studied in the field called Computable Analysis [Wei00]. Our formulation enables us to refine this computability theory to a complexity theory by bounding the time and space used in the computation. Such complexity consideration has been already partly undertaken [Ko91], but our model makes it more broadly applicable. The way we bound time and space is adopted from Kapron and Cook's work [KC96] in higher-order complexity theory. Many of the results presented in this chapter and the next are from a joint work with Stephen Cook [KC10].

2.1 Preliminaries: search problems

Although most expositions of discrete complexity theory start with the discussion of decision problems where exactly one of 0 and 1 is the right answer, it is important in our setting to deal with *search problems* where we are supposed to find *any one* of the several allowable answers. Thus, we consider problems as *multi-valued* functions.

2.1.1 Problems

We formulate, in the most general way, problems where one is given an element of a set X as input and is asked to provide an element of Y as output.

Definition 2.1 (Problems)**.** A (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 ought 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. Thus we say that a problem F *realizes* G if $\operatorname{dom} F \supseteq \operatorname{dom} G$ and $F[x] \subseteq G[x]$ for all $x \in \operatorname{dom} G$.

Our problems defined above are what are sometimes called *search problems* (or "function problems" by slightly misleading terminology) in complexity theory. The classes FP and $\mathsf{FPSPACE}$ are the sets of (Σ^*, Σ^*)-problems that are computed by a machine whose running time or space, respectively, is polynomially bounded. Here, we interpret computation by the "allowable outputs" semantics: A machine is said to compute F if, on any $x \in \operatorname{dom} F$, it outputs *some* element of $F[x]$. The classes $\mathbf{FP}$ and $\mathbf{FPSPACE}$ that we will define later will consist of $(\mathbf{Reg}, \mathbf{Reg})$-problems, where $\mathbf{Reg}$ is something introduced in the next section in order to encode objects like real numbers or real functions which cannot be encoded by Σ^*.

Note that we impose nothing about the computation when the input x does not belong to the promise $\operatorname{dom} F$. You may or may not produce an output, and in case you do, the output can be anything. Thus a problem can be easy to solve while having a nasty set as promise. Because most parts of this thesis will be talking about algorithms with time (or space) bounds, and such algorithms can be made to always output something without increasing the bounds significantly, we could make all problems F total by extending them appropriately without essentially changing the complexity argument. However, we still prefer making the promise explicit, because it is convenient to specify which inputs we really care about.

The cost for this choice is, of course, the slightly more involved definitions. For the rest of this chapter, the reader may at first want to ponder what the definitions would look like for the special case when the problem in question is single-valued or total.

For example, we need to be careful in defining the composition of two problems. 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

$$\operatorname{dom}(F \circ G) = \{\, x \in \operatorname{dom} G : G[x] \subseteq \operatorname{dom} F \,\}, \tag{2.1}$$

and that, for any x in this promise,

$$(F \circ G)[x] = \bigcup_{y \in G[x]} F[y]. \tag{2.2}$$

The informal justification of this definition is that if algorithms M and N solve problems F and G, respectively, then the algorithm that "feeds the output of N as the input for M" solves $F \circ G$.

We write id for the (single-valued total) identity (X, X)-function on any set X. Obviously, $\mathrm{id} \circ F = F \circ \mathrm{id} = F$. It is easy to see that $F \circ (G \circ H) = (F \circ G) \circ H$, so we may write $F \circ G \circ H$. The following is also easy to see.

Lemma 2.2. *If F realizes F' and G realizes G', then $F \circ G$ realizes $F' \circ G'$.*

2.1.2 Continuity

We also extend the notion of continuity from functions to problems. Recall that a function $f\colon X \to Y$ is said to be continuous at $x \in X$ when for any open neighbourhood V of $f(x)$ there is an open neighbourhood U of x such that f maps U into V. We extend this idea to (X, Y)-problems F. For a set $V \subseteq Y$, we write

$$F^{-1}V := \{\, x \in \operatorname{dom} F : F[x] \cap V \neq \emptyset \,\} \tag{2.3}$$

for its *preimage* under F.

Definition 2.3 (Continuity)**.** Let X and Y be topological spaces and let F be an (X, Y)-problem. For each $x \in \operatorname{dom} F$ and $y \in F[x]$, we say that F is *continuous* at (x, y) if for any open set V containing y, there is an open set U containing x such that $U \cap \operatorname{dom} F \subseteq F^{-1}V$. We say that F is *strongly continuous* at $x \in \operatorname{dom} F$ if it is continuous at (x, y) for every $y \in F[x]$. We say that F is *strongly continuous* if it is strongly continuous at every $x \in \operatorname{dom} F$. We say that F is *continuous* if it is realized by some strongly continuous (X, Y)-problem.

Obviously, continuity and strong continuity coincide for single-valued problems, and they are equivalent to the usual notion of continuity on the domain of definition.

A (multi-valued) problem can be continuous without being realized by a continuous (single-valued) partial function. For example, the total $(\mathbb{R}, \mathbb{R})$-problem F defined by

$$F[x] = \begin{cases} \{1\} & \text{if } x \leq -\varepsilon, \\ \{0, 1\} & \text{if } -\varepsilon < x < \varepsilon, \\ \{0\} & \text{if } \varepsilon \leq x, \end{cases} \tag{2.4}$$

for $\varepsilon > 0$, is strongly continuous (Figure 2.1, left), but there is no way to choose its single-valued "branch" which is continuous. The total $(\mathbb{R}, \mathbb{R})$-problem F given by

$$F[x] = \begin{cases} \{1\} & \text{if } x < 0, \\ \{0, 1\} & \text{if } x = 0, \\ \{0\} & \text{if } x > 0 \end{cases} \tag{2.5}$$

is not continuous at $(0, 0)$ and $(0, 1)$ (Figure 2.1, right). The total $(\mathbb{R}, \mathbb{R})$-problem F given by

$$F[x] = \begin{cases} \{x \cdot \chi_{\mathbb{Q}}, 1\} & \text{if } x \neq 0, \\ \{0\} & \text{if } x = 0, \end{cases} \tag{2.6}$$

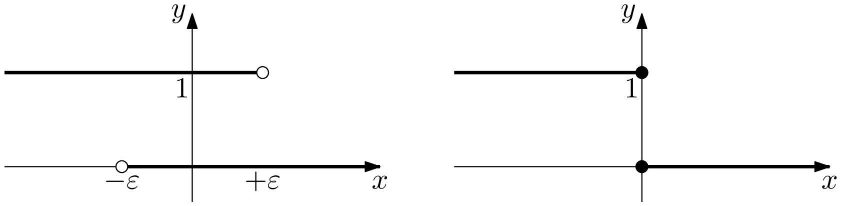


Figure 2.1: The finite precision test (left) is continuous, but the exact test (right) is not.

where $\chi_{\mathbb{Q}}$ is the $\{0,1\}$-valued characteristic function of the rational numbers $\mathbb{Q}$, is not continuous, even though for each $x \in \operatorname{dom} F$ there is $y \in F[x]$ such that F is continuous at (x, y).

Lemma 2.4. *An (X, Y)-problem F is strongly continuous if and only if, for any open set $V \subseteq Y$, the preimage $F^{-1}V$ is open in $\operatorname{dom} F$.*

Proof. For the forward direction, suppose that F is strongly continuous and $V \subseteq Y$ is open. Let x be any point in the preimage $F^{-1}V$. Then there is a point $y \in F[x] \cap V$. Since F is continuous at (x, y), there is an open set U containing x and satisfying $U \cap \operatorname{dom} F \subseteq F^{-1}V$. Since x was arbitrary, this means that $F^{-1}V$ is open in $\operatorname{dom} F$.

For the other direction, suppose that F is not strongly continuous, i.e., it is discontinuous at (x, y) for some $x \in \operatorname{dom} F$ and $y \in F[x]$. Then there is an open set V containing y such that no open set U containing x satisfies $U \cap \operatorname{dom} F \subseteq F^{-1}V$. Then $F^{-1}V$ is not open in $\operatorname{dom} F$, because x belongs to it but not to its interior. □

In the following lemma, recall that the composite $F \circ G$ is the one defined at (2.1) and (2.2).

Lemma 2.5. *If a (Y, Z)-problem F and a (X, Y)-problem G are continuous (resp. strongly continuous), so is $F \circ G$.*

Proof. The claim about continuity follows from the claim about strong continuity by Lemma 2.2. For the claim about strong continuity, let $W \subseteq Z$ be open. By the strong continuity of F and Lemma 2.4, there is an open set $V \subseteq Y$ such that

$$F^{-1}W = V \cap \operatorname{dom} F. \tag{2.7}$$

By the strong continuity of G and Lemma 2.4, there is an open set $U \subseteq X$ such that

$$G^{-1}V = U \cap \operatorname{dom} G. \tag{2.8}$$

We claim that

$$(F \circ G)^{-1}W = U \cap \operatorname{dom}(F \circ G). \tag{2.9}$$

Since W was arbitrary, this proves the strong continuity of $F \circ G$ by Lemma 2.4.

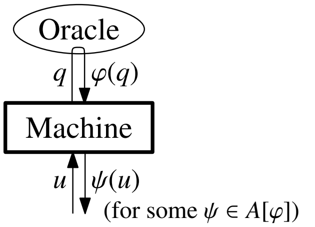


Figure 2.2: A machine solving a $(\mathbf{Reg}, \mathbf{Reg})$-problem A.

To prove one containment of (2.9), let $x \in (F \circ G)^{-1}W$. Then $x \in \operatorname{dom}(F \circ G)$ by definition, so we shall prove that $x \in U$. Since $(F \circ G)[x]$ intersects W, there is $y \in G[x]$ such that $F[y]$ intersects W by (2.2). This means that y belongs to $F^{-1}W$, and hence to V by (2.7). Thus, $G[x]$ intersects V, This means that x belongs to $G^{-1}V$, and hence to U by (2.8). Conversely, let $x \in U \cap \operatorname{dom}(F \circ G)$. Since $\operatorname{dom}(F \circ G) \subseteq \operatorname{dom} G$, we have $x \in U \cap \operatorname{dom} G = G^{-1}V$ by (2.8), and hence $G[x]$ intersects V, say at y. Since $G[x] \subseteq \operatorname{dom} F$, we have $y \in V \cap \operatorname{dom} F = F^{-1}W$ by (2.7), and hence $F[y]$ intersects W. By (2.2), this means that $(F \circ G)[x]$ intersects W, that is, $x \in (F \circ G)^{-1}W$. □

2.2 Type-two computation

We say that a total function $\varphi\colon \Sigma^* \to \Sigma^*$ is *regular* if it preserves relative lengths of strings in the sense that $|\varphi(u)| \leq |\varphi(v)|$ whenever $|u| \leq |v|$. We write **Reg** for the set of all regular functions. The rest of this chapter is about the complexity of $(\mathbf{Reg}, \mathbf{Reg})$-problems. The motivation for considering regular functions (rather than all functions from Σ^* to Σ^*) will be explained in Section 2.2.3. We write $\mathbf{Pred} \subseteq \mathbf{Reg}$ for the set of $\{0, 1\}$-valued regular functions, and sometimes consider $(\mathbf{Reg}, \mathbf{Pred})$-problems.

For later use we define the pairing and tupling function for regular functions as follows: For φ, $\psi \in \mathbf{Reg}$, define $\langle \varphi, \psi \rangle \in \mathbf{Reg}$ by setting $\langle \varphi, \psi \rangle(u) = \varphi(u)\#\psi(u)$ (that is, the two strings $\varphi(u)$ and $\psi(u)$ concatenated with a delimiter not in the original alphabet; if we need to strictly keep the binary alphabet $\{0, 1\}$, we can replace 0 and 1 in $\varphi(u)$ and $\psi(u)$ by 00 and 01, respectively, and then use 11 as a delimiter). Let $\langle \varphi, \psi, \theta \rangle = \langle \langle \varphi, \psi \rangle, \theta \rangle$, etc.

2.2.1 Oracle machines

We use an oracle Turing machine (henceforth just "machine") as the model of computation. Such a machine can convert elements of **Reg** to elements of **Reg** (Figure 2.2). For the precise conventions for issuing and answering queries, follow any of [Meh76, KC96, Ko91].

For a machine M, we define the partial $(\mathbf{Reg}, \mathbf{Reg})$-function $\underline{M}$ *computed* by it as follows. For each $\varphi \in \mathbf{Reg}$, let ψ_φ be the partial (Σ^*, Σ^*)-function such that $\operatorname{dom} \psi_\varphi$ consists of strings u such that M on oracle φ and input u halts, and for each such u the value $\psi_\varphi(u)$

is the output string. We let $\operatorname{dom} \underline{M} = \{ \varphi \in \mathbf{Reg} : \psi_\varphi \in \mathbf{Reg} \}$ and $\underline{M}(\varphi) = \psi_\varphi$ for all $\varphi \in \operatorname{dom} \underline{M}$.

Definition 2.6 (Solving type-two problems)**.** A machine M *solves* a $(\mathbf{Reg}, \mathbf{Reg})$-problem A if $\underline{M}$ realizes A.

In other words, M solves 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 $\psi(u)$. We write **FRec** for the set of $(\mathbf{Reg}, \mathbf{Reg})$-problems solved by some machine. (Throughout the thesis, we will consistently use boldface sans serif typeface for classes of $(\mathbf{Reg}, \mathbf{Reg})$-problems.)

Computability and continuity The set $\mathbf{Reg}$ is regarded naturally as a topological space as follows. Let Λ be the set of all partial (Σ^*, Σ^*)-functions k whose promise $\operatorname{dom} k$ is finite. For each $k \in \Lambda$, let B_k denote the set of regular functions that realize k (i.e., agree with k on $\operatorname{dom} k$). Now $\mathbf{Reg}$ is a topological space with open base $\{ B_k : k \in \Lambda \}$.

Because the machine M bases its string output $\underline{M}(\varphi)(u)$ on finitely many values of φ, the function $\underline{M}$ is continuous. This remains true if the machine is given some oracle $p \in \mathbf{Reg}$ (in addition to the input φ): if M is a machine that takes two oracles, and M^p denotes this machine with the oracle p hard-wired as one of the oracles, then the partial function $\underline{M^p}$ is continuous.

Theorem 2.7 (Essentially [Wei00, Lemma 2.3.11])**.** *A $(\mathbf{Reg}, \mathbf{Reg})$-problem is continuous if and only if it is solved by M^p for some machine M and some oracle $p \in \mathbf{Reg}$.*

Proof. The "if" direction is already explained above. For the other direction, let F be a continuous $(\mathbf{Reg}, \mathbf{Reg})$-problem. Define the desired oracle $p \in \mathbf{Reg}$ to be the list (encoded as a regular function in a reasonable way) of all pairs $(k, l) \in \Lambda \times \Lambda$ satisfying $B_k \cap \operatorname{dom} F \subseteq F^{-1} B_l$. The machine M^p, given $\varphi \in \operatorname{dom} F$ and $u \in \Sigma^*$, works as follows. For each $i = 0$, $1, \dots,$ let (k_i, l_i) be the first pair in the list p such that $\varphi \in B_{k_i}$, $\operatorname{dom} l_i \supseteq \Sigma^{\le i}$ and, if $i > 0$, then l_i agrees with l_{i-1} on $\Sigma^{\le i-1}$ (such a pair exists by the continuity of F). After computing l_i for all $i \le |u|$, it outputs $l_{|u|}(u)$. □

Corollary 2.8. *Every continuous $(\mathbf{Reg}, \mathbf{Reg})$-problem is realized by a continuous partial $(\mathbf{Reg}, \mathbf{Reg})$-function.*

This corollary is in contrast to the existence of $(\mathbb{R}, \mathbb{R})$-problems (such as the one defined at (2.4)) that do not have a continuous single-valued realization.

2.2.2 Second-order polynomials

Now we start restricting the amount of time and space in the computation. Recall that regular functions are those that respect lengths in the sense explained at the beginning of this section. In particular, they map strings of equal length to strings of equal length.

Therefore it makes sense to define the *size* $|\varphi|\colon \mathbb{N} \to \mathbb{N}$ of a regular function φ by $|\varphi|(|u|) = |\varphi(u)|$. It is a non-decreasing $(\mathbb{N}, \mathbb{N})$-function.

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 $\mathtt{L}(P)$. These polynomials will be used as a bound on time or space, and for simplicity we defined them to be monotone (there is no minus sign). An example is

$$\mathtt{L}\bigl(\mathtt{L}(\mathtt{n} \cdot \mathtt{n})\bigr) + \mathtt{L}\bigl(\mathtt{L}(\mathtt{n}) \cdot \mathtt{L}(\mathtt{n})\bigr) + \mathtt{L}(\mathtt{n}) + 4. \tag{2.10}$$

A second-order polynomial P specifies a function, which we also denote by P, that takes a function $L\colon \mathbb{N} \to \mathbb{N}$ to another function $P(L)\colon \mathbb{N} \to \mathbb{N}$ in the obvious way. For example, if P is the second-order polynomial (2.10) and $L(x) = x^2$, then $P(L)$ is given by

$$P(L)(x) = \bigl((x \cdot x)^2\bigr)^2 + (x^2 \cdot x^2)^2 + x^2 + 4 = 2 \cdot x^8 + x^2 + 4. \tag{2.11}$$

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

Now we can define classes by bounding the running time or space by second-order polynomials.

Definition 2.9 (Polynomial time and space)**.** A machine M *runs in polynomial time* (or *is polynomial-time*) if there is a second-order polynomial P such that, given any $\varphi \in \mathbf{Reg}$ as oracle and any $u \in \Sigma^*$ as input, M halts within $P(|\varphi|)(|u|)$ steps. Define *polynomial space* analogously by counting the number of visited cells on all (input, work, output and query) tapes.

Definition 2.10 (Type-two classes)**.**

1. We write **P** (resp. **PSPACE**) for the class of $(\mathbf{Reg}, \mathbf{Pred})$-problems solved by a polynomial-time (resp. space) machine.
2. We write **FP** (resp. **FPSPACE**) for the class of $(\mathbf{Reg}, \mathbf{Reg})$-problems solved by a polynomial-time (resp. space) machine.

Note that unlike the classes P and PSPACE of $(\Sigma^*, \{0,1\})$-problems, it is easy to separate, e.g., **P** and **PSPACE**, because a **PSPACE** machine can make exponentially many queries to the given oracle.

We can also define some close friends of **P** by analogy with the classes of string problems.

Definition 2.11 (Type-two classes, continued)**.**

1. A $(\mathbf{Reg}, \mathbf{Pred})$-problem A belongs to **NP** if there are a polynomial-time machine M and a second-order polynomial P such that $\operatorname{dom} A \subseteq \operatorname{dom} \underline{M}$ and for each $\varphi \in \operatorname{dom} A$, the function that maps each string u to

$$\begin{cases} 1 & \text{if } \underline{M}(\varphi)(u, v) = 1 \text{ for some } v \in \Sigma^{P(|\varphi|)(|u|)}, \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

is in $A[\varphi]$.

2. A $(\mathbf{Reg}, \mathbf{Reg})$-problem A belongs to **#P** if there are a polynomial-time machine M and a second-order polynomial P such that $\operatorname{dom} A \subseteq \operatorname{dom} \underline{M}$ and for each $\varphi \in \operatorname{dom} A$, the function that maps each string u to the number (written as a binary string of length $P(|\varphi|)(|u|) + 1$) of $v \in \varSigma^{P(|\varphi|)(|u|)}$ with $\underline{M}(\varphi)(u, v) = 1$ is in $A[\varphi]$.

We could alternatively characterize these classes **NP** and **#P** by defining "nondeterministic machines" appropriately and counting the number of computation paths, just as with the usual classes NP and #P of string functions.

In this thesis, we will only ask whether or not the problems belong to these complexity classes; we will not attempt to give a finer classification of problems by finding the specific second-order polynomial bounding their complexity. In fact, it is not entirely obvious how one should set about such a project. For a usual polynomial, its degree is the foremost measure indicating how large it is. For second order polynomial $P(|\varphi|)(|u|)$, the depth of the nesting of the function $|\varphi|$ is another factor. Investigating how the exact second-order polynomials compare with "intuitive" or "practical" understanding of efficiency is left for future work. Also left open is the formulation of analogues of sub-polynomial complexity classes, such as L, NL or NC.

2.2.3 Why we consider regular functions

The idea of using second-order polynomials as a bound on time and space comes from Kapron and Cook's characterization [KC96] of Mehlhorn's class [Meh76] of *polynomial-time computable operators*[1]. This is a class of (total) functionals $F\colon (\varSigma^* \to \varSigma^*) \times \varSigma^* \to \varSigma^*$, but they can be regarded as $F\colon (\varSigma^* \to \varSigma^*) \to (\varSigma^* \to \varSigma^*)$ by writing $F(\varphi)(x)$ instead of $F(\varphi, x)$. Kapron and Cook define the size $|\varphi|$ of $\varphi\colon \varSigma^* \to \varSigma^*$ by

$$|\varphi|(n) = \max_{|u| \le n} |\varphi(u)|, \qquad n \in \mathbb{N}. \tag{2.13}$$

Note that our definition of size for regular φ is a special case of this. They then defined the class of polynomial-time functionals in the way similar to Definition 2.10.2. We added **FPSPACE** and other classes by analogy.

Although their original class consists of functions over $\varSigma^* \to \varSigma^*$, we have restricted attention to regular functions. This is because, in order to obtain nice complexity notions, it seems convenient to be able to compute the resource bounds on a given input. Note that for usual (type-one) computation, it was easy, given x, to find the input length $|x|$ and thus the time bound $p(|x|)$ in unary notation for a fixed polynomial p. In contrast, finding the value (2.13) for a given φ in general requires exponentially many queries to φ and thus exponential time. For regular φ, we can easily find $|\varphi|(n)$ for each n, and thus the second-order polynomial $P(|\varphi|)(|u|)$ is a bound "time-constructible" from φ and u. Because of this property, most of the subtleties about the definition of basic feasible

[1] Kapron and Cook [KC96] call them *basic feasible functionals* or *basic polynomial-time functionals.*

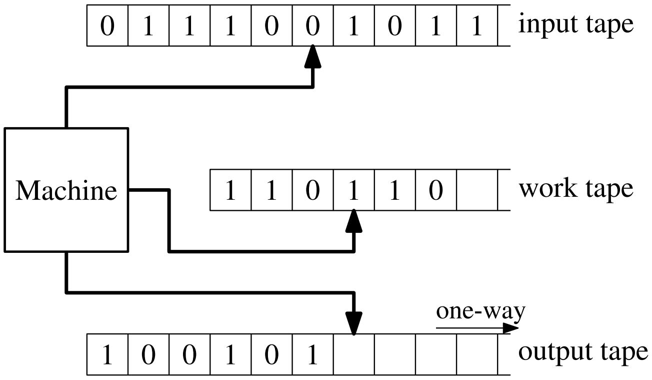


Figure 2.3: A machine converting infinite strings to infinite strings.

functionals (see [Set92, Set93]) disappear when we restrict them to **Reg**. This also gives us the standard complete problems (see Lemmas 2.15 and 2.16 below).

Imposing regularity is not much of a restriction for our purpose, because our intention is to use these functions as names of real numbers, sets and functions, and we can always declare that valid names are those that have been "padded" to be regular. More precisely, there is a polynomial-time machine that takes as oracles a possibly irregular function φ' and a regular function ψ dominating its length (i.e., $|\varphi'(u)| \leq |\psi|(|u|)$ for any string u), and delivers a regular function φ such that $|\varphi| = |\psi|$ and $\varphi(u) = \varphi'(u)\#\# \dots \#$. Thus we use φ, instead of φ', as the name. In any situation where φ' comes out of a machine whose time/space is polynomially bounded, this bound can serve as the length bound $|\psi|$.

On the other hand, we do not restrict ourselves too much by using the subset of **Reg** consisting of specific polynomial growth bound (this would be essentially equivalent, in our context, to working only with **Pred**). In some of the later applications where we encode real numbers and functions by **Reg**, the notion of the length defined above seems to be pertinent to the "size" of the encoded objects in a natural sense.

Comparison with the formulation by infinite strings Weihrauch's monograph [Wei00] uses infinite strings (elements of $\Sigma^{\mathbb{N}}$) instead of our regular functions. Computability of $(\Sigma^{\mathbb{N}}, \Sigma^{\mathbb{N}})$-functions is defined using what he calls the *type-two machine*. Just like the Turing machine, this machine has an input tape, an output tape, and a work tape, each of which is infinite to the right. We also assume that the output tape is one-way; that is, the only instruction for the output tape is "write $a \in \Sigma$ in the current cell and move the head to the right". The difference from the usual Turing machine computing (Σ^*, Σ^*)-functions is in the convention by which the machine reads the input and delivers the output. The input is now an infinite string $a_0 a_1 \dots \in \Sigma^{\mathbb{N}}$, and is written on the input tape before the computation starts (with the tape heads at the leftmost cell). We say the machine outputs an infinite string $b_0 b_1 \dots \in \Sigma^{\mathbb{N}}$ if it never halts and writes the string indefinitely (that is, for each $n \in \mathbb{N}$, it eventually writes $b_0 \dots b_{n-1}$ on the first n cells) on the output tape. (Figure 2.3). This defines what it means for the machine to solve a $(\Sigma^{\mathbb{N}}, \Sigma^{\mathbb{N}})$-problem.

By identifying each regular function $\varphi \in \mathbf{Pred}$ with the infinite string

$$p_\varphi = \varphi(\varepsilon)\#\varphi(0)\#\varphi(1)\#\varphi(00)\#\varphi(01)\#\varphi(10)\#\dots, \tag{2.14}$$

we can define the solubility (without time or space bound) of a $(\mathbf{Reg}, \mathbf{Reg})$-problem A by the computability of A' in Weihrauch's sense, where A' is the $(\Sigma^{\mathbb{N}}, \Sigma^{\mathbb{N}})$-problem defined by $\operatorname{dom} A' = \{\, p_\varphi : \varphi \in \operatorname{dom} A \,\}$ and $A'[p_\varphi] = \{\, p_\psi : \psi \in A[\varphi] \,\}$. It is routine to verify that this notion coincides with our class **FRec** defined using oracle Turing machines.

However, it is not clear how to define time or space complexity of computation by a type-two machine. Weihrauch [Wei00, Chapter 7] defines polynomial-time computability by requiring that there is a polynomial q such that for all infinite strings $p \in \Sigma^{\mathbb{N}}$ and any number $n \in \mathbb{N}$, the machine on input p finishes writing the first n symbols of the output within $q(n)$ steps (regardless of p). Notice that the above encoding (2.14) is not "efficient" enough to make the notion equivalent to ours (the index of the position at which the value $\varphi(u)$ appears in p_φ is exponential in $|u|$). As we will see in the following chapters, our formulation can be viewed as an extension of Weihrauch's, and this extension is essential for many of our applications.

2.3 Reductions and completeness

Here we define reductions between $(\mathbf{Reg}, \mathbf{Reg})$- and $(\mathbf{Reg}, \mathbf{Pred})$-problems and discuss hardness with respect to these reductions.

2.3.1 Reductions

Recall that the usual many-one reduction between (Σ^*, Σ^*)-problems A and B is defined as follows: we say that A many-one reduces to B (written $A \le^1_{\mathrm{mF}} B$) if there are (total) functions r, $t \in \mathsf{FP}$ such that for any $u \in \operatorname{dom} A$, we have $t(u) \in \operatorname{dom} B$ and $r(u, v) \in A[u]$ whenever $v \in B[t(u)]$—that is, we have a function t that converts an input for A to an input for B, and another function r that converts an output of B to an output of A (Figure 2.4, left). The many-one reduction $\le^1_{\mathrm{m}}$ between predicates ($(\Sigma^*, \{0, 1\})$-problems) is defined as the special case where where we do not convert the output, i.e., $r(u, v) = v$ (Figure 2.4, middle). These are special cases of the Turing reduction $A \le^1_{\mathrm{T}} B$ (Figure 2.4, right), which means that there is a function $r \in \mathsf{FP}$ such that $r(b)$ realizes A for any total function b realizing B. The reductions $\le^1_{\mathrm{mF}}$, $\le^1_{\mathrm{m}}$, $\le^1_{\mathrm{T}}$ are sometimes called the Karp, Levin and Cook reductions, respectively [Gol08, Section 2.2].

Since problems over **Reg** also get a function as input, the analogous definitions of reductions involve one more converter s:

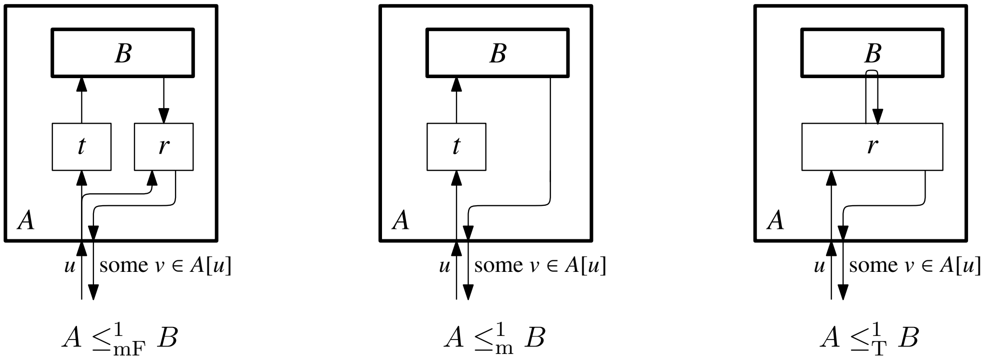

B
t
r
A
u
some $v \in A[u]$
$A \leq^1_{\mathrm{mF}} B$
B
t
A
u
some $v \in A[u]$
$A \leq^1_{\mathrm{m}} B$
B
r
A
u
some $v \in A[u]$
$A \leq^1_{\mathrm{T}} B$

Figure 2.4: Reductions between (Σ^*, Σ^*)- and $(\Sigma^*, \{0, 1\})$-problems A, B.

Definition 2.12 (Many-one reductions). 1. Let A and B be $(\mathbf{Reg}, \mathbf{Reg})$-problems. We say that A *many-one reduces* to B (written $A \leq^2_{\mathrm{mF}} B$) if there are functions r, s, $t \in \mathsf{FP}$ such that for any $\varphi \in \operatorname{dom} A$, we have $s(\varphi) \in \operatorname{dom} B$ and for any $\theta \in B[s(\varphi)]$, the function that maps each string x to $r(\varphi)(x, \theta(t(\varphi)(x)))$ is in $A[\varphi]$ (Figure 2.5, top left).

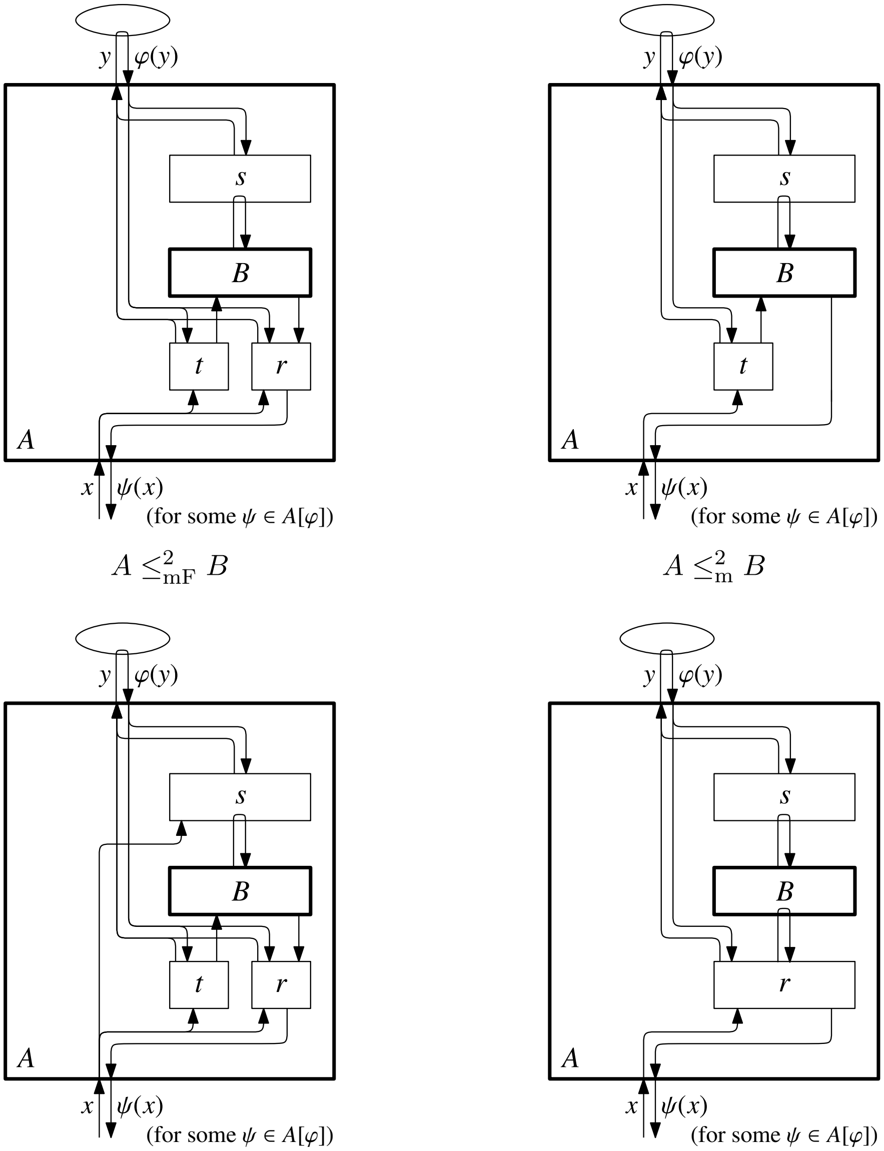
2. Let A and B be $(\mathbf{Reg}, \mathbf{Pred})$-problems. We write $A \leq^2_{\mathrm{m}} B$ if there are functions s, $t \in \mathsf{FP}$ such that for any $\varphi \in \operatorname{dom} A$, we have $s(\varphi) \in \operatorname{dom} B$ and for any $\theta \in B[s(\varphi)]$, the function $\theta \circ t(\varphi)$ is in $A[\varphi]$ (Figure 2.5, top right).

The design of these reductions is somewhat arbitrary. We chose these definitions simply because they are strong enough to make our examples (Theorems 4.14 and 4.25) complete with respect to them. What Beame et al. [BCE$^+$98] call the "many-one reduction" between type-two problems is slightly stronger than our $\leq^2_{\mathrm{mF}}$ in that it passes the string input x not only to t and r but also to s (Figure 2.5, bottom left). See the comment after Lemma 2.18 for the reason we did not choose this definition.

Another reasonable notion of reduction is the one on the bottom right of Figure 2.5:

Definition 2.13 (Turing reduction). Let A and B be $(\mathbf{Reg}, \mathbf{Reg})$-problems. We say that A *Turing reduces* to B (written $A \leq^2_{\mathrm{T}} B$) if there are functions r, $s \in \mathsf{FP}$ such that for any $\varphi \in \operatorname{dom} A$, we have $s(\varphi) \in \operatorname{dom} B$ and $r(\langle \varphi, \psi \rangle) \in A[\varphi]$ whenever $\psi \in B[s(\varphi)]$.

This is a polynomial-time version of the continuous reduction used by Weihrauch [Wei92] to compare the degrees of discontinuity of translators between real number representations (see Brattka and Gherardi [BG11] for a recent reference with an historical survey). Note that, while this reduction is somewhat analogous to the standard Turing reduction $\leq^1_{\mathrm{T}}$ of (Σ^*, Σ^*)-problems, it also formally resembles the definition of $\leq^1_{\mathrm{mF}}$. The many-one reduction $\leq^2_{\mathrm{mF}}$ is the special case of this reduction $\leq^2_{\mathrm{T}}$ where r can query ψ only once. Beame et al. [BCE$^+$98] define an even stronger "Turing reduction".



$A \leq^2_{\mathrm{mF}} B$

$A \leq^2_{\mathrm{m}} B$

A many-one reduces to B
in the sense of [BCE$^+$98].

$A \leq^2_{\mathrm{T}} B$

Figure 2.5: Reductions between (**Reg**, **Reg**)- and (**Reg**, **Pred**)-problems.

When C is one of the complexity classes defined so far, we write FP^{C} for the set of $(\mathbf{Reg}, \mathbf{Reg})$-problems A such that $A \leq^2_{\mathrm{T}} B$ for some $B \in \mathsf{C}$. (From the general point of view in complexity theory, it may be more appropriate to define what it means to put the oracle B to the machine M and then introduce the relativized classes D^B and D^{C} for the class D of *machines* M. But the above definition, treating the oracle as part of the problem instance, suffices for the classes we want to discuss. In particular, we will not mention such a relativized class for D other than FP in this thesis.)

Lemma 2.14. *1. Let A and B be $(\mathbf{Reg}, \mathbf{Pred})$-problems.*

- *$A \leq^2_{\mathrm{T}} B$ and $B \in \mathsf{P}$ imply $A \in \mathsf{P}$.*
- *$A \leq^2_{\mathrm{m}} B$ and $B \in \mathsf{NP}$ imply $A \in \mathsf{NP}$.*
- *$A \leq^2_{\mathrm{T}} B$ and $B \in \mathsf{PSPACE}$ imply $A \in \mathsf{PSPACE}$.*

2. Let A and B be $(\mathbf{Reg}, \mathbf{Reg})$-problems.

- *$A \leq^2_{\mathrm{T}} B$ and $B \in \mathsf{FP}$ imply $A \in \mathsf{FP}$.*
- *$A \leq^2_{\mathrm{T}} B$ and $B \in \mathsf{FP}^{\mathsf{NP}}$ imply $A \in \mathsf{FP}^{\mathsf{NP}}$.*
- *$A \leq^2_{\mathrm{T}} B$ and $B \in \mathsf{FPSPACE}$ imply $A \in \mathsf{FPSPACE}$.*

In fact, FP, $\mathsf{FP}^{\mathsf{NP}}$ and $\mathsf{FPSPACE}$ are the $\leq^2_{\mathrm{T}}$-closures of P, NP and PSPACE.

Now that we have the classes (Definition 2.10) and reductions (Definitions 2.12 and 2.13), we can talk about hardness. For a complexity class C and a reduction $\leq^2$, we say that a problem B is hard for C with respect to $\leq^2$ (or C-$\leq^2$-hard) if $A \leq^2 B$ for every $A \in \mathsf{C}$. It is said to be C-$\leq^2$-complete if moreover it is in C.

2.3.2 Complete problems

Here we list some NP- and PSPACE-$\leq^2_{\mathrm{m}}$-complete problems. The completeness proofs are all easily relativized versions of well-known NP- and PSPACE-$\leq^1_{\mathrm{m}}$-completeness.

We begin with NP-$\leq^2_{\mathrm{m}}$-complete problems. For a non-decreasing $(\mathbb{N}, \mathbb{N})$-function μ, define $\overline{\mu} \in \mathbf{Reg}$ by $\overline{\mu}(u) = 0^{\mu(|u|)}$. A *boolean formula involving a predicate symbol* is an expression built up inductively from boolean variables a_1, a_2, … using the connectives $f_{i_1} \wedge f_{i_2}$, $f_{i_1} \vee f_{i_2}$, $\neg f_{i_1}$ and $\Box(f_{i_1}, \dots, f_{i_n})$ (the arity n can vary) for any previously obtained formulas f_{i_1}, f_{i_2}, … . If u is such a boolean formula involving a predicate symbol and p is in $\mathbf{Pred}$, then we get a boolean formula u^p by interpreting $\Box$ as p.

Lemma 2.15. *The following partial functions* $\textsc{ntime}^2$, $\textsc{exist}^2$ *and* $\textsc{sat}^2$ *from* $\mathbf{Reg}$ *to* $\mathbf{Pred}$ *are* NP-$\leq^2_{\mathrm{m}}$*-complete:*

- $\operatorname{dom} \textsc{ntime}^2$ *consists of all quadruples $\langle M, \overline{p}, \overline{\mu}, \varphi \rangle$ such that M is a (program of an oracle Turing) machine, p and μ are non-decreasing $(\mathbb{N}, \mathbb{N})$-functions, $\varphi \in \mathbf{Reg}$, and for any $u \in \Sigma^*$ and $v \in \Sigma^{\leq p(|u|)}$, the machine M on oracle φ and input (u, v) halts in time $\mu(|u|)$ (this M is a string, so we encode it as the constant function taking this string as value). For any such quadruple and a string u, we have $\textsc{ntime}^2(\langle M, \overline{p}, \overline{\mu}, \varphi \rangle)(u) = 1$ if and only if $\underline{M}^{\varphi}(u, v) = 1$ for some v.*

- $\operatorname{dom} \textsc{exist}^2 = \mathbf{Pred}$. *For any $p \in \mathbf{Pred}$, $u \in \Sigma^*$ and $n \in \mathbb{N}$, we have $\textsc{exist}^2(p)(u, 0^n) = 1$ if and only if there is a string v of length n with $p(u, v) = 1$.*

- $\operatorname{dom} \textsc{sat}^2 = \mathbf{Pred}$. *For any $p \in \mathbf{Pred}$ and any string u, we have $\textsc{sat}^2(p)(u) = 1$ if and only if u is a boolean formula involving a predicate symbol and u^p is satisfiable.*

Proof. For $\textsc{ntime}^2$, let $A \in \textsf{NP}$. Then there are a polynomial-time machine M and a second-order polynomial P as in Definition 2.11. Let Q be its time bound (some second-order polynomial). To see that $A \leq^2_{\mathrm{m}} \textsc{ntime}^2$, define the functions s and t of Definition 2.12 by $s(\varphi) = \langle M, \overline{P(|\varphi|)}, \overline{Q(|\varphi|)}, \varphi\rangle$ and $t(\varphi)(u) = u$.

To see that $\textsc{ntime}^2 \leq^2_{\mathrm{m}} \textsc{exist}^2$, define the functions s and t of Definition 2.12 as follows. Let $s(\langle M, \overline{p}, \overline{\mu}, \varphi\rangle)$ be the function that maps each pair (u, v) to 1 if the machine M on oracle φ and input (u, v) halts in time $\mu(|u|)$ and outputs 1. Let $t(\langle M, \overline{\mu}, \varphi\rangle)(u)$ be the function that maps each string u to the pair $(u, 0^{p(|u|)})$.

To see that $\textsc{exist}^2 \leq^2_{\mathrm{m}} \textsc{sat}^2$, define the functions s and t of Definition 2.12 as follows. Let $s(p) = p$. Let $t(p)(u, 0^n)$ be the formula $p(u, a_1 a_2 \dots a_n)$, where the a_i are boolean variables. □

Note that in the proof of $A \leq^2_{\mathrm{m}} \textsc{ntime}^2$, we needed the regularity of φ in order to compute $\overline{P(|\varphi|)}$ and $\overline{Q(|\varphi|)}$ efficiently from given φ.

We can prove in almost the same way that the analogously defined counting problems are **#P**-$\leq^2_{\mathrm{mF}}$-complete.

Now we move on to **PSPACE**-$\leq^2_{\mathrm{m}}$-complete problems. If φ_p is a boolean formula involving a predicate symbol, then an expression of the form

$$Q_1 a_1.\, Q_2 a_2.\, Q_3 a_3 \dots Q_k a_k.\, \varphi_p(a_1, \dots, a_k), \tag{2.15}$$

where each Q_i is either $\forall$ or $\exists$, is called a *quantified boolean formula involving a predicate symbol.* Its truth value is determined in the obvious way relative to the predicate to be substituted.

Lemma 2.16. *The following partial functions $\textsc{space}^2$, $\textsc{power}^2$, $\textsc{qbf}^2$ from* **Reg** *to* **Pred** *are* **PSPACE**-$\leq^2_{\mathrm{m}}$*-complete:*

- $\operatorname{dom} \textsc{space}^2$ *consists of all triples $\langle M, \overline{\mu}, \varphi\rangle$ such that M is a (program of a) deterministic (oracle Turing) machine, μ is a non-decreasing $(\mathbb{N}, \mathbb{N})$-function, $\varphi \in \mathbf{Reg}$, and for any string u, the computation $M^\varphi(u)$ uses no more than $\mu(|u|)$ tape cells and either accepts or rejects (this M is a string, so we encode it as the constant function taking this string as value). For any such triple and a string u, we have $\textsc{space}^2(\langle M, \overline{\mu}, \varphi\rangle)(u) = 1$ if and only if $M^\varphi(u)$ accepts.*

- $\operatorname{dom} \textsc{power}^2$ *consists of all $f \in \mathbf{Reg}$ that are length-preserving (i.e., $|f| = id$). For any such f and a string u, we have $\textsc{power}^2(f)(u) = 1$ if and only if $f^{2^{|u|}}(u) = 0^{|u|}$, where we write f^k for f iterated k times.*

- $\operatorname{dom}\textsc{qbf}^2 = \mathbf{Pred}$. *For any* $p \in \mathbf{Pred}$ *and any string* u, *we have* $\textsc{qbf}^2(p)(u) = 1$ *if and only if* u *is a quantified boolean formula involving a predicate symbol and it is made true by* p.

Proof. The proof for $\textsc{space}^2$ is similar to that of the NP-$\leq^2_{\mathrm{m}}$-completeness of $\textsc{ntime}^2$.

We reduce $\textsc{space}^2$ to $\textsc{power}^2$. Fix a reasonable way to encode the configuration at any time in any computation of an oracle Turing machine by a string called the instantaneous description (ID). We may assume that if the computation uses at most k tape cells, then each configuration can be encoded by an ID (called its space-k ID) whose length is exactly $l(k)$, where l is a strictly increasing polynomial. We may also assume that the accepting configuration is unique and is represented by the space-k ID $0^{l(k)}$.

We may assume that $\textsc{space}^2$ is restricted to those triples $\langle M, \overline{\mu}, \varphi\rangle$ such that M^φ, on any input u, halts in exactly $2^{l(\mu(|u|))}$ steps. To show that this restricted $\textsc{space}^2$ reduces to $\textsc{power}^2$, we define the functions s and t of Definition 2.12 as follows. Define s by saying that $s(\langle M, \overline{\mu}, \varphi\rangle)$ is the length-preserving function that maps each ID of the machine M^φ to its successor ID (of the same length); if such a successor ID does not exist (this happens if M^φ has just used up too much space to have an ID of the same length), then the value is an ID corresponding to an irrevocable rejecting state. Define t by saying that $t(\langle M, \overline{\mu}, \varphi\rangle)(u)$ is the space-$\mu(|u|)$ initial ID of the machine M on input u.

We reduce $\textsc{power}^2$ to $\textsc{qbf}^2$. For each length-preserving string function f, define $[f] \in \mathbf{Pred}$ by $[f](v, w) = 1$ if and only if $w = f(v)$. Note that $[f^2](v, w)$ can be written

$$\exists x.\, \forall y.\, \forall z.\, \bigl(\bigl((y, z) = (v, x) \lor (y, z) = (x, w)\bigr) \longrightarrow [f](y, z)\bigr), \tag{2.16}$$

where x, y, z range over binary strings of the same length as v and w. Using this inductively, we can express $[f^{2^n}](v, w)$ as a quantified boolean formula involving $[f]$ whose length is bounded polynomially in n, $|v|$, $|w|$. Thus to get the functions s and t of Definition 2.12 for $\textsc{power}^2 \leq^2_{\mathrm{m}} \textsc{qbf}^2$, let $s(f) = [f]$ and let $t(f)$ be the formula that means $[f^{2^{|u|}}](u, 0^{|u|})$ relative to $[f]$. □

Note that, since $\mathsf{FP}^{\mathsf{NP}}$ and $\mathsf{FPSPACE}$ are the $\leq^2_{\mathrm{T}}$-closures of NP and PSPACE, the problems in Lemma 2.15 are $\mathsf{FP}^{\mathsf{NP}}$-$\leq^2_{\mathrm{T}}$-complete and the problems in Lemma 2.16 are $\mathsf{FPSPACE}$-$\leq^2_{\mathrm{T}}$-complete.

2.4 Effective and ineffective statements

In Chapter 4, we will study the complexity of objects such as an operator F that maps real functions to real functions. Some of such complexity issues have been addressed indirectly since the 1980s by assuming the real function f to be simple and then asking how complex $F(f)$ can be. In order to bridge between these two formulations, we show in this section how statements about the complexity of $(\mathbf{Reg}, \mathbf{Reg})$-problems F (discussed so far) imply statements about the complexity of the *values* of F.

The relations between the positive statements are easy to see. The type-two classes defined so far respect the corresponding type-one complexity classes:

Lemma 2.17. *1. Let* $(\mathbf{C}, \mathsf{B}, \mathsf{C})$ *be either* $(\mathbf{P}, \mathsf{FP}, \mathsf{P})$*,* $(\mathbf{NP}, \mathsf{FP}, \mathsf{NP})$ *or* $(\mathbf{PSPACE}, \mathsf{FPSPACE}, \mathsf{PSPACE}$ *Then partial functions in* $\mathbf{C}$ *map regular functions in* B *into* C*.*

2. Let $(\mathbf{C}, \mathsf{B}, \mathsf{C})$ *be either* $(\mathbf{FP}, \mathsf{FP}, \mathsf{FP})$*,* $(\mathbf{FP}^{\mathbf{NP}}, \mathsf{FP}, \mathsf{FP}^{\mathsf{NP}})$ *or* $(\mathbf{FPSPACE}, \mathsf{FPSPACE}, \mathsf{FPSPACE})$*. Then partial functions in* $\mathbf{C}$ *map regular functions in* B *into* C*.*

On the other hand, the following lemma states roughly that a $\mathbf{C}$-$\le^2$-hard $(\mathbf{Reg}, \mathbf{Reg})$-problem B maps some function $\psi \in \mathsf{FP} \cap \mathbf{Reg}$ to a C-$\le^1$-hard function, where C and $\le^1$ are the type-one versions of the class $\mathbf{C}$ and the reduction $\le^2$. But since $B[\psi]$ may consist of more than one function, we need to assert hardness for the (Σ^*, Σ^*)-problem $\bigcup(B[\psi])$ defined as follows: for a nonempty set F of (single-valued total) (X, Y)-functions, we write $\bigcup F$ to mean the (X, Y)-problem defined by $(\bigcup F)[x] = \{\, f(x) : f \in F \,\}$. Saying that the function $\bigcup F$ is hard is a stronger claim than saying that each of the functions in F is hard, because the former requires that one reduction work for all functions in F. We need to state the following lemma in this stronger form in order to derive Lemma 3.4 later.

Lemma 2.18. *1. Let* $(\mathbf{C}, \mathsf{C}, \le^2, \le^1)$ *be either* $(\mathbf{NP}, \mathsf{NP}, \le^2_{\mathrm{m}}, \le^1_{\mathrm{m}})$*,* $(\mathbf{PSPACE}, \mathsf{PSPACE}, \le^2_{\mathrm{T}}, \le^1_{\mathrm{T}})$ *or* $(\mathbf{PSPACE}, \mathsf{PSPACE}, \le^2_{\mathrm{m}}, \le^1_{\mathrm{m}})$*. If B is a* $\mathbf{C}$*-*$\le^2$*-hard* $(\mathbf{Reg}, \mathbf{Pred})$*-problem, then there is* $\psi \in \mathsf{FP} \cap \operatorname{dom} B$ *such that* $\bigcup(B[\psi])$ *is* C*-*$\le^1$*-hard.*

2. Let $(\mathbf{C}, \mathsf{C})$ *be either* $(\mathbf{FP}^{\mathbf{NP}}, \mathsf{FP}^{\mathsf{NP}})$ *or* $(\mathbf{FPSPACE}, \mathsf{FPSPACE})$ *and let* $(\le^2, \le^1)$ *be either* $(\le^2_{\mathrm{T}}, \le^1_{\mathrm{T}})$ *or* $(\le^2_{\mathrm{mF}}, \le^1_{\mathrm{mF}})$*. If B is a* $\mathbf{C}$*-*$\le^2$*-hard* $(\mathbf{Reg}, \mathbf{Reg})$*-problem, then there is* $\psi \in \mathsf{FP} \cap \operatorname{dom} B$ *such that* $\bigcup(B[\psi])$ *is* C*-*$\le^1$*-hard.*

Proof. There is a function $A \in \mathbf{C}$ that maps some function $\varphi \in \mathsf{FP} \cap \mathbf{Reg}$ to a C-$\le^1$-complete function. Since B is $\mathbf{C}$-$\le^2$-hard, there are functions r, s, $t \in \mathbf{FP}$ as in Definition 2.12. By Lemma 2.17, we have $r(\varphi)$, $s(\varphi)$, $t(\varphi) \in \mathsf{FP}$. Let $\psi = s(\varphi)$. Since $r(\varphi)$ and $t(\varphi)$ give a reduction $A(\varphi) \le^1 \bigcup(B[\psi])$, and $A(\varphi)$ is C-$\le^1$-complete, $\bigcup(B[\psi])$ is also C-$\le^1$-hard. □

We note that Lemma 2.18 would not have been true, if in the definition of reductions we had fed s with the string input as Beame et al. [BCE⁺98] do (see the comment after Definition 2.12). For let $\le^2_*$ be the reduction which is like $\le^2$ but feeds s with the string input, and let B be a $\mathbf{C}$-$\le^2$-complete problem. Then the problem B' defined by

$$\operatorname{dom} B' = \{\, \langle \mathrm{const}_u, \varphi \rangle : u \in \Sigma^*,\ \varphi \in \operatorname{dom} B \,\}, \tag{2.17}$$

$$B'[\langle \mathrm{const}_u, \varphi \rangle] = \{\, \mathrm{const}_{\psi(u)} : \psi \in B[\varphi] \,\}, \tag{2.18}$$

where $\mathrm{const}_u \in \mathbf{Reg}$ denotes the constant function with value u, is $\mathbf{C}$-$\le^2_*$-complete by the modified reduction where "s does the jobs that t and s used to do". Yet each one of the values of B' is a constant function.

3 Representations

This chapter discusses how to apply the complexity theory over regular functions developed in the previous chapter to various mathematical objects. Section 3.1 introduces the general framework to talk about computation of objects encoded by regular functions via *representations*. That is, we will extend our complexity notions about $(\mathbf{Reg}, \mathbf{Reg})$-problems from Chapter 2 to (X, Y)-problems through representations of X and Y by $\mathbf{Reg}$.

Formally, a *representation* γ of a set X is a partial $(\mathbf{Reg}, X)$-function which is *surjective*—that is, for each $x \in X$, there is at least one $\varphi \in \mathbf{Reg}$ with $\gamma(\varphi) = x$. For the applications considered in this thesis, X will be equipped with a natural topology, and a good representation of it is one that captures how the points in X can be approximated (more on this in Section 3.2).

For example, a representation $\rho_{\mathbb{R}}$ of the real numbers $\mathbb{R}$ is defined as follows. First, we fix an encoding of a dense subset of $\mathbb{R}$ whose elements are used to approximate real numbers. For each $n \in \mathbb{N}$, let $\mathbb{D}_n$ denote the set of strings of the form

$$sx/1\underbrace{00\ldots0}_{n}, \tag{3.1}$$

where $s \in \{+, -\}$ and $x \in \{0,1\}^*$. Let $\mathbb{D} = \bigcup_{n \in \mathbb{N}} \mathbb{D}_n$. A string in $\mathbb{D}$ *encodes* a number in the obvious sense—read (3.1) as a fraction whose numerator and denominator are integers written in binary with leading zeros allowed. We write $[\![u]\!]$ for the number encoded by $u \in \mathbb{D}$. The numbers that can be encoded in this way are called *dyadic* numbers. We define the representation $\rho_{\mathbb{R}}$ of $\mathbb{R}$ by setting $\rho_{\mathbb{R}}(\varphi) = x$ when $\varphi(0^i) \in \mathbb{D}$ and $|[\![\varphi(0^i)]\!] - x| < 2^{-i}$ for each $i \in \mathbb{N}$.

Once we fix representations γ and δ of sets X and Y, we can talk about solving a (X, Y)-problem with respect to (γ, δ): an (X, Y)-problem A is (γ, δ)-solved by (an oracle Turing) machine M if for any $x \in \operatorname{dom} A$ and any φ with $\gamma(\varphi) = x$, the outcome $\psi = \underline{M}(\varphi)$ satisfies $\delta(\psi) \in A[x]$ (Definition 3.1.3 and Lemma 3.2). This gives rise to the easiness and hardness notions for (X, Y)-problems. Of course, they depend sensitively on the choice of representations. In Sections 3.3 and 3.4, we introduce suitable representations for specific objects of interest, including real numbers, sets, functions, and spaces obtained from them by basic set-theoretic operations.

Note that we will represent all these spaces by $\mathbf{Reg}$—for example, our representation $\delta_{\square}$ of the space $\mathrm{C}[0,1]$ of continuous functions $f\colon [0,1] \to \mathbb{R}$ will still encode them by regular functions (from strings to strings), even though real functions may seem to be "of higher type" than real numbers. This is because we know less about the complexity of types higher than two than we know (from Chapter 2) about type-two. Nevertheless, we will see

(Theorem 3.33) that our representation $\delta_\square$ of real functions is the one which, in a sense, canonically arise from $\rho_{\mathbb{R}}$.

3.1 Computation through representations

As mentioned above, a *representation* γ of a set X is a partial $(\mathbf{Reg}, X)$-function which is surjective. When $\gamma(\varphi) = x$, we say that φ is a γ-*name* of x. We write γ^{-1} for the total $(X, \mathbf{Reg})$-problem defined by $\gamma^{-1}[x] = \{\, \varphi \in \operatorname{dom}\gamma : \gamma(\varphi) = x \,\}$. Note that $\gamma \circ \gamma^{-1} = \mathrm{id}$.

The following definition extends the terminology about $(\mathbf{Reg}, \mathbf{Reg})$-problems from Chapter 2 to (X, Y)-problems by regarding an (X, Y)-problem A as the $(\mathbf{Reg}, \mathbf{Reg})$-problem $\delta^{-1} \circ A \circ \gamma$, where γ and δ are representations of X and Y.

Definition 3.1 (Computation relative to representations)**.** Let γ and δ be representations of sets X and Y, respectively. An (X, Y)-problem A is said to be

1. (γ, δ)-*realized* by a $(\mathbf{Reg}, \mathbf{Reg})$-problem F, if F realizes $\delta^{-1} \circ A \circ \gamma$.
2. (γ, δ)-*continuous*, if $\delta^{-1} \circ A \circ \gamma$ is continuous.
3. (γ, δ)-*solved* by a machine M, if M solves $\delta^{-1} \circ A \circ \gamma$.
4. in (γ, δ)-C, if $\delta^{-1} \circ A \circ \gamma$ is in C, where C is one of the classes of $(\mathbf{Reg}, \mathbf{Reg})$-problems defined in Section 2.2.2. This C can be a class of $(\mathbf{Reg}, \mathbf{Pred})$-problems if $\operatorname{dom}\delta \subseteq \mathbf{Pred}$.
5. (γ, δ)-C-$\le^2$-*hard*, if $\delta^{-1} \circ A \circ \gamma$ is C-$\le^2$-hard, where C is one of the classes of $(\mathbf{Reg}, \mathbf{Reg})$-problems defined in Section 2.2.2, and $\le^2$ is one of the reductions defined in Section 2.3.1.
6. (γ, δ)-C-$\le^2$-*complete*, if it belongs to (γ, δ)-C and is (γ, δ)-C-$\le^2$-*hard*.

Recall that the composition of problems was defined by (2.1) and (2.2); that is,

$$\operatorname{dom}(\delta^{-1} \circ A \circ \gamma) = \{\, \varphi \in \operatorname{dom}\gamma : \gamma(\varphi) \in \operatorname{dom} A \,\}, \tag{3.2}$$

and for each φ in this set,

$$(\delta^{-1} \circ A \circ \gamma)[\varphi] = \{\, \psi \in \operatorname{dom}\delta : \delta(\psi) \in A[\gamma(\varphi)] \,\}. \tag{3.3}$$

The following is easy to prove using Lemma 2.2 and the facts that $\gamma \circ \gamma^{-1} = \delta \circ \delta^{-1} = \mathrm{id}$ and that $\gamma^{-1} \circ \gamma$ and $\delta^{-1} \circ \delta$ are realized by id.

Lemma 3.2. *Let γ and δ be representations of sets X and Y, respectively. Let A be an (X, Y)-problem and F be a* $(\mathbf{Reg}, \mathbf{Reg})$*-problem (Figure 3.1). The following are equivalent:*

- *F realizes $\delta^{-1} \circ A \circ \gamma$ (that is, A is (γ, δ)-realized by F as per Definition 3.1.1).*

$$\begin{tikzcd}
\mathbf{Reg} \arrow[r, "F"] \arrow[d, "\gamma"'] & \mathbf{Reg} \arrow[d, "\delta"] \\
X \arrow[r, "A"'] & Y
\end{tikzcd}$$

Figure 3.1: A $(\mathbf{Reg}, \mathbf{Reg})$-problem F is said to (γ, δ)-realize a (X, Y)-problem A if F realizes $\delta^{-1} \circ A \circ \gamma$, or equivalently, $\delta \circ F$ realizes $A \circ \gamma$.

- *$\delta \circ F$ realizes $A \circ \gamma$.*

- *$\delta \circ F \circ \gamma^{-1}$ realizes A.*

Our Definition 3.1.1 is equivalent to the one in Weihrauch [Wei00, Definition 3.1.3.4] which formulates (γ, δ)-realization[1] by the second characterization above. Weihrauch then defines (γ, δ)-continuity (resp. (γ, δ)-computability) of A by the existence of a continuous (resp. computable) partial function that (γ, δ)-realizes A. These definitions are also equivalent to the one obtained from our Definitions 3.1.2 and 3.1.3. For continuity this is because of Corollary 2.8.

Thus, saying that A is (γ, δ)-solved by a machine M means that whenever M is given a γ-name of an element $x \in \operatorname{dom} A$ as oracle, it must output some δ-name of some element of $A[x]$.

Effective and ineffective statements In Section 2.4, we saw for $(\mathbf{Reg}, \mathbf{Reg})$-problems F that effective statements (about the complexity of F) implies ineffective statements (about the complexity of the values of F). Here we discuss how this implication carry through for computation on represented spaces.

We need some terminology for the ineffective statements. When γ is a representation of a set X, we write γ-C (where C is a type-one complexity class) for the set of elements $x \in X$ is that have a γ-name in C. We say that x is γ-C-$\leq^1$-*hard* (where $\leq^1$ is one of the type-one reductions $\leq^1_{\mathrm{mF}}$, $\leq^1_{\mathrm{m}}$ and $\leq^1_{\mathrm{T}}$) if $\bigcup(\gamma^{-1}[x])$ (recall that $\bigcup$ was defined before Lemma 2.18) is C-$\leq^1$-hard. Now we have the represented versions of Lemmas 2.17 and 2.18:

Lemma 3.3. *Let γ and δ be representations of sets X and Y, respectively.*

1. *Suppose that* $\operatorname{dom} \delta \subseteq \mathbf{Pred}$ *and let $(\boldsymbol{\mathsf{C}}, \mathsf{B}, \mathsf{C})$ be as in 2.17.1. Then a partial (X, Y)-function F in (γ, δ)-$\boldsymbol{\mathsf{C}}$ maps elements of γ-$\mathsf{B} \cap \operatorname{dom} F$ into δ-C.*

2. *Let $(\boldsymbol{\mathsf{C}}, \mathsf{B}, \mathsf{C})$ be as in 2.17.2. A partial (X, Y)-function F in (γ, δ)-$\boldsymbol{\mathsf{C}}$ maps elements of γ-$\mathsf{B} \cap \operatorname{dom} F$ into δ-C.*

[1] Weihrauch [Wei00] uses $\Sigma^{\mathbb{N}}$ instead of our $\mathbf{Reg}$, see Section 2.2.3. Also, he uses this terminology only when F (the realizer) is single-valued.

Proof. Let $x \in \gamma\text{-}\mathsf{B} \cap \operatorname{dom} F$ be arbitrary. Then x has a γ-name $\varphi \in \mathsf{B}$. By assumption, $\delta^{-1} \circ F \circ \gamma$ belongs to $\mathbf{C}$. Since $\mathbf{C}$ is a class defined by machine computation, there is a partial function $G \in \mathbf{C}$ that realizes $\delta^{-1} \circ F \circ \gamma$. By Lemma 2.17, we have $G(\varphi) \in \mathsf{C}$. Since $G(\varphi) \in (\delta^{-1} \circ F \circ \gamma)[\varphi] = \delta^{-1}[F(x)]$, we have $F(x) \in \delta\text{-}\mathsf{C}$. □

Lemma 3.4. *Let γ and δ be representations of sets X and Y, respectively.*

1. *Suppose that* $\operatorname{dom} \delta \subseteq \mathbf{Pred}$ *and let $(\mathbf{C}, \mathsf{C}, \le^2, \le^1)$ be as in 2.18.1. Then a (γ, δ)-$\mathbf{C}$-$\le^2$-hard partial function from X to Y maps some element of γ-FP to a δ-C-$\le^1$-hard element of Y.*

2. *Let $(\mathbf{C}, \mathsf{C}, \le^2, \le^1)$ be as in 2.18.2. A (γ, δ)-$\mathbf{C}$-$\le^2$-hard partial function from X to Y maps some element of γ-FP to a δ-C-$\le^1$-hard element of Y.*

Proof. Suppose that A is a (γ, δ)-$\mathbf{C}$-$\le^2$-hard partial function. This means that $\delta^{-1} \circ A \circ \gamma$ is $\mathbf{C}$-$\le^2$-hard. By Lemma 2.18, there is $\psi \in \mathsf{FP} \cap \operatorname{dom}(A \circ \gamma)$ such that $\bigcup(\delta^{-1}[A(\gamma(\varphi))])$ is C-$\le^1$-hard. Thus, A maps an element $\gamma(\varphi)$ of γ-FP to a δ-C-$\le^1$-hard element $A(\gamma(\varphi))$. □

These lemmas will be used later in Chapter 4 to derive the ineffective theorems from their effective versions.

3.2 Comparing representations

Here we study how the class (γ, δ)-$\mathbf{C}$ and the notion of (γ, δ)-$\mathbf{C}$-$\le^2$-hardness depend on the choice of representations γ and δ.

3.2.1 Translation and equivalence

We can compare two representations γ and δ of the same set by asking whether γ-names can be translated to δ-names by a continuous, computable, or polynomial-time computable partial function:

Definition 3.5 (Translation)**.** Let γ and δ be two representations of the same set X. We write[2]

1. $\gamma \le_{\mathrm{cont}} \delta$, if $\delta^{-1} \circ \gamma$ is continuous.

2. $\gamma \le_{\mathbf{FRec}} \delta$, if $\delta^{-1} \circ \gamma$ is in $\mathbf{FRec}$.

3. $\gamma \le_{\mathbf{FP}} \delta$, if $\delta^{-1} \circ \gamma$ is in $\mathbf{FP}$.

Thus, $\gamma \le_{\mathrm{cont}} \delta$ (resp. $\gamma \le_{\mathbf{FRec}} \delta$, $\gamma \le_{\mathbf{FP}} \delta$) if there is a continuous $(\mathbf{Reg}, \mathbf{Reg})$-problem F (resp. in $\mathbf{FRec}$, in $\mathbf{FP}$) that *Turing translates* γ to δ in the sense that $\delta \circ F$ realizes γ

$$\begin{tikzcd}
\mathbf{Reg} \arrow[r, "F"] \arrow[dr, "\gamma"'] & \mathbf{Reg} \arrow[d, "\delta"] \\
 & X
\end{tikzcd}$$

Figure 3.2: Function F translating γ to δ.

(Figure 3.2), or equivalently, the identity function id is (γ, δ)-solved by F. Because of Corollary 2.8, it makes no difference whether we require this F to be single-valued. The existence of such a translation may be interpreted as γ being "more informative" or "less general" than δ.

Since the classes we have defined so far are closed under polynomial-time Turing reduction, we have the following:

Lemma 3.6. *Let C be one of the classes of $(\mathbf{Reg}, \mathbf{Reg})$-problems defined in Chapter 2. Let γ and γ' be representations of a set X, and δ and δ' be representations of a set Y. If $\gamma' \leq_{\mathsf{FP}} \gamma$ and $\delta \leq_{\mathsf{FP}} \delta'$, then (γ, δ)-$\mathsf{C} \subseteq (\gamma', \delta')$-$\mathsf{C}$.*

We write $\gamma \equiv_{\mathsf{FP}} \gamma'$ if $\gamma \leq_{\mathsf{FP}} \gamma'$ and $\gamma' \leq_{\mathsf{FP}} \gamma$ (and likewise for $\equiv_{\mathrm{cont}}$ and $\equiv_{\mathsf{FRec}}$). By Lemma 3.6, the class (γ, δ)-C is invariant under replacing γ or δ with $\equiv_{\mathsf{FP}}$-equivalent representations. This is the polynomial-time analogue of the similar observation about $\equiv_{\mathsf{FRec}}$-equivalent representations [Wei00, Corollary 3.1.9]

The next lemma shows how the notion of (γ, δ)-C-$\leq^2_{\mathrm{T}}$-hardness is kept by different choices of representations γ and δ.

Lemma 3.7. *Let C be one of the classes of $(\mathbf{Reg}, \mathbf{Reg})$-problems defined in Chapter 2. Let γ and γ' be representations of a set X, and δ and δ' be representations of a set Y. If $\gamma \leq_{\mathsf{FP}} \gamma'$ and $\delta' \leq_{\mathsf{FP}} \delta$, then a (γ, δ)-C-$\leq^2_{\mathrm{T}}$-hard (X, Y)-problem is (γ', δ')-C-$\leq^2_{\mathrm{T}}$-hard.*

Thus the notion of (γ, δ)-C-$\leq^2_{\mathrm{T}}$-completeness is invariant when replacing γ or δ by $\equiv_{\mathsf{FP}}$-equivalent representations. The hardness in Lemma 3.7 needs to be the one defined by the reduction $\leq^2_{\mathrm{T}}$, because our translation $\leq_{\mathsf{FP}}$ (Definition 3.5) does a conversion similar to Turing reduction. If we want to get an analogous result for the weaker reductions $\leq^2_{\mathrm{mF}}$ and $\leq^2_{\mathrm{m}}$, we need to replace $\leq_{\mathsf{FP}}$ in the assumption by a more restrictive kind of "many-one" translation, but we leave it to the reader.

By Lemmas 3.6 and 3.7, $\equiv_{\mathsf{FP}}$-equivalent representations define the same notions of computability and hardness (as long as we are interested in classes above FP). The question, then, is choosing the "right" $\equiv_{\mathsf{FP}}$-equivalence class. A first test of a good representation is that it is compatible with the topology of the represented space in the sense explained next.

[2] The translations $\leq_{\mathrm{cont}}$ and $\leq_{\mathsf{FRec}}$ defined here are essentially the same things as $\leq_{\mathrm{t}}$ and $\leq$ of [Wei00, Definition 2.3.2].

3.2.2 Admissible representations

Although by definition a representation of a set X is any surjective partial $(\mathbf{Reg}, X)$-function, only a handful of them are meaningful. Recall from Chapter 2 that a regular function $\varphi \in \mathbf{Reg}$ can be accessed only through querying. Thus, a reasonable representation γ would be such that partial information on the name φ (i.e., its values at finitely many strings) reveals some meaningful information about the point $\gamma(\varphi)$. For example, a nice representation γ of $\mathbb{R}$ would be such that reading a finite portion of a name φ of a number $x \in \mathbb{R}$ would allow us to narrow x down into some neighbourhood of x. This motivates us to consider the represented set X as a topological space and to look for a representation that matches this topology.

A topological space X is said to be *second countable* if it has a countable base. It is T_0 if any two distinct points have different families of open neighbourhoods (i.e., the set of all open sets containing x is different for different $x \in X$). For example, the set $\mathbf{Reg}$ of regular functions is a second countable T_0 space. (Recall from Section 2.2 that we define the topology of this space by the base $\{\, B_k : k \in \Lambda \,\}$, where Λ is the set of partial (Σ^*, Σ^*)-functions with finite promise, and B_k is the set of regular functions realizing k.)

We will be henceforth interested only in second countable T_0 spaces. We obtain a natural representation of such a space if we fix the encoding of the elements of a (countable) base:

Definition 3.8 (Effective topological spaces)**.** A pair (X, ν) is called an *effective topological space* if ν is a partial surjective $(\Sigma^*, \mathfrak{B})$-function, where $\mathfrak{B}$ is a family of subsets of X and the space X whose topology is defined by the open base $\mathfrak{B}$ is T_0. The *standard representation* $\hat{\delta} = \delta_{(X,\nu)}$ of X is then defined by saying that a regular function $\varphi \in \mathbf{Reg}$ is a $\hat{\delta}$-name of a point $x \in X$ if and only if

$$\{\, \nu\big(\varphi(u)\big) : u \in \Sigma^* \,\} = \{\, B \in \mathfrak{B} : x \in B \,\}. \tag{3.4}$$

Thus, a $\hat{\delta}$-name of a point $x \in X$ is a list of all basic open sets containing x. This representation $\hat{\delta}$ is well-defined (no two elements of X get the same $\hat{\delta}$-name) because X is T_0.

An example of an effective topological space is $(\mathbb{R}, \nu)$, where ν is a reasonable encoding of all balls whose centre and radius are rational numbers. The standard representation $\hat{\delta}$ for this space is the one where a $\hat{\delta}$-name of $x \in \mathbb{R}$ is an infinite list of all such rational balls. It is easy to see that $\hat{\delta}$ is $\equiv_{\mathsf{FRec}}$-equivalent (and hence $\equiv_{\mathrm{cont}}$-equivalent) to the representation $\rho_{\mathbb{R}}$ defined at the beginning of this chapter, but it encodes information "less efficiently" so that $\rho_{\mathbb{R}} \le_{\mathsf{FP}} \hat{\delta}$ but not $\hat{\delta} \le_{\mathsf{FP}} \rho_{\mathbb{R}}$.

Lemma 3.9 (Essentially [Wei00, Lemma 3.2.5])**.** *Let $\hat{\delta}$ be the standard representation of an effective topological space (X, ν) (Definition 3.8). Then both $\hat{\delta}$ and $\hat{\delta}^{-1}$ are continuous.*

Proof. Let $\varphi \in \mathbf{Reg}$ be a $\hat{\delta}$-name of $x \in X$. Then $\hat{\delta}$ is continuous at (φ, x), because, for any basic open set $\nu(v)$ containing x, its preimage

$$\hat{\delta}^{-1}\big(\nu(v)\big) = \bigcup_{u \in \Sigma^*} \{\, \psi \in \operatorname{dom} \hat{\delta} : \psi(u) = v \,\} \tag{3.5}$$

is open in $\operatorname{dom}\hat{\delta}$. Also, $\hat{\delta}^{-1}$ is continuous at (x, φ), because, for any basic open set B_k containing φ (see Section 2.2.1 for the notation of a base of **Reg**), its preimage

$$(\hat{\delta}^{-1})^{-1}B_k = \bigcap_{u \in \operatorname{dom} k} \nu\bigl(\varphi(u)\bigr) \tag{3.6}$$

is open. Thus $\hat{\delta}$ and $\hat{\delta}^{-1}$ are continuous (in fact, strongly continuous). □

The following definition formulates the desirable property of a representation from the topological point of view.

Definition 3.10 (Admissible representation[3]). Let X be a second-countable T_0 space. A representation δ of X is *admissible* if it is continuous and $\gamma \le_{\mathrm{cont}} \delta$ for every continuous representation γ of X.

Such a representation exists by Lemma 3.9 and the following.

Lemma 3.11. *A representation δ of a second countable T_0 space is admissible if and only if both δ and δ^{-1} are continuous.*

Proof. Suppose that a representation δ of a second-countable T_0 space X and its inverse δ^{-1} are both continuous, and let γ be another continuous representation of X. Then $\delta^{-1} \circ \gamma$ is continuous, and this was the definition of $\gamma \le_{\mathrm{cont}} \delta$.

Conversely, suppose that δ is admissible (and hence continuous). The whole space can be regarded as an effective topological space by fixing a notation of a countable base, so let $\hat{\delta}$ be its standard representation (Definition 3.8). To see that δ^{-1} is continuous, we use the fact that $\hat{\delta} \le_{\mathrm{cont}} \delta$, that is, $\delta^{-1} \circ \hat{\delta}$ is continuous. Composing from the right the problem $\hat{\delta}^{-1}$, which is continuous by Lemma 3.9, we conclude that δ^{-1} is continuous. □

Theorem 3.12. *Let γ and δ be admissible representations of second-countable T_0 spaces X and Y, respectively. An (X, Y)-problem is continuous if and only if it is (γ, δ)-continuous.*

Proof. Let A be an (X, Y)-problem. Recall that (δ, γ)-continuity of A means the continuity of $\delta^{-1} \circ A \circ \gamma$ by definition. The claim follows from Lemmas 2.5 and 3.11 and the fact that $A = \delta \circ \delta^{-1} \circ A \circ \gamma \circ \gamma^{-1}$. □

In summary, the admissible representations constitute a unique $\equiv_{\mathrm{cont}}$-equivalence class of representations that are compatible with the topology of the given second-countable T_0 space. One of the representations in this equivalence class is the standard representation, which we obtain as soon as we fix an encoding of a countable base of the space (i.e., specify

[3] Weihrauch [Wei00, Definition 3.2.7] defines admissible representations as those $\equiv_{\mathrm{cont}}$-equivalent to the standard representation, and then derives this characterization as a theorem [Wei00, Theorem 3.2.9]. Schröder [Sch02] chooses Definition 3.10 as the basis of his generalization of admissibility to non-countably based spaces.

an effective metric space). The (smaller) $\equiv_{\mathsf{FRec}}$-equivalence class of the standard representation consists of the representations which define the common computability pertinent to this specific encoding of the base.

We still need to choose an even smaller $\equiv_{\mathsf{FP}}$-equivalence class which defines meaningful complexity notions. This does not seem to be possible in complete generality. Note that the standard representation $\hat{\delta}$ (Definition 3.8) of an effective metric space (say $\mathbb{R}$) is useless for discussing $\hat{\delta}$-FP: a $\hat{\delta}$-name φ of $t \in \mathbb{R}$ is such that the values $\varphi(u)$ as u ranges over all strings list all basic open sets containing t, but there is no rule about how this listing is ordered, so it does not make much sense to try to bound the time to get $\varphi(u)$ in terms of $|u|$. Thus we will need to design a "less redundant" representation for each application. We will return to this issue as we discuss some specific spaces in Section 3.4.

3.3 General constructions of representations

In this section, we present canonical methods to construct representations from given ones.

3.3.1 Finite objects

As we explained above, we use the elements of **Reg** as names of various objects, such as real numbers, that cannot be encoded by Σ^*. But some mathematical problems may partly involve discrete objects as well, such as natural numbers, that could be encoded by Σ^*. Let ν be a *notation* of a set X, i.e., a surjective single-valued partial (Σ^*, X)-function (X is of course countable if there is such a notation). Then we define the representation κ_ν of X by

$$\operatorname{dom} \kappa_\nu = \{ \operatorname{const}_u : u \in \operatorname{dom} \nu \}, \qquad \kappa_\nu(\operatorname{const}_u) = \nu(u), \tag{3.7}$$

where const_u is the (total) constant (Σ^*, Σ^*)-function taking value u.

This allows us to discuss the complexity in the conventional sense within our framework. Note, for example, that an (X, Y)-problem is in (κ_μ, κ_ν)-FP, where μ and ν are notations of X and Y, respectively, if and only if it can be solved in polynomial time (with respect to the notations μ and ν) in the usual sense.

3.3.2 Pairs and sequences

For representations γ_0 and γ_1 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

$$\operatorname{dom}[\gamma_0, \gamma_1] = \{ \langle \varphi_0, \varphi_1 \rangle : \varphi_0 \in \operatorname{dom} \gamma_0,\ \varphi_1 \in \operatorname{dom} \gamma_1 \}, \tag{3.8}$$

$$[\gamma_0, \gamma_1](\langle \varphi_0, \varphi_1 \rangle) = (\gamma_0(\varphi_0), \gamma_1(\varphi_1)) \tag{3.9}$$

using the pairing function defined at the beginning of 2.2. This can be extended to finitely many sets $X_0, \ldots, X_{n-1}$ by $[\gamma_0, \gamma_1, \ldots, \gamma_{n-1}] = [\ldots[[\gamma_0, \gamma_1], \gamma_2], \ldots, \gamma_{n-1}]$. We write γ^n for the representation $[\gamma, \ldots, \gamma]$ of X^n.

For infinitely many regular functions φ_0, φ_1, ..., we define the total (Σ^*, Σ^*)-function $\langle \varphi_i \rangle_{i \in \mathbb{N}}$ by $\langle \varphi_i \rangle_{i \in \mathbb{N}}(0^i 1u) = \varphi_i(u)$ for each $i \in \mathbb{N}$. This is not a regular function, but $|\langle \varphi_i \rangle_{i \in \mathbb{N}}|(n) \le \max_{i<n}(i + 1 + |\varphi_i|(n - 1 - i))$, so given the φ_i, it can be made regular by padding the values in some reasonable way. Thus for a representation γ of a set X, we define $\gamma^{\mathbb{N}}$ to be the representation of $X^{\mathbb{N}}$ given by $\operatorname{dom} \gamma^{\mathbb{N}} = \{ \langle \varphi_i \rangle_{i \in \mathbb{N}} : \varphi_0, \varphi_1, \ldots \in \operatorname{dom} \gamma \}$ and $\gamma^{\mathbb{N}}(\langle \varphi_i \rangle_{i \in \mathbb{N}}) = (\gamma(\varphi_i))_i$.

3.3.3 Subspaces and quotients

Let γ be a representation of space X. The representation $\gamma|^{X'}$ of a subspace $X' \subseteq X$ (equipped with the relative topology) is defined by $\operatorname{dom} \gamma|^{X'} = \{ \varphi \in \operatorname{dom} \gamma : \gamma(\varphi) \in X' \}$ and $\gamma|^{X'}(\varphi) = \gamma(\varphi)$.

Let $\sim$ be an equivalence relation on X. The representation $\gamma/{\sim}$ of the quotient space $X/{\sim}$ (equipped with the finest topology that makes the projection map from X to $X/{\sim}$ continuous) is defined by saying that $\operatorname{dom}(\gamma/{\sim}) = \operatorname{dom} \gamma$ and that $(\gamma/{\sim})(\varphi)$ is the equivalence class of $\gamma(\varphi)$.

3.3.4 Function spaces

For topological spaces X and Y, we write $\mathrm{C}[X \to Y]$ for the set of all continuous (total single-valued) functions from X to Y. In this section, we discuss how to construct a nice representation of $\mathrm{C}[X \to Y]$ from given admissible representations γ and δ of X and Y, respectively.

If we do not consider resource bounds, there is a very general answer to this question: we define the representation $[\gamma \to \delta]$ of $\mathrm{C}[X \to Y]$ as follows. The promise $\operatorname{dom}[\gamma \to \delta]$ consists of $\langle M, p \rangle \in \mathbf{Reg}$ such that $\delta \circ \underline{M}^p \circ \gamma^{-1}$ is a (total single-valued) function. For such $\langle M, p \rangle$, we set $[\gamma \to \delta](\langle M, p \rangle) = \delta \circ \underline{M}^p \circ \gamma^{-1}$. Note that every element of $\mathrm{C}[X \to Y]$ gets a name by Theorems 2.7 and 3.12.

The following property is one reason to believe that this representation $[\gamma \to \delta]$ is very natural for the space.

Theorem 3.13 ([Wei00, Lemma 3.3.14]). *Let γ and δ be representations of X and Y, respectively. Let Apply denote the total $(\mathrm{C}[X \to Y] \times X, Y)$-problem defined by Apply$(f, x) = f(x)$. For any representation ζ of $\mathrm{C}[X \to Y]$,*

1. *Apply is $([\zeta, \gamma], \delta)$-continuous if and only if $\zeta \le_{\mathrm{cont}} [\gamma \to \delta]$.*
2. *Apply is in $([\zeta, \gamma], \delta)$-**FRec** if and only if $\zeta \le_{\mathbf{FRec}} [\gamma \to \delta]$.*

Thus, $[\gamma \to \delta]$ is the weakest (in the sense of $\leq_{\mathrm{cont}}$ and $\leq_{\mathsf{FRec}}$) representation that makes *Apply* continuous/computable.

There does not seem to be a way to construct a representation that satisfies the analogous property at the level of **FP**. But this is sometimes possible for specific spaces. We will present such a representation for the space of continuous real functions later.

3.4 Representations for some specific spaces

3.4.1 Real numbers

For real numbers, we use the representation $\rho_{\mathbb{R}}$ defined at the beginning of this chapter. We list some problems solvable in FP with respect to $\rho_{\mathbb{R}}$.

Example 3.14. Addition $+$ (as an $(\mathbb{R} \times \mathbb{R}, \mathbb{R})$-problem) is in $([\rho_{\mathbb{R}}, \rho_{\mathbb{R}}], \rho_{\mathbb{R}})$-**FP** (recall the construction of the representation $[\rho_{\mathbb{R}}, \rho_{\mathbb{R}}]$ of pairs of real numbers, Section 3.3.2). To see this, first note that adding two rational numbers can be done in polynomial time. More precisely, there is a polynomial-time algorithm A that, given two strings u, $v \in \mathbb{D}$, outputs a string $A(u, v) \in \mathbb{D}$ satisfying $[\![A(u, v)]\!] = [\![u]\!] + [\![v]\!]$ within time polynomial in $|u|$ and $|v|$. We present a polynomial-time machine M that $([\rho_{\mathbb{R}}, \rho_{\mathbb{R}}], \rho_{\mathbb{R}})$-solves the real addition $+$. Suppose that M is given $\rho_{\mathbb{R}}$-names φ and ψ of real numbers s and t, and we are also given the string input 0^m. Since $u := \varphi(0^{m+1})$ satisfies a $|[\![u]\!] - s| < 2^{-m-1}$ and $v := \psi(0^{m+1})$ satisfies a $|[\![v]\!] - t| < 2^{-m-1}$, the sum $[\![A(u, v)]\!] = [\![u]\!] + [\![v]\!]$ is a 2^{-m}-approximation of $s + t$, so M can output $A(u, v)$. This can be done in time polynomial in $|u| = |\varphi|(m + 1)$ and $|v| = |\psi|(m + 1)$, and thus in $|\varphi|$, $|\psi|$ and m.

Example 3.15. Similarly, multiplication $\times$ (as a $(\mathbb{R} \times \mathbb{R}, \mathbb{R})$-problem) is in $([\rho_{\mathbb{R}}, \rho_{\mathbb{R}}], \rho_{\mathbb{R}})$-**FP**. For suppose that we are given $\rho_{\mathbb{R}}$-names φ and ψ of real numbers s and t. First, let $k = \max\{|\varphi(\varepsilon)|, |\psi(\varepsilon)|\}$ (where ε denotes the empty string). Since $[\![\varphi(\varepsilon)]\!]$ and $[\![\psi(\varepsilon)]\!]$ are near s and t, and it takes more than k digits to encode a number with absolute value $\geq 2^k$, we have $|s|$, $|t| < 2^k$. Hence, $s \times t$ is approximated within precision 2^{-m} by $[\![u_{m+k+1}]\!] \cdot [\![v_{m+k+1}]\!]$.

Example 3.16. For each ε, the problem F defined at (2.4) (the finite-precision test, Figure 2.1, left) is in $(\rho_{\mathbb{R}}, \rho_{\mathbb{R}})$-**FP**. For let n be such that $2^{-n} < \varepsilon$. Given a $\rho_{\mathbb{R}}$-name φ of $x \in \mathbb{R}$, the machine can read $\varphi(0^n)$ $|[\![\varphi(0^n)]\!] - x| < 2^{-n}$. If $[\![\varphi(0^n)]\!]$ is negative, then $x < 2^{-n} < \varepsilon$, so the machine can safely tell that $1 \in F(x)$. If $[\![\varphi(0^n)]\!]$ is nonnegative, then $x \geq -2^{-n} > -\varepsilon$, so the machine can safely tell that $0 \in F(x)$. Note that this machine yields different answers for different representations of x. Thus, multi-valuedness is essential for the solubility of this problem.

Example 3.17. The exponential function exp as a $([0, 1], \mathbb{R})$-problem is in $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-**FP**, because it can be written as the sum of the series

$$\exp t = 1 + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^2}{3!} + \cdots \tag{3.10}$$

that converges fast on $[0, 1]$ (that is, given a desired precision, the machine can tell how many initial terms it needs to add). However, the $(\mathbb{R}, \mathbb{R})$-problem exp (defined on the whole real line) is not in $(\rho_\mathbb{R}, \rho_\mathbb{R})$-**FP**, because it grows too fast.

Example 3.18. The sine function $\sin\colon \mathbb{R} \to \mathbb{R}$ is in $(\rho_\mathbb{R}, \rho_\mathbb{R})$-**FP**. To see this, note that just like exp in the previous example, sin is polynomial-time computable if restricted to, say, $[-4, 4]$. It is also possible, given $t \in \mathbb{R}$ as oracle and a desired precision, to find efficiently a number in $[-4, 4]$ which is close enough to t modulo 2π. We can compute $\sin t$ by combining these algorithms.

Example 3.19. A function can be in $(\rho_\mathbb{R}, \rho_\mathbb{R})$-**FP** without even an explicit description known. The *distance trisector curves* between the points $(0, 1)$ and $(0, -1)$ in the plane are the pair of sets C_1, $C_2 \subseteq \mathbb{R}^2$ such that C_1 is the set of points equidistant from $(0, 1)$ and C_2, and C_2 is the set of points equidistant from $(0, -1)$ and C_1. Asano, Matoušek and Tokuyama [AMT07] showed that such a pair (C_1, C_2) exists and is unique (see [KMT10] for a simpler and more general proof), and that C_1 (as well as C_2, which is the mirror reflection) is the graph of an $(\mathbb{R}, \mathbb{R})$-function f in $(\rho_\mathbb{R}, \rho_\mathbb{R})$-**FP**. This is despite the fact that very little is understood about the curves and it is suspected that they are different from any curve previously known to mathematicians [AMT07].

The representation $\rho_\mathbb{R}$ is based on the idea of *absolute error*: the nth approximation $u = \varphi(0^n)$ written in a name φ of x is such that the error $\varepsilon := |[\![u]\!] - x|$ is less than 2^{-n}. Numerical analysts also talk about the *relative error* $\delta := \varepsilon/x$ (consider $x > 0$ and $\varepsilon \ll x$ for now). But saying that $[\![u]\!]$ is within relative error δ from x is roughly equivalent (when δ is small) to saying that $\log[\![u]\!]$ is within absolute error δ from $\log x$. Thus, a situation where we want to find a real number x with a small relative error can be modeled as a situation where we want to find a $\rho'_\mathbb{R}$-name of x, where $\rho'_\mathbb{R} = \exp \circ \rho_\mathbb{R}$ is another representation of $\mathbb{R}$.

Thus we do not insist on choosing a single "right" representation of $\mathbb{R}$, but leave the possibility of scaling to an appropriate representation whenever appropriate for specific applications. We see this as a natural thing to do. When a scientist uses a semi-logarithmic plot to visualize the relation between quantities x and y, he believes that the essence of the relation is best understood by considering it as a relation between x and $\log y$. The representation to be used for a real number depends on what quantity the number stands for and how we measure or interpret them. Our choice of $\rho_\mathbb{R}$ as the standard representation of $\mathbb{R}$ in this thesis is not for any intrinsic reason, but is motivated by the fact that it defines the same polynomial-time computability as the ones previously studied by other authors, as we will discuss next.

Equivalence with other formulations

We write $\rho_\mathbb{R}|^{[0,1]}$ for the restriction of $\rho_\mathbb{R}$ to (names of) real numbers in the interval $[0, 1]$ (see Section 3.3.3).

Here we point out that (the total functions in) our classes defined through $\rho_\mathbb{R}$ coincide with some classes that have been studied by other authors. Because of this, in later parts of the thesis where we talk about problems whose inputs and outputs are (one or more) real numbers, we say such problems are *polynomial-time computable* when they belong to (γ, δ)-**FP**, where the representations γ and δ are tacitly chosen from $\rho_{\mathbb{R}}$, $\rho_{\mathbb{R}}|^{[0,1]}$, $[\rho_{\mathbb{R}}, \rho_{\mathbb{R}}]$, etc., for spaces $\mathbb{R}$, $[0,1]$, $\mathbb{R}^2$, etc.

The infinite string approach Because a $\rho_{\mathbb{R}}$-name φ encodes information only in the values $\varphi(0^n)$, we can define an analogous encoding $\tilde{\rho}_{\mathbb{R}}$ (used in the following paragraph only) of real numbers by infinite strings: a $\tilde{\rho}_{\mathbb{R}}$-name of a real number x is an infinite list $u_0\#u_1\#\dots$ of (codes of) rational numbers $[\![u_i]\!]$ with $|[\![u_i]\!] - x| < 2^{-i}$.

However, a little thought shows that the simple combination of Weihrauch's infinite string machine (at the end of Section 2.2.3) and this representation $\tilde{\rho}_{\mathbb{R}}$ is useless [Wei00, Examples 7.2.1, 7.2.3] (recall that such a machine running in polynomial time means that the time needed before writing out the first n symbols of the output infinite string is bounded polynomially in n). On the one hand, the machine M could "cheat" by writing redundant $\tilde{\rho}_{\mathbb{R}}$-names: By writing +10000/100000 instead of +1/10 it gets more time to compute the next approximation. On the other hand, the machine may suffer by receiving redundant names as input, such as the one in which the first approximation is too long to even read in time.

Thus to develop a meaningful complexity theory, we need to disallow redundancy carefully. This leads to the use of *signed digit representation* $\tilde{\rho}_{\mathrm{sd}}$ of $\mathbb{R}$ [Wei00, Definition 7.2.4], defined as follows: $\operatorname{dom} \tilde{\rho}_{\mathrm{sd}}$ consists of sequences $\varphi \in \Sigma^{\mathbb{N}}$ of form $a_k \dots a_1 a_0 \bullet a_{-1} a_{-2} \dots$ for some k, where each a_i is either 0, 1 or -1, such that either $k = 0$ or $(a_k, a_{k-1}) \in \{(1,0), (1,1), (-1,0), (-1,-1)\}$; if this is the case, set

$$\tilde{\rho}_{\mathrm{sd}}(\varphi) = \sum_{i=-\infty}^{k} a_i \cdot 2^i. \tag{3.11}$$

Thus we read the digit sequence as a binary expansion of a real number (with decimal point $\bullet$) with digits 0, 1 and -1; we forbid certain patterns in the first two digits of the integer part in order to exclude redundancy (see [Wei00, Example 2.1.4.7] for the reason why the usual binary expansion without the "-1" digit does not work). Let $\tilde{\rho}_{\mathrm{sd}}|^{[0,1]}$ be the restriction of $\tilde{\rho}_{\mathrm{sd}}$ to (names of) numbers in $[0,1]$ (as in Section 3.3.3).

Müller [Mül86, Mül87] and Weihrauch [Wei00, Chapter 7] use this (or essentially the same) encoding $\tilde{\rho}_{\mathrm{sd}}$ of real numbers by infinite strings to study the complexity of various real functions in detail.

Theorem 3.20. *A $([0,1], \mathbb{R})$-function f is in $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-**FP** if and only if it is $(\tilde{\rho}_{\mathrm{sd}}|^{[0,1]}, \tilde{\rho}_{\mathrm{sd}})$-computed by Weihrauch's infinite string machine in polynomial time.*

This equivalence is proved by an easy simulation. A key fact used for the "only if" direction is that there is a polynomial r such that for each $\tilde{\rho}_{\mathrm{sd}}$-name $p = a_0 a_1 \dots$ of a real number $t \in [0,1]$ there is a $\rho_{\mathbb{R}}$-name $\varphi_p \in \mathbf{Reg}$ of t whose length $|\varphi_p|$ is bounded by r and which encodes essentially the same information as p. Hence, if the assumed oracle Turing machine that $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-computes f runs in time P (a second-order polynomial), then on oracle φ_p and 0^n it runs in $P(r)(n)$, which is simply a polynomial in n only, so this computation can be simulated by an infinite string machine reading p.

This is not true if t ranges over $\mathbb{R}$ rather than $[0,1]$ (there is no single polynomial r that bounds the length of the most efficient $\rho_{\mathbb{R}}$-name of $t \in \mathbb{R}$). Because of this, the class of $(\mathbb{R}, \mathbb{R})$-functions $(\tilde{\rho}_{\mathrm{sd}}, \tilde{\rho}_{\mathrm{sd}})$-computed by infinite string machines in polynomial time is not a nice one. Even very simple functions are not polynomial-time computable, as the following example shows:

Example 3.21. Addition (as an $(\mathbb{R} \times \mathbb{R}, \mathbb{R})$-problem) is not polynomial-time $([\tilde{\rho}_{\mathrm{sd}}, \tilde{\rho}_{\mathrm{sd}}], \tilde{\rho}_{\mathrm{sd}})$-computed by any infinite string machine M running in polynomial time (cf. Example 3.14). For suppose that M runs within a polynomial time bound p. In particular, M has to write the first symbol of the output in $s := p(1)$ steps or fewer. Note that this first symbol must be 1 if the sum is greater than 1, and -1 if the sum is less than -1. In particular, it must be 1 if the two summands are 2^{s+100} and -2^{s+50}, and -1 if they are 2^{s+50} and -2^{s+100}. However, M cannot tell between these two cases, because it can read at most s symbols of the input in time.

In fact, it is easy to see by a similar adversarial argument that no encoding (instead of $\tilde{\rho}_{\mathrm{sd}}$) of real numbers by infinite strings can induce the same polynomial-time computability as our $(\rho_{\mathbb{R}}, \rho_{\mathbb{R}})$-**FP**. Thus, our choice to use regular functions and their lengths is an essential extension to the infinite string approach. We could avoid the problem described in Example 3.21 (in this specific case of $\mathbb{R}$) by simply saying, as Hoover [Hoo90] and Takeuti [Tak01] did, that the time to output the ith bit below the decimal point may depend polynomially in both i and the logarithm of the absolute value of the input real number; see Theorem 3.23 and Lemma 3.27.

Another advantage of our approach using functions with lengths (even for the discussion of $([0,1], \mathbb{R})$-functions) is that we did not have to worry so much about making our representations $\rho_{\mathbb{R}}$ concise as we did in the definition (3.11) of $\tilde{\rho}_{\mathrm{sd}}$. With our model where the running time depends on the size of the oracle, a redundant input name would automatically give the machine more time.

Ko, Friedman and Hoover's approach Ko and Friedman seem to be the first to define the polynomial-time computability of $([0,1], \mathbb{R})$-functions. Hoover extends their definition to $(\mathbb{R}, \mathbb{R})$-functions. Their formulations look similar to our classes $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$- and $(\rho_{\mathbb{R}}, \rho_{\mathbb{R}})$-**FP** in that they are based on Turing machines taking function oracles, but they do not explicitly mention **FP**. In particular, they do not have the notion of size of oracles. Nevertheless, their polynomial-time computability coincides with our notions:

Theorem 3.22. *A $([0,1], \mathbb{R})$-function f is in $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-***FP** *if and only if it is polynomial-time computable in Ko and Friedman's sense.*

Instead of presenting a tedious proof of this fact formally, we shall point out informally what features of their formulation make it equivalent to ours.

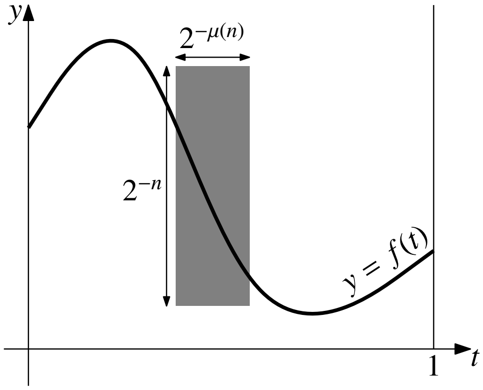


Figure 3.3: Modulus of continuity μ of a $([0,1],\mathbb{R})$-function f.

Consider the situation where we want to determine the value $f(t)$ to within error bound 2^{-n}. Just as with our model, Ko and Friedman give a name φ of t to the machine as an oracle (which is a (Σ^*,Σ^*)-function). However, the running time of the Turing machine is not allowed to depend on the oracle φ; the computation is required to finish in time polynomial in n only. Notice that, our representation $\rho_{\mathbb{R}}$ would be useless in this model because of too much redundancy, just as $\tilde{\rho}_{\mathbb{R}}$ is useless with the infinite string machine (see the above discussion on the infinite string approach). To get around this, Ko and Friedman's representation of real numbers is carefully designed to forbid redundancy. What they essentially do is to require the nth approximation $\varphi(0^n)$ to be of special form so that for each $t \in \mathbb{R}$ there are only finitely many possible values for $\varphi(0^n)$.

Hoover [Hoo90] extends Ko and Friedman's polynomial-time computability to $(\mathbb{R},\mathbb{R})$-functions f by saying that the 2^{-m}-approximation of the value $f(t)$ should be delivered within polynomial time not only in m but also in $\log|t|$. This notion is equivalent to ours:

Theorem 3.23. *An $(\mathbb{R},\mathbb{R})$-function f is in $(\rho_{\mathbb{R}},\rho_{\mathbb{R}})$-**FP** if and only if it is polynomial-time computable in Hoover's sense.*

This equivalence was already essentially pointed out by Lambov [Lam06].

Modulus of continuity

We say that a non-decreasing $(\mathbb{N},\mathbb{N})$-function μ is a *modulus of continuity* of a $([0,1],\mathbb{R})$-function f if for all $n \in \mathbb{N}$ and $t, t' \in [0,1]$ such that $|t-t'| \le 2^{-\mu(n)}$, we have $|f(t)-f(t')| \le 2^{-n}$ (Figure 3.3). Note that any continuous $([0,1],\mathbb{R})$-function f is uniformly continuous and hence has a modulus of continuity.

Using the aforementioned fact that numbers in $[0,1]$ have short $\rho_{\mathbb{R}}|^{[0,1]}$-names (see the discussion after Theorem 3.20), it is easy to see that functions in $(\rho_{\mathbb{R}}|^{[0,1]},\rho_{\mathbb{R}})$-**FP** (or $(\rho_{\mathbb{R}}|^{[0,1]},\rho_{\mathbb{R}})$-**FPSPACE**, for that matter) have a polynomial modulus of continuity. In fact, functions in $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-**FP** are characterized by this plus the property that their values at dyadic numbers can be easily approximated:

Theorem 3.24 ([Ko91, Corollary 2.21]). *A* $([0,1], \mathbb{R})$*-function* f *is in* $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-**FP** *(resp.* $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-**FPSPACE***) if and only if there are a polynomial* μ *and a function* $\varphi \in \mathsf{FP}$ *(resp.* $\mathsf{FPSPACE}$*) such that*

(a) μ *is a modulus of continuity of* f*, and*

(b) *for every* $n \in \mathbb{N}$ *and* $u \in \mathbb{D}$*, we have* $v := \varphi(0^n, u) \in \mathbb{D}$ *and* $|f(\llbracket u \rrbracket) - \llbracket v \rrbracket| < 2^{-n}$.

3.4.2 Effective metric spaces

We extend slightly the discussion of computation over real numbers in Section 3.4.1. Just as an effective topological space (a topological space with an encoded base) had the standard representation (Definition 3.8), a separable metric space with an encoded countable base induces a representation:

Definition 3.25 (Effective metric spaces). A triple (X, d, α) is called an *effective metric space*[4] if $d\colon X \times X \to \mathbb{R}$ is a metric on X and α is a surjective partial (Σ^*, Q)-function, where Q is a dense subset of X (with respect to the topology induced by d). The *Cauchy representation* $\hat{\delta} = \delta_{(X,d,\alpha)}$ of X is then defined by saying that a regular function $\varphi \in$ **Reg** is a $\hat{\delta}$-name of a point $x \in X$ when[5] for all $n \in \mathbb{N}$ we have $\varphi(\mathtt{0}^n) \in \operatorname{dom} \alpha$ and $d(\alpha(\varphi(\mathtt{0}^n)), x) < 2^{-n}$.

The Cauchy representation is indeed a representation (i.e., it is surjective) because of the assumption that Q is dense in X. An example of an effective metric space is $(\mathbb{R}, |\cdot - \cdot|, \llbracket\cdot\rrbracket)$, the real numbers with the usual distance, where $\llbracket\cdot\rrbracket$ is the encoding of dyadic numbers from (3.1). The induced Cauchy representation of $\mathbb{R}$ is $\rho_{\mathbb{R}}$.

Equivalence with Takeuti's approach Takeuti defines a class of "polynomial-time" functions between effective metric spaces without explicitly using type-two functions [Tak01, Definition 3.20]. Here we show that his class coincides with our $(\hat{\gamma}, \hat{\delta})$-**FP** (where $\hat{\gamma}$ and $\hat{\delta}$ are the Cauchy representations of the effective metric spaces) under certain conditions. Then we show in Lemma 3.28 that these conditions are unnecessary for contractive functions, which were the only application discussed in Takeuti's paper.

[4] This terminology is by Weihrauch [Wei00, Definition 8.1.2]. Also called just a *separable metric space* by Brattka [Bra03, Section 5] or a *numbered metric space* by Takeuti [Tak01, Definition 3.1]. Some authors use $\mathbb{N}$ instead of Σ^* to define essentially the same thing (via the binary encoding of $\mathbb{N}$ by Σ^*). We will henceforth translate other statements of theirs to our setting without explicit notice. Some authors assume that α is total function.

[5] Instead of this inequality, some authors require that $d(\alpha(\varphi(\mathtt{0}^m)), \alpha(\varphi(\mathtt{0}^n))) \le 2^{-m}$ for all nonnegative integers m and $n \ge m$ and that $\alpha(\varphi(\mathtt{0}^n))$ converges to x as n increases. This difference is inessential, because any name φ of x in one sense can be translated to a name φ' of x in the other by shifting indices by at most one: $\varphi'(\mathtt{0}^n) = \varphi(\mathtt{0}^{n+1})$.

Definition 3.26 (Takeuti's representation)**.** Let (X, d_X, α_X) and (Y, d_Y, α_Y) be effective metric spaces. An (X, Y)-function T is *represented in Takeuti's sense*[6] by a pair of (Σ^*, Σ^*)-functions g, h if there is[7] a string λ in the domain of α and

(a) $d_Y\bigl(T(\alpha_X(u)), \alpha_Y(g(\mathsf{0}^m, u))\bigr) \le 2^{-m}$ for any $u \in \Sigma^*$ and $m \in \mathbb{N}$, and

(b) $d_Y(T(x), T(y)) \le 2^{-m}$ for any $x, y \in X$ and $m, n \in \mathbb{N}$ with $d_X(x, \alpha_X(\lambda)), d_X(y, \alpha_X(\lambda)) \le 2^n$ and $d_X(x, y) \le 2^{-|h(\mathsf{0}^m, \mathsf{0}^n)|}$.

For example, consider the case X and Y are the space $\mathbb{R}$ of real numbers (see the paragraph after Definition 3.25). The condition (a) says that T can be evaluated approximately on dyadic numbers via g. The condition (b) says that T has something similar to a modulus of continuity h, but h takes as an argument a bound on the distance from the origin. This is necessary in order to admit functions, such as $x \mapsto x^2$, that are not uniformly continuous but still quite simple.

Under some conditions, the classes defined through Takeuti's representation coincides with our classes defined using the Cauchy representations:

Lemma 3.27. *Let (X, d_X, α_X) and (Y, d_Y, α_Y) be effective metric spaces and let $\hat{\gamma}$ and $\hat{\delta}$ be their Cauchy representations, respectively.*

1. *Suppose that $d_X(\alpha_X(u), \alpha_X(v)) < 2^{p(|u|,|v|)}$ for some polynomial p. Then every (X, Y)-function represented in Takeuti's sense by a pair of functions in* FP *(resp.* $\mathsf{FPSPACE}$*) belongs to $(\hat{\gamma}, \hat{\delta})$-***FP** *(resp.* **FPSPACE***).*
2. *Suppose that there is a polynomial q such that for any i, $k \in \mathbb{N}$ and $x \in X$ with $d(x, \alpha(\lambda)) < 2^k$, there is a string v with $|v| < q(i, k)$ and $d(x, \alpha(v)) < 2^{-i}$. Then every (X, Y)-function in $(\hat{\gamma}, \hat{\delta})$-***FP** *(resp.* **FPSPACE***), is represented in Takeuti's sense by a pair of functions in* FP *(resp.* $\mathsf{FPSPACE}$*).*

We omit the proofs by easy but tedious simulations.

Since $\mathbb{R}$ as an effective metric space satisfies both conditions in Lemma 3.27, an $(\mathbb{R}, \mathbb{R})$-function is in $(\rho_{\mathbb{R}}, \rho_{\mathbb{R}})$-**FP** (resp. $(\rho_{\mathbb{R}}, \rho_{\mathbb{R}})$-**FPSPACE**) if and only if it is represented in Takeuti's sense by a pair of functions in FP (resp. $\mathsf{FPSPACE}$).

Takeuti's paper was about contractive functions. An (X, X)-function T on a metric space X with distance function d is said to be *contractive* if

$$d\bigl(T(x), T(y)\bigr) \le c \cdot d(x, y), \qquad x, y \in X \tag{3.12}$$

for some constant $c < 1$. Because this implies a trivial modulus of continuity 1, the condition (b) is superfluous when T is contractive. Furthermore, for such functions we do not need the assumptions in Lemma 3.27:

[6] This terminology has nothing to do with the word "representation" that we have been using since Section 3.1 to mean the way to encode sets by **Reg**.

[7] Takeuti's definition uses numbers instead of strings, and the role of this string λ is played by the specific number 0. It is easy to see that quantifying over λ does not change the definition.

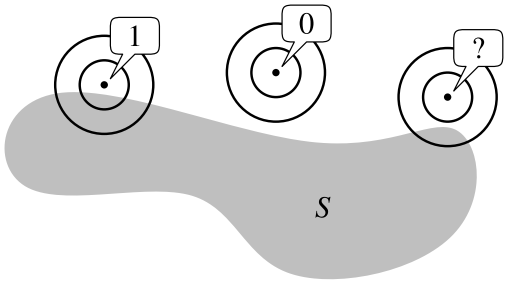


Figure 3.4: A ψ-name of a set S says 1 if the given dyadic point is close to S, and 0 if it is far from S, but it can say either way in between.

Lemma 3.28. *Let (X, d, α) be an effective metric space and $\hat{\delta}$ be its Cauchy representation. A contractive (X, X)-function T is in $(\hat{\delta}, \hat{\delta})$-**FP** (resp. **FPSPACE**) if and only if there is a function $g \in$ FP (resp. FPSPACE) satisfying* (a) *of Definition 3.26.*

In Section 4.1.3, we will recall and simplify Takeuti's discussion about the complexity of the fixed point of a contractive function.

3.4.3 Real sets

Let $\mathcal{A}_{[0,1]^2}$ be the set of closed subsets of $[0,1]^2$. We are going to introduce a representation of $\mathcal{A}_{[0,1]^2}$ that we consider natural, but before that, we recall a notion of polynomial-time computability of sets $S \in \mathcal{A}_{[0,1]^2}$ that has been used by many computable analysts: we say that S is *(nondeterministic) polynomial-time computable* if there is a (nondeterministic) polynomial-time Turing machine that computes a function $\varphi \in \mathbf{Pred}$ such that for any $n \in \mathbb{N}$ and u, $v \in \mathbb{D}$,

- $\varphi(u, v, 0^n) = 1$ if $\mathit{dist}((\llbracket u \rrbracket, \llbracket v \rrbracket), S) < 2^{-n}$, and
- $\varphi(u, v, 0^n) = 0$ if $\mathit{dist}((\llbracket u \rrbracket, \llbracket v \rrbracket), S) > 2 \cdot 2^{-n}$,

where $\mathit{dist}(p, S) := \inf_{q \in S} \|p - q\|$ denotes the Euclidean distance of point $p \in \mathbb{R}^2$ from S (Figure 3.4). The motivation for this definition is that a set is polynomial-time computable if it can be efficiently displayed on a computer screen where the user can zoom in (we require that a part of the set should be visible on the screen however "thin" it is). See e.g. Braverman [Bra05] for a discussion on this definition and its variants.

Now we define the representation $\psi_{\circledcirc}$ of $\mathcal{A}_{[0,1]^2}$ by saying that $\varphi \in \mathbf{Pred}$ be a $\psi_{\circledcirc}$-name of $S \in \mathcal{A}_{[0,1]^2}$ when it satisfies the above conditions.

Note that this representation is more succinct than the one that we would be able to define using the infinite string model (see [Wei03, Example 6.9]).

Since $\operatorname{dom} \psi_{\circledcirc} \subseteq \mathbf{Pred}$, it makes sense to talk about $\psi_{\circledcirc}$-NP, $(\psi_{\circledcirc}, \psi_{\circledcirc})$-**NP**, etc. (Section 3.1), and we will do so in one of the applications in Chapter 4.

The following is immediate from the definition:

Lemma 3.29. *A set in $\mathcal{A}_{[0,1]^2}$ is (nondeterministic) polynomial-time computable if and only if it is in $\psi_{\circledcirc}$-P (resp. $\psi_{\circledcirc}$-NP).*

3.4.4 Real functions

For $A \subseteq \mathbb{R}^d$, we abbreviate $\mathrm{C}[A \to \mathbb{R}]$ (the set of continuous functions from A to $\mathbb{R}$) to $\mathrm{C}[A]$. Define the representation $\delta_{\square}$ of $\mathrm{C}[0,1]$ as follows (see Lemma 3.30 and Theorem 3.33 below for the reasons why we believe $\delta_{\square}$ to be a natural representation): for $\mu\colon \mathbb{N} \to \mathbb{N}$, $\varphi \in \mathbf{Reg}$ and $f \in \mathrm{C}[0,1]$, we set $\delta_{\square}(\langle \overline{\mu}, \varphi \rangle) = f$ if and only if μ and φ satisfy the two conditions in Theorem 3.24, i.e., μ is a modulus of continuity of f and for every $n \in \mathbb{N}$ and $u \in \mathbb{D}$, we have $v := \varphi(0^n, u) \in \mathbb{D}$ and $|f([\![u]\!]) - [\![v]\!]| < 2^{-n}$ (the string v may have to have leading 0s padded in order to make φ regular—but this need for padding is inconsequential, see the penultimate paragraph of Section 2.2.3; in the sequel, we sometimes omit this padding in the description of algorithms).

To see that $\delta_{\square}(\varphi)$ is well-defined, suppose that the above condition holds for two real functions f and f'. Let $t \in [0,1]$ be arbitrary. Then for each $n \in \mathbb{N}$, there is $u \in \mathbb{D}$ with $|t - [\![u]\!]| \le 2^{-\mu(n)}$ and thus

$$\begin{aligned} |f(t) - f'(t)| &\le |f(t) - f([\![u]\!])| + |f([\![u]\!]) - [\![\varphi(0^n, u)]\!]| \\ &\quad + |f'([\![u]\!]) - [\![\varphi(0^n, u)]\!]| + |f'(t) - f'([\![u]\!])| \\ &\le 2^{-n} + 2^{-n} + 2^{-n} + 2^{-n} = 2^{-n+2}. \end{aligned} \tag{3.13}$$

Since $n \in \mathbb{N}$ was arbitrary, $f(t) = f'(t)$. Since $t \in [0,1]$ was arbitrary, $f = f'$.

Recall that the only reason that a real number can require long $\rho_{\mathbb{R}}$-names was having a large absolute value. In contrast, functions in $\mathrm{C}[0,1]$ may have long $\delta_{\square}$-names for two possible reasons: having big values, and having a big modulus of continuity.

The following lemma says that the complexity of $\delta_{\square}$-names of $f \in \mathrm{C}[0,1]$ matches the complexity of f that was defined in Section 3.4.1 by representing real numbers by $\rho_{\mathbb{R}}$:

Lemma 3.30. *A function in $\mathrm{C}[0,1]$ is in $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-$\mathbf{FP}$ (resp. $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}})$-$\mathbf{FPSPACE}$) if and only if it is in $\delta_{\square}$-FP (resp. $\delta_{\square}$-$\mathsf{FPSPACE}$).*

Proof. Immediate from Theorem 3.24. □

Lemma 3.31. *A function in $\mathrm{C}[0,1]$ is PSPACE-complete in the sense of [Kaw10, Section 1.2] if it is $\delta_{\square}$-$\mathbf{FPSPACE}$-$\le^1_{\mathrm{mF}}$-complete and has a polynomial modulus of continuity.*

Proof. Suppose that $f \in \mathrm{C}[0,1]$ is $\delta_{\square}$-$\mathsf{FPSPACE}$-$\le^1_{\mathrm{mF}}$-complete and has a polynomial modulus of continuity μ. Then for any $A \in \mathsf{PSPACE}$ there are polynomial-time functions t and r that satisfy the left picture of Figure 2.4 for any $B \in \delta_{\square}^{-1}[f]$—and thus for any B of the form $\langle \overline{\mu}, \varphi \rangle$ (that is, those which have this particular polynomial μ as the first component). A query to B can ask either a value of μ or a value of φ, but μ is just a polynomial, so we may assume that t only asks a query of form "$\varphi(0^n, v)$?". Thus given a quantified boolean formula u, its truth value can be computed by r from a 2^{-n}-approximation of $f([\![v]\!])$. This implies that f is PSPACE-complete in the sense of [Kaw10]. □

Because a $\delta_\Box$-name contains the information on the modulus of continuity, we can extract from it an upper bound of the function:

Example 3.32 (Upper bound). Define a $(\mathrm{C}[0,1],\mathbb{R})$-problem UB by $\textsc{ub}[f] = [\max f, +\infty)$. Then UB is in $(\delta_\Box, \rho_\mathbb{R})$-**P**, i.e., it is easy, given (a $\delta_\Box$-name of) a continuous $([0,1],\mathbb{R})$-function, to obtain an upper bound of it. For suppose that $\langle \overline{\mu}, \varphi \rangle$ is a $\delta_\Box$-name of $f \in \mathrm{C}[0,1]$. Then $u_0 := \varphi(\varepsilon, +0/1)$ satisfies

$$|[\![u_0]\!] - f(0)| < 1. \tag{3.14}$$

Also, $m := \mu(0)$ is a number such that $|f(s) - f(t)| < 1$ for all $|s - t| < 2^{-m}$, and hence

$$|f(s) - f(0)| < 2^m \tag{3.15}$$

for all $s \in [0,1]$. These imply that $[\![u_0]\!] + 1 + 2^m \in \textsc{ub}[f]$. This number can be obtained in time polynomial in μ and $|\varphi|$.

In contrast, finding the exact maximum of a real function is harder unless $\mathsf{P} = \mathsf{NP}$; see Section 4.3.1.

A universal property The representation $\delta_\Box$ of $\mathrm{C}[0,1]$ may look somewhat arbitrary at first sight. Here we show a property of $\delta_\Box$ that seems to make it a "natural" representation (once we have chosen $\rho_\mathbb{R}$ as the representation for real numbers).

Define the $(\mathrm{C}[0,1] \times [0,1], \mathbb{R})$-function *Apply* by $\mathit{Apply}(f, x) = f(x)$, as in Theorem 3.13. The following theorem says that $\delta_\Box$ is the weakest representation of $\mathrm{C}[0,1]$ that makes *Apply* efficiently computable (see Section 3.2.1 for the definitions of $\le_{\mathsf{FP}}$ and $\equiv_{\mathsf{FP}}$). Note that the analogous claims for $\le_{\mathrm{cont}}$ and $\le_{\mathsf{FRec}}$ have been proved in Theorem 3.13 in a more general form, since $\delta_\Box \equiv_{\mathsf{FRec}} [\rho_\mathbb{R}|^{[0,1]} \to \rho_\mathbb{R}]$.

Theorem 3.33. *For any representation δ of $\mathrm{C}[0,1]$, we have $\mathit{Apply} \in ([\delta, \rho_\mathbb{R}|^{[0,1]}], \rho_\mathbb{R})$-**FP** if and only if $\delta \le_{\mathsf{FP}} \delta_\Box$.*

Proof. For the "if" part, it suffices to prove that *Apply* is in $([\delta_\Box, \rho_\mathbb{R}|^{[0,1]}], \rho_\mathbb{R})$-**FP**. Let a $\delta_\Box$-name $\langle \overline{\mu}, \varphi \rangle$ of $f \in \mathrm{C}[0,1]$ and a $\rho_\mathbb{R}|^{[0,1]}$-name θ of $t \in [0,1]$ 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 $([\delta, \rho_\mathbb{R}|^{[0,1]}], \rho_\mathbb{R})$-**FP**. That is, there is a second-order polynomial P such that given a δ-name ψ of a function $f \in \mathrm{C}[0,1]$, a $\rho_\mathbb{R}|^{[0,1]}$-name θ of a number $t \in [0,1]$, and a unary string 0^n, it is possible to compute 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 $f \in \mathrm{C}[0,1]$ 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 $[0,1]$ has a $\rho_\mathbb{R}|^{[0,1]}$-name of size λ; for example, $\lambda(n) = 2n + 4$ will do, because every number in $[0, 1]$ has a 2^{-n}-approximation of form (3.1) 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 $\rho_{\mathbb{R}}|^{[0,1]}$-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 is such that $\varphi(0^n, u)$, where $u \in \mathbb{D}$, is a 2^{-n}-approximation of $f(\llbracket u \rrbracket)$. This can be computed from ψ, 0^n, u as follows. Let $\theta \in$ **Reg** be the constant function taking value u. Note that θ is a $\rho_{\mathbb{R}}$-name of $\llbracket u \rrbracket$. 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. □

The above definitions and lemmas generalize easily to $\mathrm{C}[A]$, where $A \subseteq \mathbb{R}^d$ is a fixed compact rectangle other than $[0, 1]$; we keep writing $\delta_{\Box}$ by abuse of notation.

4 Operators in Analysis and Geometry

This chapter lists examples of complexity results about various real functions and operators obtained by applying the theory developed in the previous chapters. In Section 4.1, we discuss problems that are in **FP** (with respect to suitable representations, mainly $\rho_{\mathbb{R}}$). In Section 4.2, we deal with problems related to sets of real numbers, using the representation $\psi_{\circledcirc}$ of subsets of $[0,1]^2$. The next two chapters are about operators on real functions and thus use the representation $\delta_{\square}$ of continuous real functions. In Section 4.3, we formulate and prove the effective versions of Ko and Friedman's results about the complexity of maximization and integration of real functions. In Section 4.4, we discuss the complexity of the initial value problem of ordinary differential equations. The problems considered here are not meant to be exhaustive in any sense. We believe that the same framework can be applied to many other problems in numerical computation.

4.1 Within polynomial time

In Section 4.1.1 we point out that Schönhage's algorithm to approximate the roots of a polynomial means the polynomial-time computability in our sense. Thus, whether to represent a polynomial by listing its coefficients or in the factorized form does not matter (up to polynomial-time). In Section 4.1.2 we show that, an analytic function on the compact domain $[0,1]$ is polynomial-time computable (under the representation $\delta_{\square}$) if and only if the sequence of its Taylor coefficients is polynomial-time computable. However, this result is ineffective: we do not have a procedure to convert between the $\delta_{\square}$-name and the Taylor sequence. In Section 4.1.3, we show a stronger version of Takeuti's result that the fixed point of a polynomial-time computable contractive function on an effective metric space with a certain property has a polynomial-time computable name.

4.1.1 Polynomials and their roots

In Examples 3.14 and 3.15 we have seen that addition and multiplication are polynomial-time computable (i.e., in $([\rho_{\mathbb{R}}, \rho_{\mathbb{R}}], \rho_{\mathbb{R}})$-**FP**). Hence any polynomial function is polynomial-time computable. This is still so if we consider complex polynomials by introducing the representation $\rho_{\mathbb{C}}$ of $\mathbb{C}$ by $\rho_{\mathbb{C}}(\langle \varphi, \psi \rangle) = \rho_{\mathbb{R}}(\varphi) + \sqrt{-1}\,\rho_{\mathbb{R}}(\psi)$.

Schönhage proved a polynomial-time version of the fundamental theorem of algebra, which in our terminology can be stated as follows.

Let $n \in \mathbb{N}$, and define an equivalence relation $\sim$ on $\mathbb{C}^n$ by saying that $r \sim s$ if $r \in \mathbb{C}^n$ is a permutation of $s \in \mathbb{C}^n$. Thus, the quotient $\mathbb{C}^n/{\sim}$ is the space of *multisets* of n complex numbers. Define a $(\mathbb{C}^n, \mathbb{C}^n/{\sim})$-function $\textsc{roots}_n$ by saying that $\textsc{roots}_n(a_{n-1}, \dots, a_0)$ is a multiset of roots of the polynomial $\sum_{i=0}^{n-1} a_i x^i$.

Theorem 4.1 (Essentially [Sch82])**.** $\textsc{roots}_n \in (\rho_{\mathbb{C}}^n, \rho_{\mathbb{C}}^n/{\sim})$-**FP**.

4.1.2 Power series

The discussion presented in this section is from [Kaw]. Some of the results below (Lemmas 4.2 and 4.3) were essentially pointed out earlier by Müller [Mül87, Theorems 3.4 and 3.6].

In this section, we use subscripts to denote the components of any vector $\mu \in \mathbb{N}^m$ or $x \in \mathbb{R}^m$: thus $\mu = (\mu_0, \dots, \mu_{m-1})$ and $x = (x_0, \dots, x_{m-1})$. For any μ, $\nu \in \mathbb{N}^m$ and $x \in \mathbb{R}^m$, we write

$$|\mu| = \sum_{i=0}^{m-1} \mu_i, \qquad \mu! = \prod_{i=0}^{m-1} \mu_i!, \qquad \binom{\mu}{\nu} = \frac{\mu!}{\nu!\,(\mu-\nu)!}, \qquad x^\mu = \prod_{i=0}^{m-1} x_i^{\mu_i}. \tag{4.1}$$

For $\varepsilon > 0$, write $\mathrm{B}(x, \varepsilon)$ for the open cube $(x_0 - \varepsilon, x_0 + \varepsilon) \times \dots \times (x_{m-1} - \varepsilon, x_{m-1} + \varepsilon)$.

A function $f\colon D \to \mathbb{R}$ on an open set $D \subseteq \mathbb{R}^m$ is said to be *analytic* at $\hat{x} \in D$ if there are an open neighbourhood $V \subseteq D$ of $\hat{x}$ and real numbers a_μ, one for each $\mu \in \mathbb{N}^m$, such that for all $x \in V$, the series

$$\sum_{\mu \in \mathbb{N}^m} a_\mu (x - \hat{x})^\mu \tag{4.2}$$

converges to $f(x)$. It is easy to see [KP02, Proposition 2.1.7] that if (some serial rearrangement of) the sum (4.2) converges, so does the sum of $|a_\mu| \cdot r^\mu$ over $\mu \in \mathbb{N}^m$ for any $r \in [0, |x_0 - \hat{x}_0|) \times \dots \times [0, |x_{m-1} - \hat{x}_{m-1}|)$. The above definition therefore makes sense regardless of the order of summation, and the maximal such open neighbourhood V equals the interior of the set of all x for which $|a_\mu| \cdot |(x - \hat{x})^\mu|$ is bounded. We call this maximal V the *domain of convergence*, and a_μ the *Taylor coefficients*, of f at $\hat{x}$. A real function on set $K \subseteq \mathbb{R}^m$ is said to be *analytic* if it can be extended to a function on some open set $D \supseteq K$ that is analytic at every point in D.

Computing Taylor coefficients of analytic functions

We compare the computability of an analytic function and its Taylor coefficients.

To discuss computation of Taylor coefficients, recall from Section 3.3.2 the construction of the representation of infinite sequences. By using appropriate pairing functions for indices, it is easy to define a representation $\rho_{\mathbb{R}}^{\mathbb{N}^m}$ of $\mathbb{R}^{\mathbb{N}^m}$, the space of m-dimensional sequences of real numbers. Thus a sequence $(x_\mu)_{\mu \in \mathbb{N}^m}$ is in the class $\rho_{\mathbb{R}}^{\mathbb{N}^m}$-**FP** if there is a polynomial-time machine that, given $n \in \mathbb{N}$ and (each component of) μ in unary notation as inputs, outputs some $d \in \mathbb{D}$ with $|d - x_\mu| < 2^{-n}$. This is the case if and only if the $(\mathbb{N}^\mu, \mathbb{R})$-function that maps μ to x_μ is in $(\kappa_\tau, \mathbb{R})$-**FP**, where κ_τ^m is the representation of $\mathbb{N}^m$ defined in the way explained in Sections 3.3.1 and 3.3.2 from the tally notation τ of $\mathbb{N}$.

We will show in Theorem 4.6 that an analytic $(K, \mathbb{R})$-function on a compact set $K \subseteq \mathbb{R}^m$ is in $\delta_\square$-FP (or equivalently, in $(\rho_\mathbb{R}^m|^K, \rho_\mathbb{R})$-**FP**) if and only if the sequence of its Taylor coefficients (or equivalently the sequence of derivatives) at a rational point is in $\rho_\mathbb{R}^{\mathbb{N}^m}$-FP. A similar fact is pointed out by Ko and Friedman [KF88] for $m = 1$. Generalization to $m > 1$ is not hard, but we present it here for the sake of completeness and simpler proof. For readability we say that a real function or a sequence of real numbers is *polynomial-time computable* when it is in $\delta_\square$- or $\rho_\mathbb{R}^{\mathbb{N}^m}$-FP.

We begin by showing that series (4.2) is easy to compute on a compact subset of the domain of convergence.

Lemma 4.2. *Suppose that a function $f\colon D \to \mathbb{R}$ on an open set $D \subseteq \mathbb{R}^m$ is analytic at $\hat{x} \in D \cap \mathbb{Q}^m$, with domain of convergence $V \subseteq D$ and Taylor coefficients $(a_\mu)_{\mu \in \mathbb{N}^m}$. If the sequence $(a_\mu)_{\mu \in \mathbb{N}^m}$ is polynomial-time computable, so is the restriction of f to any compact subset K of V.*

Proof. Fix any $r \in \mathbb{R}^m$ with $\hat{x} + r \in K$. We will show that there is a compact set K_r containing both $\hat{x}$ and $\hat{x}+r$ in its interior such that the restriction of f to K_r is polynomial-time computable. This suffices because K is compact.

Since V is open, $\hat{x}+r/(1-\varepsilon)^2 \in V$ for some $\varepsilon \in (0, 1)$. As mentioned above, $|a_\mu| \cdot |(r/(1-\varepsilon)^2)^\mu|$ is bounded, say by M. Let K_r be the hyperrectangle with vertices $(\hat{x}_0 \pm |r_0|/(1-\varepsilon), \ldots, \hat{x}_{m-1} \pm |r_{m-1}|/(1-\varepsilon))$. Then $|a_\mu| \cdot |(x-\hat{x})^\mu| \le M(1-\varepsilon)^{|\mu|}$ for each $x \in K_r$. Hence, (4.2) converges fast: the sum (4.2) differs from the partial sum over $\mu_0, \ldots, \mu_{m-1} < N$ by at most

$$\sum_{\mu \in \mathbb{N}^m \setminus \{0,\ldots,N-1\}^m} M(1-\varepsilon)^{|\mu|} = M \cdot \frac{1-(1-(1-\varepsilon)^N)^m}{\varepsilon^m} \le \frac{Mm}{\varepsilon^m}(1-\varepsilon)^N, \tag{4.3}$$

which is bounded by 2^{-n} by making N only polynomially large in n. Thus, (4.2) is approximated by (approximately) adding up polynomially many terms. This can be done in time if $(a_\mu)_{\mu \in \mathbb{N}^m}$ is polynomial-time computable and $\hat{x}$ is rational. □

Now we consider the other direction: computing the Taylor coefficients a_μ from the values of f near $\hat{x}$. If f is analytic at $\hat{x}$, then each derivative $\mathrm{D}^\mu f(\hat{x})$ of f at $\hat{x}$ of order μ exists and equals $a_\mu \mu!$ [KP02, Remark 2.2.4]. Since $\mu!$ has length polynomial in $|\mu|$ and can be multiplied easily, $(a_\mu)_{\mu \in \mathbb{N}^m}$ is polynomial-time computable if and only if $(\mathrm{D}^\mu f(\hat{x}))_{\mu \in \mathbb{N}^m}$ is. Therefore, we will consider how to compute the sequence of derivatives.

The following lemma allows us to approximate the kth derivative of a unary function f at $\hat{x}$ by using the values of f at $k+1$ points near $\hat{x}$.

Lemma 4.3. *Let n, k, $A \in \mathbb{N}$ and $B > kA2^n$. If a real function f on an open interval (u, v) is $k+1$ times differentiable and $|D^{k+1} f(x)| \le A$ for all $x \in (u, v)$, then for all $\hat{x}$ with*

$u < \hat{x} < v - k/B$, the value

$$B \sum_{i=0}^{k} (-1)^i \binom{k}{i} f\left(\hat{x} + \frac{i}{B}\right) \tag{4.4}$$

differs from $\mathrm{D}^k f(\hat{x})$ by at most 2^{-n}.

Proof. Fix n, k, A, B, (u, v), f and $\hat{x}$ as assumed. Consider the polynomial

$$P(X) = B \sum_{i=0}^{k} \frac{(-1)^i}{i!\,(k-i)!} f\left(\hat{x} + \frac{i}{B}\right) \prod_{j \neq i} \left(X - \hat{x} - \frac{j}{B}\right). \tag{4.5}$$

This P is called the *Lagrange interpolating polynomial.* It agrees with f at $k+1$ points $\hat{x}$, $\hat{x} + 1/B$, …, $\hat{x} + k/B$. The mean value theorem yields inductively on $j = 0$, …, k that $\mathrm{D}^j P$ agrees with $\mathrm{D}^j f$ at (at least) $k + 1 - j$ distinct points between $\hat{x}$ and $\hat{x} + k/B$. In particular, $\mathrm{D}^k P(\xi) = \mathrm{D}^k f(\xi)$ for some $\xi \in (\hat{x}, \hat{x} + k/B)$. Hence,

$$\begin{aligned} |\mathrm{D}^k f(\hat{x}) - \mathrm{D}^k P(\xi)| &= |\mathrm{D}^k f(\hat{x}) - \mathrm{D}^k f(\xi)| \\ &\leq \int_{\hat{x}}^{\xi} \mathrm{D}^{k+1} f(X)\,\mathrm{d}X \leq \int_{\hat{x}}^{\xi} A\,\mathrm{d}X \leq \frac{kA}{B} < 2^{-n}. \end{aligned} \tag{4.6}$$

Calculating the leading coefficient in (4.5), we see that $\mathrm{D}^k P(\xi)$ equals (4.4). □

Observe that perturbing each $f(\hat{x} + i/B)$ by ε affects (4.4) by at most $\varepsilon \cdot 2^{\mu} \cdot B$. We use this to prove polynomial-time computability of the Taylor coefficients.

Lemma 4.4. *Assume that a function $f\colon V \to \mathbb{R}$ on an open set $V \subseteq \mathbb{R}^m$ is infinitely differentiable and that there is a polynomial $\alpha\colon \mathbb{N}^m \to \mathbb{N}$ such that $|D^{\mu} f(x)| \leq 2^{\alpha(\mu)}$ for all $\mu \in \mathbb{N}^m$ and $x \in V$. Let $K \subseteq V$ be a compact set containing a point $\hat{x} \in \mathbb{Q}^m$ in its interior. If the restriction of f to K is polynomial-time computable, so is the sequence $(D^{\mu} f(\hat{x}))_{\mu \in \mathbb{N}^m}$.*

Proof. Given $\mu \in \mathbb{N}^m$ in unary notation, we can find in polynomial time integers A and B that are big enough to satisfy $B > |\mu| A 2^n$, $\mathrm{B}(\hat{x}, \mu/B) \subseteq K$ and $A \geq 2^{\alpha(\mu_0, \ldots, \mu_{i-1}, \mu_i + 1, 0, \ldots, 0)}$ for all $i = 0, \ldots, m-1$, but yet $B = 2^{n + \beta(\mu) - |\mu| - 1}$ for some polynomial β. For each i, Lemma 4.3 implies that $\mathrm{D}^{(\mu_0, \ldots, \mu_{i-1}, \mu_i, 0, \ldots, 0)} f(x)$ can be approximated to precision $1/2^{n-1}$ in time polynomial in μ and n if we are given approximations of $\mathrm{D}^{(\mu_0, \ldots, \mu_{i-1}, 0, 0, \ldots, 0)} f$ at certain $\mu_i + 1$ points near x to precision $1/(2^n \cdot 2^{\mu_i} \cdot B) \geq 1/2^{2n + \beta(\mu) - 1}$. Repeating this inductively for $i = 0, \ldots, m-1$, we can approximate $\mathrm{D}^{\mu} f(\hat{x})$ to precision 2^{-n} in polynomial time using approximations of f at certain $(\mu_0 + 1) \cdots (\mu_{m-1} + 1)$ points near $\hat{x}$ to precision $1/2^{2^m (n + \beta(\mu)) - \beta(\mu) - 1}$. □

If f is analytic at $\hat{x}$, then it can be shown [KP02, Proposition 2.2.10] that there are an open neighbourhood V of $\hat{x}$ and constants C and R such that

$$\frac{|\mathrm{D}^{\mu} f(x)|}{\mu!} \leq \frac{C}{R^{|\mu|}}, \qquad x \in V,\ \mu \in \mathbb{N}^m. \tag{4.7}$$

Thus f (restricted to some small enough V) meets the requirement of Lemma 4.4.

We prove the converse by extending the local result we saw in Lemma 4.2.

Lemma 4.5. *Let $K \subseteq \mathbb{R}^m$ be a compact connected set containing $\hat{x} \in \mathbb{Q}^m$, and let f be an analytic $(K, \mathbb{R})$-function. If the sequence $(D^{\mu} f(\hat{x}))_{\mu \in \mathbb{N}^m}$ is polynomial-time computable, so is f.*

Proof. Let $\overline{f}$ be an analytic extension of f to an open set $D \supseteq K$. For each $x \in D$, let $\varepsilon_x > 0$ be so small that $\overline{\mathrm{B}(x, 3\varepsilon_x)} \subseteq V_x$, where V_x is the domain of convergence of $\overline{f}$ at x. It is easy to see that $\overline{\mathrm{B}(x, \varepsilon_x)} \subseteq V_r$ for every $r \in \mathrm{B}(x, \varepsilon_x)$ by the remark following (4.2).

Let $y \in K$. Since K is connected and compact, there are $x_0, \ldots, x_p \in D$ such that $\hat{x} = x_0$, $y \in \mathrm{B}(x_p, \varepsilon_{x_p})$ and $\mathrm{B}(x_{i-1}, \varepsilon_{x_{i-1}})$ intersects $\mathrm{B}(x_i, \varepsilon_{x_i})$ for each $i = 1, \ldots, p$. Let $r_i \in \mathrm{B}(x_{i-1}, \varepsilon_{x_{i-1}}) \cap \mathrm{B}(x_i, \varepsilon_{x_i}) \cap \mathbb{Q}^m$. By the above paragraph, $r_1 \in V_{\hat{x}}$, $r_2 \in V_{r_1}$, $\ldots$, $r_p \in V_{r_{p-1}}$, $y \in V_{r_p}$. The conclusion follows inductively by using Lemmas 4.2 and 4.4 at each step. □

Theorem 4.6. *Let $K \subseteq \mathbb{R}^m$ be a compact connected set containing $\hat{x} \in \mathbb{Q}^m$. A an analytic $(K, \mathbb{R})$-function f is polynomial-time computable if and only if the sequence of Taylor coefficients of f at $\hat{x}$ is.*

Proof. By Lemma-ta 4.4 and 4.5. □

For a $(K, \mathbb{R})$-function f and its Taylor sequence $(a_{\mu})_{\mu \in \mathbb{N}^m}$, Theorem 4.6 asserts the equivalence of $f \in \delta_{\square}$-FP and $(a_{\mu})_{\mu \in \mathbb{N}} \in \rho_{\mathbb{R}}^{\mathbb{N}^m}$-$\mathsf{FP}$. This inspires us to ask whether the operator that maps f to $(a_{\mu})_{\mu \in \mathbb{N}}$ is in $(\delta_{\square}, \rho_{\mathbb{R}}^{\mathbb{N}^m})$-$\mathbf{FP}$, and whether the operator that maps $(a_{\mu})_{\mu \in \mathbb{N}}$ to f is in $(\rho_{\mathbb{R}}^{\mathbb{N}^m}, \delta_{\square})$-$\mathbf{FP}$. An easy adversarial argument shows that neither is the case.

4.1.3 Contractions

Recall the definition (3.12) of a *contractive* (X, X)-function on a metric space X. When an (X, X)-function T is contractive and X is complete, it is easy to see that T has a unique fixed point fix_T that can be approached by iterating T on arbitrary point $x \in X$ in the sense that

$$d\bigl(\mathit{fix}_T, T^k(x)\bigr) < c^k \cdot d(\mathit{fix}_T, x). \tag{4.8}$$

This fact is called the *contraction mapping principle*, also known as the *Banach fixed point theorem.* Here we will discuss the complexity of this point fix_T when (X, d, α) is an effective metric space (Definition 3.25). Let $\hat{\delta}$ be the corresponding Cauchy representation of X.

As we pointed out in Lemma 3.28, a contractive (X, X)-function T is in $\hat{\delta}$-FP if and only if it only satisfies the condition (a) of Definition 3.26, i.e., there is a $(\Sigma^* \times \Sigma^*, \Sigma^*)$-function g such that

$$d\big(T(\alpha(u)), \alpha(g_{(m)}(u))\big) \leq 2^{-m} \tag{4.9}$$

for all $u \in \Sigma^*$ and $m \in \mathbb{N}$, where we write $g_{(m)}$ for the function that takes each string v to $g(v, 0^m)$. For each nonnegative integer i, write T^i for T iterated i times. We have

$$d\Big(T^{i+1}\big(\alpha(u)\big), T^i\big(\alpha\big(g_{(m)}(u)\big)\big)\Big) < c^i \cdot 2^{-m}, \tag{4.10}$$

since T^i is a contraction with ratio c^i. Let k be a nonnegative integer. Substituting $u = g_{(m)}^{k-i-1}(\lambda)$ into the above inequality and summing over i, we obtain

$$\sum_{i=0}^{k-1} d\Big(T^{i+1}\big(\alpha\big(g_{(m)}^{k-i-1}(\lambda)\big)\big), T^i\big(\alpha\big(g_{(m)}^{k-i}(\lambda)\big)\big)\Big) < \sum_{i=0}^{k-1} c^i \cdot 2^{-m} < \frac{2^{-m}}{1-c}. \tag{4.11}$$

By the triangle inequality, the left-hand side bounds $d(T^k(\alpha(\lambda)), \alpha(g_{(m)}^k(\lambda)))$ from above. This and (4.8) with $x = \alpha(\lambda)$ show, again by the triangle inequality, that

$$d\Big(\mathit{fix}_T, \alpha\big(g_{(m)}^k(\lambda)\big)\Big) < c^k \cdot d\big(\mathit{fix}_T, \alpha(\lambda)\big) + \frac{2^{-m}}{1-c}. \tag{4.12}$$

Thus, $\alpha(g_{(m)}^k(\lambda))$ approximates fix_T with precision 2^{-n}, if we set $m = n + 1 - \lg(1-c)$ and $k = (n + 1 + \lg d(\mathit{fix}_T, \alpha(\lambda)))/(-\lg c)$, which are bounded by polynomials in n. This string $g_{(m)}^k(\lambda)$ can be computed in time polynomial in $m + k$ and thus in n, as long as the lengths of λ, $g_{(m)}(\lambda)$, $g_{(m)}^2(\lambda)$, ..., $g_{(m)}^k(\lambda)$ are polynomially bounded. We have shown:

Theorem 4.7. *Let (X, d, α) be a complete effective metric space, and $\hat{\delta}$ be the Cauchy representation of X. Let T be a contractive (X, X)-function. Suppose that there is a polynomial-time function g satisfying (4.9) (thus $T \in (\hat{\delta}, \hat{\delta})$-$\mathsf{FP}$). Suppose also that $|g_{(m)}^k(\lambda)| < p(m + k)$ for some polynomial p and string λ. Then the fixed point of T is in $\hat{\delta}$-FP.*

Takeuti's version [Tak01, Theorem 1] assumes additionally that X be a *polynomial-time computable metric space*. We have seen that this additional assumption is not needed.

Example 4.8 (Fixed points of real functions). Recall that $X = \mathbb{R}$ is an effective metric space with the Cauchy representation $\rho_{\mathbb{R}}$ (see the paragraph after Definition definition: effective metric space), and let T be a contractive $(\mathbb{R}, \mathbb{R})$-function. Suppose that T is in $(\rho_{\mathbb{R}}, \rho_{\mathbb{R}})$-**FP** and hence there is a function g satisfying 4.9. We may assume that this g satisfies the additional property required in Theorem 4.7, namely that $|g_{(m)}^k(\lambda)|$ is polynomially bounded in m and k. For if it does not, then we may redefine $g_{(m)}(u)$ as the result of rounding $g_{(m+1)}(u)$ to the nearest multiple of 2^{-m-1}. Thus the fixed point of T is in $\rho_{\mathbb{R}}$-FP. Hoover shows a lower bound that complements this [Hoo91]: there is an NC-computable contraction on disjoint intervals for which the problem of finding a fixed point on a given interval is P-complete in a certain precise sense.

As another application of Theorem 4.7, we reprove Takeuti's result [Tak01, Corollary 4.26] that the *Takagi curve* is in $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}}|^{[0,1]})$-**FP**. The Takagi curve, also called the *blancmange function*, was defined in 1903 as an example of continuous but nowhere differentiable function. It is the fixed point fix_T of the following $(\mathrm{C}[[0,1] \to [0,1]], \mathrm{C}[[0,1] \to [0,1]])$-function T:

$$T(f)(x) = \begin{cases} f(2x)/2 + x & \text{if } x \le 1/2, \\ f(2-2x)/2 + 1 - x & \text{if } x \ge 1/2. \end{cases} \tag{4.13}$$

It is immediate from the definition that T is contractive with respect to the distance d on $\mathrm{C}[[0,1] \to [0,1]]$ defined by $d(f,g) = \max_{x \in [0,1]} |f(x) - g(x)|$.

Theorem 4.9. *The Takagi curve is in* $(\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}}|^{[0,1]})$-**FP**.

Thus, the nowhere differentiability of Takagi curve, which says that it is "complicated in an analytic sense", does not at all contradict the fact that it is "simple in a computational sense". In fact, it can be proved [BJL04] that, in the sense of Baire category, almost every polynomial-time computable $([0,1], \mathbb{R})$-function is nowhere differentiable.

By Lemma 3.30, Theorem 4.9 says that the Takagi curve is in $\delta_\square|^{\mathrm{C}[[0,1]\to[0,1]]}$-**FP**. We are going to prove this by applying Theorem 4.7. Although Takeuti [Tak01] formulated a theorem similar to Theorem 4.7, his version required that the space be a *polynomial-time computable metric space*, which is unlikely to be the case, and therefore he used a different (more complicated) argument.

To apply Theorem 4.7 to the space $\mathrm{C}[0,1]$, we regard it as an effective metric space $(\mathrm{C}[0,1], d, \alpha)$, where d is the sup-norm and the partial $(\varSigma^*, \mathrm{C}[0,1])$-function α is defined as follows. For each $n \in \mathbb{N}$, recall that $\mathbb{D}_n$ was defined at (3.1), and let

$$\mathbb{D}_n^- = \{\, u \in \mathbb{D}_n \cap \{0, 1, +, /\}^{2n+4} : [\![u]\!] \in [0,1] \,\} \tag{4.14}$$

Note that every multiple of 2^{-n} in $[0,1]$ is encoded by exactly one element of $\mathbb{D}_n^-$. We set $\alpha(M) = f$ if M is a circuit with inputs $\mathbb{D}_l^-$ and outputs $\mathbb{D}_k^-$ and f is the piecewise linear function with vertices $([\![u]\!], [\![M(u)]\!])_{u \in \mathbb{D}_l^-}$.

Let $\hat{\delta}$ be the Cauchy representation of this effective metric space. It is routine to verify that $\hat{\delta}$-**FP** $= \delta_\square|^{\mathrm{C}[[0,1]\to[0,1]]}$-**FP**.

Proof of Theorem 4.9. Because a circuit corresponds to a machine with fixed input and output lengths, we specify a circuit by an informal program below.

Let $(\mathrm{C}[0,1], d, \alpha)$ and $\hat{\delta}$ as above. It suffices to find the function g in Theorem 4.7. Given 0^m and a circuit M with inputs in $\mathbb{D}_l^-$ and outputs in $\mathbb{D}_k^-$, let $N = g_{(m)}(M)$ be the circuit with inputs in $\mathbb{D}_{l+1}^-$ and outputs in $\mathbb{D}_{\max\{l,k+1\}}^-$ specified by the following program:

```
input: $u \in \mathbb{D}_{l+1}^-$;
if $[\![u]\!] \le 1/2$ then
   $[\![v]\!] := 2[\![u]\!]$ (where $v \in \mathbb{D}_l^-$);
else
```

```
    $\llbracket v \rrbracket := 2 - 2\llbracket u \rrbracket$ (where $v \in \mathbb{D}_l^-$);
  fi
    $w := M(v) \in \mathbb{D}_k^-$;
  if $\llbracket u \rrbracket \leq 1/2$ then
    $\llbracket x \rrbracket := \llbracket w \rrbracket/2 + \llbracket u \rrbracket$ (where $x \in \mathbb{D}^-_{\max\{l,k+1\}}$);
  else
    $\llbracket x \rrbracket := \llbracket w \rrbracket/2 + 1 - \llbracket u \rrbracket$ (where $x \in \mathbb{D}^-_{\max\{l,k+1\}}$);
  fi
  output $x$;
```

It is clear that $\alpha(N) = T(\alpha(M))$. The size of N is larger than that of M only by a polynomial in l and k (we have written it down above). Thus, $|g^i_{(m)}(\lambda)| < p(m+i)$ for some polynomial p, and we can apply Theorem 4.7. □

4.2 Geometric operations

Recall the representation $\psi_{\circledcirc}$ of $\mathcal{A}_{[0,1]^2}$, the set of all closed subsets of $[0,1]^2$ (Section 3.4.3).

4.2.1 Some basic operations

Example 4.10 (Union, intersection and complement). The binary union $\cup$ (as an $(\mathcal{A}^2_{[0,1]^2}, \mathcal{A}_{[0,1]^2})$-problem) is in $([\psi_{\circledcirc}, \psi_{\circledcirc}], \psi_{\circledcirc})$-**P**. For if φ_0 and φ_1 are $\psi_{\circledcirc}$-names of S_0, $S_1 \in \mathcal{A}_{[0,1]^2}$, respectively, then a $\psi_{\circledcirc}$-name φ of $S_0 \cup S_1$ is given by $\varphi(u, v, 0^n) = \max\{\varphi_0(u, v, 0^n), \varphi_1(u, v, 0^n)\}$. However, the binary intersection $\cap$ is not in $([\psi_{\circledcirc}, \psi_{\circledcirc}], \psi_{\circledcirc})$-**P**, because it is not even $(\psi_{\circledcirc}, \psi_{\circledcirc})$-continuous. The $(\mathcal{A}_{[0,1]^2}, \mathcal{A}_{[0,1]^2})$-problem of taking the closure of the complement is also not $(\psi_{\circledcirc}, \psi_{\circledcirc})$-continuous.

Example 4.11 (Polygons). Drawing line segments and polygons with specified vertices is one of the most basic geometric primitives. For $k \in \mathbb{N}$, let $\textsc{polygon}_k$ be the $(([0,1]^2)^k, \mathcal{A}_{[0,1]^2})$-function that maps each n-tuple of points to their convex hull. We claim that $\textsc{polygon}_k \in (((\rho_{\mathbb{R}}|^{[0,1]})^2)^k, \psi_{\circledcirc})$-**FP**.

In the special case where $k = 2$ (thus $\textsc{polygon}_2(r_0, r_1)$ is the line segment with endpoints r_0 and r_1), this means that given a $(\rho_{\mathbb{R}}|^{[0,1]})^2$-names φ_0, φ_1 of points r_0, $r_1 \in [0,1]^2$ as oracles and $(u, v, 0^n)$ as string input, we can put a finger on one of the cases: whether the distance $d := \mathit{dist}((\llbracket u \rrbracket, \llbracket v \rrbracket), \textsc{polygon}_2(r_0, r_1))$ is $\geq 2^{-n}$ or $\leq 2 \cdot 2^{-n}$. To do so, first read φ_0 and φ_1 to obtain dyadic numbers u_0, v_0, u_1, $v_1 \in \mathbb{D}$ such that the points $(\llbracket u_0 \rrbracket, \llbracket v_0 \rrbracket)$ and $(\llbracket u_1 \rrbracket, \llbracket v_1 \rrbracket)$ are at distance $< 2^{-n-1}$ from r_0 and r_1, respectively. Note that $d' := \mathit{dist}((\llbracket u \rrbracket, \llbracket v \rrbracket), \textsc{polygon}_2((\llbracket u_0 \rrbracket, \llbracket v_0 \rrbracket), (\llbracket u_1 \rrbracket, \llbracket v_1 \rrbracket)))$ differs from d by $< 2^{-n-1}$.

Simple calculation shows that

$$d' = \begin{cases} \sqrt{\alpha_0^2 + \beta_0^2} & \text{if } \alpha_0(\alpha_0 - \alpha_1) + \beta_0(\beta_0 - \beta_1) \le 0, \\ \sqrt{\alpha_1^2 + \beta_1^2} & \text{if } \alpha_1(\alpha_0 - \alpha_1) + \beta_1(\beta_0 - \beta_1) \ge 0, \\ |\alpha_0\beta_1 - \alpha_1\beta_0| / \sqrt{(\alpha_0 - \alpha_1)^2 + (\beta_0 - \beta_1)^2} & \text{otherwise,} \end{cases} \tag{4.15}$$

where $\alpha_0 = [\![u_0]\!] - [\![u]\!]$, $\alpha_1 = [\![u_1]\!] - [\![u]\!]$, $\beta_0 = [\![v_0]\!] - [\![v]\!]$, $\beta_1 = [\![v_1]\!] - [\![v]\!]$. We can tell whether $d' \ge \frac{3}{2} \cdot 2^{-n}$ or not, in time polynomial in $|u|$, $|v|$, $|u_0|$, $|v_0|$, $|u_1|$, $|v_1|$. Because $|d' - d| < \frac{1}{2} \cdot 2^{-n}$, this lets us safely say either $d \ge 2^{-n}$ or $d \le 2 \cdot 2^{-n}$.

The general claim with $k \ge 3$ can be proved by the same approach, solving the problem exactly for dyadic numbers close to the input real numbers.

There are many algorithms for finding the convex hull of given points. Some of them perform poorly on "bad" (degenerate) inputs, while some are more robust than others. Now that we have a solid formulation of computing of a set with guaranteed precision, comparing them in our framework would be an interesting future work.

4.2.2 Convex hulls

This time, we consider the problem of drawing the convex hull of a given *set*, rather than points (cf. Example 4.11). For each $S \in \mathcal{A}_{[0,1]^2}$, let $\textsc{ch}(S)$ be the convex hull of S. Thus, $\textsc{ch}$ is an $(\mathcal{A}_{[0,1]^2}, \mathcal{A}_{[0,1]^2})$-function.

Ko and Yu [KY08] and Zhao and Müller [ZM08] essentially proved[1] the following non-constructive theorems about the complexity of $\textsc{ch}$.

Theorem 4.12. *If a closed set $S \in \mathcal{A}_{[0,1]^2}$ is polynomial-time computable, then $\textsc{ch}(S)$ is nondeterministic polynomial-time computable.*

Theorem 4.13. *Unless $\mathsf{P} = \mathsf{NP}$, there exists a closed set $S \in \mathcal{A}_{[0,1]^2}$ which is polynomial-time computable, but whose convex hull $\textsc{ch}(S)$ is not.*

Using our framework, we can state and prove the following effectivized version, from which Theorems 4.12 and 4.13 follow as corollaries.

Theorem 4.14. $\textsc{ch}$ *is $(\psi_{\circledcirc}, \psi_{\circledcirc})$-$\mathbf{NP}$-$\le^2_{\mathrm{m}}$-complete.*

[1] Ko and Yu state both the positive and the negative results (Theorems 4.12 and 4.13) for polynomial-time *strong recognizability* instead of computability [KY08, Corollaries 4.3 and 4.6, respectively], but their proof almost works for computability as well. See Braverman [Bra05] for a comparison of the two notions ([Bra05] says *weak computability* for Ko's strong recognizability). Zhao and Müller use the polynomial-time computability equivalent to ours and prove Theorem 4.12 [ZM08, Theorem 4.3]. For the positive part, in fact they prove a constructive result essentially equivalent to the positive part of our Theorem 4.14 [ZM08, Theorem 4.1]. Although they state the upper bound of "exponential time", their proof contains the argument that is necessary to derive the non-constructive NP upper bound (our Theorem 4.13), as essentially pointed out in their [ZM08, Lemma 4.2].

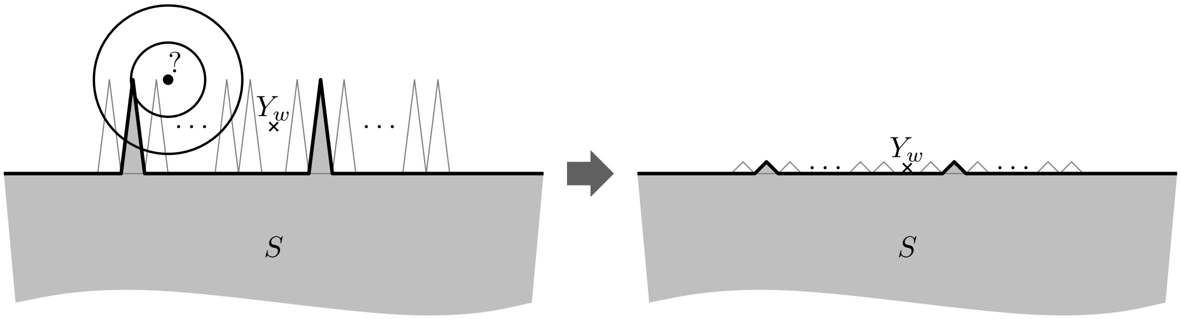


Figure 4.1: Widget for reducing **NP** to CH. We have $Y_w \in \text{CH}(S)$ if and only if there is u such that the slot for $\langle w, u\rangle$ has a bump. In Ko and Yu's construction of S, the bumps can be high (left), and there can be a query that requires the knowledge of $B(\langle w, u\rangle)$ for many u. We make the bumps low (right) in order to make S polynomial-time computable in our sense.

Proof. The main technical ideas are already in Ko and Yu's proof of the corresponding ineffective versions (Theorems 4.12 and 4.13), so we will only sketch the proof.

That CH belongs to $(\psi_{\circledcirc}, \psi_{\circledcirc})$-**NP** is no surprise: A point p belongs to $\text{CH}(S)$ if there are two points p' and p'' in S such that p is on the line segment $p'p''$. All we need is to guess p' and p'' using nondeterminism, with appropriate consideration of precision.

For hardness, we need to modify the proof slightly, because, as we noted in footnote 1, Ko and Yu's original results were about a different notion of computability: our computability of sets is more demanding in the sense that on query $(u, v, 0^n)$, where $u, v \in \mathbb{D}$ and $n \in \mathbb{N}$, if $([\![u]\!], [\![v]\!])$ is within distance 2^{-n} from the set, then we must say 1 (see the definition before Theorems 4.12 and 4.13), whereas for weak computability both 0 or 1 are allowed in this case.

We assume that the reader has Ko and Yu's proof [KY08, Corollary 4.6] at hand. The proof of their Lemma 4.4 begins by taking an arbitrary set $A \in \mathsf{NP}$ and noting that there are $B \in \mathsf{P}$ and a polynomial p such that $w \in A$ if and only if $(w, u) \in B$ for some string u of length exactly $p(|w|)$. Relativizing this, we take $A \in \mathbf{NP}$, and note that there are $B \in \mathbf{P}$ and a second-order polynomial P such that $A(\varphi)(w) = 1$ if and only if $B(\varphi)(w, u) = 1$ for some string u of length $P(|\varphi|)(|w|)$.

We need to provide s and t of Definition 2.12. We define s by describing the set $S = \psi_{\circledcirc}(s(\varphi))$ for a given $\varphi \in \mathbf{Reg}$. The construction of S is similar to that of the original proof, replacing $p(n)$ by $P(|\varphi|)(n)$ and B by $B(\varphi)$. But we need to modify one part to get $s \in \mathbf{P}$.

The original proof constructs a widget depicted in Figure 4.1 left (or Figure 2 of [KY08]) for each string w. In each of the left and right halves, there are exponentially many slots, one for each u, that has a bump if and only if (w, u) is in B (or $B(\varphi)$ for us). The point of this construction is that, while the set S is easy to compute, $\text{CH}(S)$ is hard in the sense that we can tell if w is in A (or $A(\varphi)$) by checking whether the middle point Y_w belongs to $\textsc{ch}(S)$.

However S is not easy in our sense, because in order to answer the query shown in Figure 4.1, we need to know $B(\varphi)(w,u)$ for exponentially many u. To avoid this, we make the bumps low, so they make an angle of at most 45° (Figure 4.1 right). This ensures that any one query to the $\psi_{\circledcirc}$-name of S can be answered by checking $B(\varphi)(w,u)$ for at most one (w,u). Thus a name of S can be **P**-computed from φ.

The function t of Definition 2.12 queries whether the point Y_w is in $\textsc{ch}(S)$ with appropriate precision. Note that t needs access to φ in order to find the location of Y_w and the necessary precision. □

As corollaries of Theorem 4.14, we get Theorem 4.12 by Lemmas 3.3.1 and 3.29, and Theorem 4.13 by Lemmas 3.4.1 and 3.29.

4.3 Operators on real functions

The first complexity results about real functions were Ko and Friedman's work [KF82, Fri84] on the operators taking the maximum and the integral of a real function. As mentioned in the introduction, their original results are in ineffective forms: they studied how complex the outcome of the operator can be when the input to the operator is assumed to be polynomial-time computable. Here we state and prove the effective versions that directly talk about the complexity of the operators.

4.3.1 Maximization

Define the $(\mathrm{C}[[0,1]^2], \mathrm{C}[0,1])$-function $\textsc{max}$ by

$$\textsc{max}(f)(x) = \max_{y\in[0,1]} f(x,y), \qquad x \in [0,1]. \tag{4.16}$$

We will show that $\textsc{max}$ is $(\delta_\square, \delta_\square)$-$\mathbf{FP}^{\mathbf{NP}}$-$\leq^2_{\mathrm{T}}$-complete. Note that the two $\delta_\square$ here denote the representations of $\mathrm{C}[[0,1]^2]$ and $\mathrm{C}[0,1]$, respectively, that were introduced at the beginning of Section 3.4.4 (and generalized at the end of that section). The proof is by an easy relativization of the argument by Friedman [Fri84]. It is also true, by the argument also presented there, that the $(\mathrm{C}[0,1], \mathrm{C}[0,1])$-function $\textsc{max}'$ defined by

$$\textsc{max}'(f)(x) = \max_{y\in[0,x]} f(y), \qquad x \in [0,1] \tag{4.17}$$

is $(\delta_\square, \delta_\square)$-$\mathbf{FP}^{\mathbf{NP}}$-$\leq^2_{\mathrm{T}}$-complete, but we omit the proof.

Lemma 4.15. $\textsc{max}$ *is in* $(\delta_\square, \delta_\square)$-$\mathbf{FP}^{\mathbf{NP}}$.

Before proving Lemma 4.15, it is convenient to introduce another representation of $\mathrm{C}[K]$ (for a compact subset $K \subseteq \mathbb{R}^d$ for some $d \in \mathbb{N}$) that is $\equiv_{\mathsf{FP}}$-equivalent to $\delta_\Box$.

Define the representation δ_{graph} of $\mathrm{C}[0,1]$ as follows: for $\mu\colon \mathbb{N} \to \mathbb{N}$, $\varphi \in \mathbf{Pred}$, $M \in \mathbb{N}$ and $f \in \mathrm{C}[0,1]$, we set $\delta_{\mathrm{graph}}(\langle \overline{\mu}, \varphi, 0^M \rangle) = f$ if and only if μ is a modulus of continuity of f, the values of f are in $[-2^M, 2^M]$, and for every $n \in \mathbb{N}$ and u, $w \in \mathbb{D}$, we have

$$\varphi(0^n, u, w) = \begin{cases} 1 & \text{if } f(\llbracket u \rrbracket) > \llbracket w \rrbracket + 2^{-n}, \\ 0 & \text{if } f(\llbracket u \rrbracket) < \llbracket w \rrbracket - 2^{-n}. \end{cases} \tag{4.18}$$

Thus, $\varphi(0^n, u, w)$ tells us whether the point $(\llbracket u \rrbracket, \llbracket w \rrbracket)$ is below the graph of f, but an error is allowed when this point is just above or below the graph by an amount smaller than 2^{-n}.

Lemma 4.16. $\delta_\Box \equiv_{\mathsf{FP}} \delta_{\mathrm{graph}}$.

Proof. To see $\delta_\Box \leq_{\mathsf{FP}} \delta_{\mathrm{graph}}$, recall from Example 3.32 that an upper bound can be obtained from its $\delta_\Box$-name (translating the φ-part is easy). For $\delta_{\mathrm{graph}} \leq_{\mathsf{FP}} \delta_\Box$, read the bound in the δ_{graph}-name and then use binary search to translate the φ-part. □

Proof of Lemma 4.15. Because $\delta_\Box \equiv_{\mathsf{FP}} \delta_{\mathrm{graph}}$, it suffices by Lemma 3.6 to prove that MAX is in $(\delta_{\mathrm{graph}}, \delta_{\mathrm{graph}})$-$\mathsf{FP}^{\mathsf{NP}}$. We claim that the partial $(\mathbf{Reg}, \mathbf{Reg})$-function B defined as follows is in $\mathsf{FP}^{\mathsf{NP}}$ and $(\delta_{\mathrm{graph}}, \delta_{\mathrm{graph}})$-computes MAX.

Let $\operatorname{dom} B = \operatorname{dom} \delta_{\mathrm{graph}}$ and define $B(\langle \overline{\mu}, \varphi, 0^M \rangle) = \langle \overline{\mu}, \psi, 0^M \rangle$ for each $\langle \overline{\mu}, \varphi \rangle \in \operatorname{dom} B$, where

$$\psi(0^n, u, w) = \begin{cases} 1 & \text{if } \varphi(0^{n+1}, u, v, w) = 1 \text{ for some } v \in \mathbb{D}^-_{\mu(n+1)}, \\ 0 & \text{if } \varphi(0^{n+1}, u, v, w) = 0 \text{ for all } v \in \mathbb{D}^-_{\mu(n+1)} \end{cases} \tag{4.19}$$

for all n and u, $w \in \mathbb{D}$ (recall from (4.14) that $\mathbb{D}^-_m \subseteq \mathbb{D}_m$ is the set of strings encoding a multiples of 2^{-m} in $[0,1]$). This B is in $\mathsf{FP}^{\mathsf{NP}}$ because the above ψ can be computed from $\langle \overline{\mu}, \varphi \rangle$ by nondeterministically guessing a string $v \in \mathbb{D}^-_{\mu(n+1)}$ of length $2\mu(n+1)+4$. To see that MAX is $(\delta_{\mathrm{graph}}, \delta_{\mathrm{graph}})$-computed by B (i.e., that the above ψ is a δ_{graph}-name of $\mathrm{MAX}(f)$), we need to argue that μ is a modulus of continuity of $\mathrm{MAX}(f)$ and that

$$\psi(0^n, u, w) = \begin{cases} 1 & \text{if } \mathrm{MAX}(f)(\llbracket u \rrbracket) > \llbracket w \rrbracket + 2^{-n}, \\ 0 & \text{if } \mathrm{MAX}(f)(\llbracket u \rrbracket) < \llbracket w \rrbracket - 2^{-n}. \end{cases} \tag{4.20}$$

The first claim is obvious ($\mathrm{MAX}(f)$ cannot change its values faster than f). To see the second claim, notice that the upper and lower if-clauses on the right-hand side of (4.20) imply those of (4.19), respectively, by the triangle inequality. □

Now we prove the negative result, the $(\delta_\Box, \delta_\Box)$-$\mathsf{FP}^{\mathsf{NP}}$-$\leq^2_{\mathrm{T}}$-hardness of MAX. We use the following lemma, which is obtained simply by unwinding Definitions 2.13 (reduction $\leq^2_{\mathrm{T}}$) and 3.1.5 ((γ, δ)-C-$\leq^2_{\mathrm{T}}$-hardness).

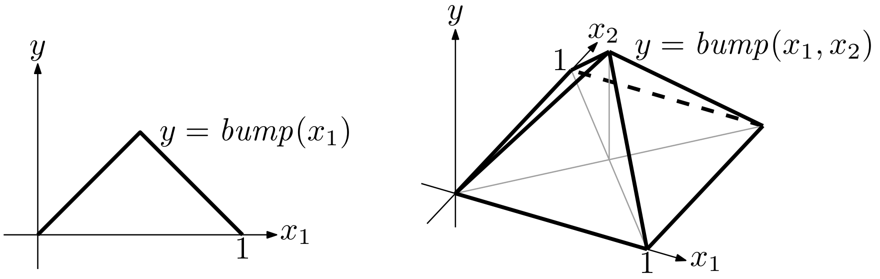


Figure 4.2: The functions *bump*.

Lemma 4.17. *Let γ and δ be representations of X and Y, respectively. An (X, Y)-problem B is C-$\leq^2_{\mathrm{T}}$-hard if and only if for any $A \in \mathsf{C}$, there are $R \in ([\mathrm{id}, \delta], \mathrm{id})$-$\mathsf{FP}$ and $S \in (\mathrm{id}, \gamma)$-$\mathsf{FP}$ such that for any $\varphi \in \operatorname{dom} A$, we have $\varphi \in \operatorname{dom}(B \circ S)$ and for any $\Psi \in (B \circ S)[\varphi]$ we have $(\varphi, \Psi) \in \operatorname{dom} R$ and $R[(\varphi, \Psi)] \subseteq A[\varphi]$.*

Theorem 4.18. $\textsc{max}$ *is $(\delta_\square, \delta_\square)$-$\mathsf{FP}^{\mathsf{NP}}$-$\leq^2_{\mathrm{T}}$-complete.*

Proof. The positive part ($\textsc{max} \in (\delta_\square, \delta_\square)$-$\mathsf{FP}^{\mathsf{NP}}$) was proved in Lemma 4.15. For the hardness, it suffices to give the functions R and S of Lemma 4.17 for $A = \textsc{sat}^2$ and $B = \textsc{max}$ (see Lemma 2.15 for the definition of the NP-$\leq^2_{\mathrm{mF}}$-complete problem $\textsc{sat}^2$). In fact, we will give a reduction where S is single-valued and R ignores the first component of its argument: thus, we will give a problem $R \in (\delta_\square, \mathrm{id})$-$\mathsf{FP}$ and a partial function $S \in (\mathrm{id}, \delta_\square)$-$\mathsf{FP}$ such that $R \circ \textsc{max} \circ S = \textsc{sat}^2$. For each $k \in \mathbb{N}$, define $bump\colon [0,1]^k \to \mathbb{R}$ by $bump(x_1, \dots, x_k) = 1/2 - \max_{i=1}^k |x_i - 1/2|$ (Figure 4.2).

Let $\operatorname{dom} S = \mathbf{Pred}$ and for each $p \in \mathbf{Pred}$ define $S(p) \in \mathrm{C}[0,1]$ by specifying its value $S(p)(x, y)$ at each point $(x, y) \in [0,1]^2$ as follows. For each binary string u, define

$$l_u = 1 - 2^{-|u|} + 2^{-2|u|-1} u, \qquad r_u = l_u + 2^{-2|u|-1}, \tag{4.21}$$

where the u at the end of the first equation is to be read as an integer in $\{0, \dots, 2^{-|u|} - 1\}$ written in binary notation (possibly with leading zeros). This divides the interval $[0, 1)$ into infinitely many subintervals $[l_u, r_u)$, one for each u, without overlap. We set $S(p)(1, y) = 0$ for all $y \in [0,1]$, and define $S(p)(x, y)$ for each $x \in [0, 1)$ as follows. Let u be the unique binary string with $x \in [l_u, r_u)$, i.e.,

$$x = l_u + (r_u - l_u)\xi \tag{4.22}$$

for some $\xi \in [0, 1)$. Furthermore, if this u encodes a boolean formula involving a predicate symbol, then let n be the number of boolean variables in this formula, let $v \in \{0,1\}^n$ and $\eta \in [0,1]$ be such that

$$y = 2^{-n}(v + \eta), \tag{4.23}$$

and define

$$S(p)(x,y) = \begin{cases} 2^{-2|u|-1-n} bump(\xi,\eta) & \text{if } u^p \text{ is satisfied by assignment } v, \\ 0 & \text{otherwise.} \end{cases} \tag{4.24}$$

If u does not encode a formula, let $n = 0$ and $S(p)(x,y) = 0$ for all y.

By the definition of this function $S(p)$ and the operator $\textsc{max}$, we have

$$\textsc{max}\big(S(p)\big)(x) = \begin{cases} 2^{-2|u|-1-n} bump(\xi) & \text{if } u \text{ encodes a boolean formula involving} \\ & \text{a predicate symbol and } u^p \text{ is satisfiable,} \\ 0 & \text{otherwise,} \end{cases} \tag{4.25}$$

and in particular

$$\textsc{max}\big(S(p)\big)(c_u) = 2^{-2|u|-2-n} \cdot \textsc{sat}^2(p)(u), \tag{4.26}$$

where $c_u = (l_u + r_u)/2$ is the midpoint of the interval $[l_u, r_u)$. Hence, we have $R \circ \textsc{max} \circ S = \textsc{sat}^2$ if we define R by saying that $R[f]$ consists of functions ψ satisfying

$$\psi(u) = \begin{cases} 1 & \text{if } f(c_u) > \frac{3}{4} \cdot 2^{-2|u|-2-n}, \\ 0 & \text{if } f(c_u) < \frac{1}{4} \cdot 2^{-2|u|-2-n}. \end{cases} \tag{4.27}$$

It is routine to verify that $R \in (\delta_\square, \mathrm{id})$-**FP** and $S \in (\mathrm{id}, \delta_\square)$-**FP**. For S, note that since *bump* has modulus of continuity 1, and in the definition of $S(p)$ we have scaled it down appropriately, $S(p)$ also has modulus of continuity 1. □

This and Lemmas 3.3 and 3.4 yield the following corollary.

Corollary 4.19. *For all $f \in \delta_\square$-FP, we have $\textsc{max}(f) \in \delta_\square$-$\mathsf{FP}^{\mathsf{NP}}$. There is a function $f \in \delta_\square$-FP such that $\textsc{max}(f)$ is $\delta_\square$-$\mathsf{FP}^{\mathsf{NP}}$-$\leq^1_{\mathrm{T}}$-complete.*

Corollary 4.20 ([Fri84])**.** $\mathsf{P} = \mathsf{NP}$ *if and only if* $\textsc{max}(f) \in \delta_\square$-$\mathsf{FP}$ *for all* $f \in \delta_\square$-FP.

Recall from Lemma 3.30 that a real function with a compact promise $K \subseteq \mathbb{R}^d$ is in $\delta_\square$-FP if and only if it is in $(\rho^d_{\mathbb{R}}|^K, \rho_{\mathbb{R}})$-**FP**, which is the same class as the one defined by Ko and Friedman's approach.

4.3.2 Integration

Define the $(\mathrm{C}[[0,1]^2], \mathrm{C}[0,1])$-function $\textsc{int}$ by

$$\textsc{int}(f)(x) = \int_0^1 f(x,y)\,\mathrm{d}y, \qquad x \in [0,1]. \tag{4.28}$$

The following can be proved in a similar way to Theorem 4.18.

Theorem 4.21. INT *is* $(\delta_\square, \delta_\square)$-$\mathbf{FP}^{\#\mathbf{P}}$-$\leq^2_{\mathrm{T}}$-*complete.*

The ineffective statement analogous to Corollary 4.20, due to Friedman [Fri84], also follows from this theorem. Again, we omit the proof of the other version which states that the $(\mathrm{C}[0,1], \mathrm{C}[0,1])$-function $\textsc{int}'$ defined by

$$\textsc{int}'(f)(x) = \int_0^x f(y)\,\mathrm{d}y, \qquad x \in [0,1] \tag{4.29}$$

is $(\delta_\square, \delta_\square)$-$\mathbf{FP}^{\#\mathbf{P}}$-$\leq^2_{\mathrm{T}}$-complete.

Proof sketch of Theorem 4.21. The positive part ($\textsc{int} \in (\delta_\square, \delta_\square)$-$\mathbf{FP}^{\#\mathbf{P}}$) can be proved similarly to Lemma 4.15 using the representation δ_{graph}. Instead of (4.19), we use

$$\psi(0^n, u, w) = \begin{cases} 1 & \text{if } 2^{-\mu(n+1)}2^{-n-1}N - 2^M[\![v]\!] \geq [\![w]\!], \\ 0 & \text{otherwise,} \end{cases} \tag{4.30}$$

where N is the number of pairs $(v, w) \in \mathbb{D}^-_{\mu(n+1)} \times \mathbb{D}'_{n+1}$ with $([\![v]\!], [\![w]\!]) \in [0,1] \times [-2^M, 2^M]$ and $\varphi(0^{n+1}, u, v, w) = 1$. Here, $\mathbb{D}'_{n+1} \subseteq \mathbb{D}_{n+1}$ is the set of strings of form (3.1) where the numerator has no leading zeros (so that one dyadic number does not get two codes in $\mathbb{D}'_{n+1}$). Note that the number $2^{-\mu(n+1)}2^{-n-1}N - 2^M[\![v]\!]$ in the right-hand side of (4.30) is an approximate value of $\textsc{int}(f)([\![u]\!])$ as estimated by counting the number of points that φ deems to be below the graph of f.

For hardness, we need to reduce $\#\textsc{sat}^2$ instead of $\textsc{sat}^2$ in Theorem 4.18, where $\#\textsc{sat}^2$ is the obvious "counting version" of $\textsc{sat}^2$. To give a reduction (R, S) from $\#\textsc{sat}^2$ to INT, we use exactly the same S as in Theorem 4.18. The other part R is easy, because the integral $\textsc{int}(S(p))(x)$ indicates how many bumps $S(p)$ has on the line $(x, \mathbb{R})$. □

As was the case with maximization, we can derive the corollaries analogous the ones at the end of Section 4.3.1 using Lemmas 3.3 and 3.4. The ineffective statement about INT in the same form as Corollary 4.20 also appears in [Fri84].

4.4 Differential equations

Consider the differential equation

$$h(0) = 0, \qquad h'(t) = g\bigl(t, h(t)\bigr) \tag{4.31}$$

called the *initial value problem* (IVP), where $g \in \mathrm{C}[[0,1] \times \mathbb{R}]$ is given and $h \in \mathrm{C}[0,1]$ is the unknown.

4.4.1 The Lipschitz continuous case

It is well known (see [Kaw10, beginning of Section 3]) that the solution h exists and is unique if g is *Lipschitz continuous* (in the second argument), that is,

$$|g(t, y_0) - g(t, y_1)| \leq L \cdot |y_0 - y_1| \tag{4.32}$$

for some nonnegative number L (called the *Lipschitz constant*) independent of t, y_0, y_1.

As is the case with $\textsc{max}$ (Section 4.3.1), the complexity of the operator of solving this sort of differential equation has been addressed in ineffective forms: the following results state how complex h can be, assuming that g is polynomial-time computable. Since polynomial-time computability is defined only for functions with compact domain, we restrict g to the rectangle $[0, 1] \times [-1, 1]$. If there is a solution $h \in \mathrm{C}[0, 1]$ whose values stay in $[-1, 1]$ (in which case h is unique, as mentioned above), we write $\textsc{livp}(g)$ for this h. Thus $\textsc{livp}$ is a partial function from $\mathrm{CL}[[0, 1] \times [-1, 1]]$ to $\mathrm{C}[0, 1]$, where the former set is the subset of $\mathrm{C}[[0, 1] \times [-1, 1]]$ consisting of functions Lipschitz continuous in the second argument.

Theorem 4.22 (Essentially [Ko83, Section 4][2]). $\textsc{livp}(g) \in \delta_\square\text{-}\mathsf{FPSPACE}$ *for every* $g \in \operatorname{dom} \textsc{livp} \cap \delta_\square\text{-}\mathsf{FPSPACE}$.

Theorem 4.23 ([Kaw10, Theorem 3.2]). *There is* $g \in \operatorname{dom} \textsc{livp} \cap \delta_\square\text{-}\mathsf{FP}$ *such that* $\textsc{livp}(g)$ *is polynomial-space complete (in the sense defined in [Ko92] or [Kaw10]).*

We can derive from Theorem 4.23 a statement of the form similar to Theorem 4.13:

Corollary 4.24 ([Kaw10, Corollary 3.3]). *Unless* $\mathsf{P} = \mathsf{PSPACE}$, *there is a real function* $g \in \operatorname{dom} \textsc{livp}$ *which is polynomial-time computable but* $\textsc{livp}(g)$ *is not.*

We are going to present an effectivized version of the above statements about the complexity of $\textsc{livp}$. To do so, we define a representation $\delta_{\square\mathrm{L}}$ of the space $\mathrm{CL}[[0, 1] \times [-1, 1]]$ of Lipschitz continuous functions by setting $\delta_{\square\mathrm{L}}(\langle \varphi, 0^L \rangle) = g$ if and only if φ is a $\delta_\square$-name of g and $L \in \mathbb{N}$ satisfies (4.32) (regard the string 0^L as the constant function whose value is 0^L).

Theorem 4.25. $\textsc{livp}$ *is* $(\delta_{\square\mathrm{L}}, \delta_\square)$-$\mathsf{FPSPACE}$-$\leq^2_{\mathrm{mF}}$*-complete.*

As corollaries, we have Theorem 4.22 by Lemmas 3.3.2 and 3.30, and Theorem 4.23 by Lemmas 3.4.2 and 3.31.

The following weaker version of Theorem 4.25, stated with the reduction $\leq^2_{\mathrm{T}}$ (Definition 2.12), is slightly easier to prove (see the end of the section):

Corollary 4.26. $\textsc{livp}$ *is* $(\delta_{\square\mathrm{L}}, \delta_\square)$-$\mathsf{FPSPACE}$-$\leq^2_{\mathrm{T}}$*-complete.*

[2] The original theorem is stated with a condition slightly weaker than Lipschitz continuity.

As we noted in Lemma 3.7, this is a more robust result in the sense that it is invariant under replacing representations to $\equiv_{\mathsf{FP}}$-equivalent ones. This seems to be especially nice in view of Theorem 3.33. A drawback is that Corollary 4.26 does not directly yield Theorem 4.23, because Lemma 3.31 requires **FPSPACE**-$\leq^1_{\mathrm{mF}}$-completeness, whereas replacing $\leq^2_{\mathrm{mF}}$ by $\leq^2_{\mathrm{T}}$ in the assumption of 3.4.2 also changes $\leq^1_{\mathrm{mF}}$ to $\leq^1_{\mathrm{T}}$ in the conclusion. We can still obtain Corollary 4.24.

The rest of the section is devoted to the proof of Theorem 4.25. The positive part will be verified by checking that the proof of Theorem 4.22 can be effectivized. For the hardness part, we need to modify slightly the construction in the original proof of Theorem 4.23 in order to get Theorem 4.25 (this modification is not needed if we only want Corollary 4.26). We begin with the positive part.

Proof of the positive part of Theorem 4.25. Given a $\delta_{\square\mathrm{L}}$-name $\langle\overline{\mu}, \varphi, 0^L\rangle$ of g, we need to find a $\delta_{\square}$-name $\langle\overline{\nu}, \psi\rangle$ of $h = \textsc{livp}(g)$. Recall that μ is a modulus of continuity of g, and φ gives approximations of the the values g at dyadic points, that is,

$$|[\![\varphi(0^q, u, v)]\!] - f([\![u]\!], [\![v]\!])| < 2^{-q} \tag{4.33}$$

for each u, $v \in \mathbb{D}$ (such that $([\![u]\!], [\![v]\!]) \in [0,1] \times [-1,1]$).

It is easy to find a modulus of continuity ν of h: let $\nu(n) = n + M$, where $M \in \mathbb{N}$ is any number such that 2^M bounds the maximum absolute value of g. For example, $M = \lceil \log_2(|[\![\varphi(\varepsilon, +0/1, +0/1)]\!]| + 1 + 2^{\mu(0)})\rceil$ will do.

To obtain the ψ part of a $\delta_{\square}$-name $\langle\overline{\nu}, \psi\rangle$ of h, we apply the forward Euler method with step size 2^{-p} to the approximation of g with precision 2^{-q} (we will specify p and q shortly). That is, we define an approximation $\tilde{h}_{p,q} \in \mathrm{C}[0,1]$ of h by letting $\tilde{h}_{p,q}(0) = 0$ and then saying, for each $T = 0, \dots, 2^p - 1$ inductively, that $\tilde{h}_{p,q}(0) = 0$ is linear on the interval $[2^{-p}T, 2^{-p}(T+1)]$, with slope approximately $g(2^{-p}T, h(2^{-p}T))$: formally,

$$\tilde{h}_{p,q}(2^{-p}T + \mathit{\Delta} t) = \tilde{h}_{p,q}(2^{-p}T) + \mathit{\Delta} t[\![\varphi(0^q, u, v)]\!], \qquad 0 \leq \mathit{\Delta} t \leq 2^{-p}, \tag{4.34}$$

for some u, $v \in \mathbb{D}$ with $[\![u]\!] = 2^{-p}T$ and $[\![v]\!] = \tilde{h}_{p,q}(2^{-p}T)$. Obviously, we can compute such a function $\tilde{h}_{p,q}$ in space polynomial in p and q in the sense that there is $\mathit{Euler} \in$ **FPSPACE** such that $[\![\mathit{Euler}(\varphi)(0^p, 0^q, u)]\!] = \tilde{h}_{p,q}([\![u]\!])$ for every $u \in \mathbb{D}$.

Let $\psi(0^n, u) = \mathit{Euler}(\varphi)(0^p, 0^q, u)$, where $p = \max\{\mu(n+8L), n+8L+M\}$ and $q = n+8L$. We claim that $\langle\overline{\nu}, \psi\rangle$ is a $\delta_{\square}$-name of h (note that this shows the desired membership in **FPSPACE**, since p and q are bounded polynomially in $|\varphi|$, μ and n, L). This means that $|\tilde{h}_{p,q}(t) - h(t)| \leq 2^{-n}$ for any $t \in [0,1]$. More strongly, we prove, by induction on $T = 0$, …, $2^p - 1$, that

$$|\tilde{h}_{p,q}(t) - h(t)| \leq 2^{-n}\mathrm{e}^{4L(t-1)} \tag{4.35}$$

for all $t \in [2^{-p}T, 2^{-p}(T+1)]$. We may assume (4.35) for $t = 2^{-p}$ as the induction hypothesis. The approximate value $\tilde{h}_{p,q}(2^{-p}T + \mathit{\Delta} t)$ is defined by (4.34), whereas the true solution h satisfies

$$h(2^{-p}T + \mathit{\Delta} t) = h(2^{-p}T) + \int_{2^{-p}T}^{2^{-p}T+\mathit{\Delta} t} g\bigl(\tau, h(\tau)\bigr)\,\mathrm{d}\tau. \tag{4.36}$$

The error added by this approximation is

$$\left|\Delta t[\![\varphi(0^q,u,v)]\!] - \int_{2^{-p}T}^{2^{-p}T+\Delta t} g\bigl(\tau,h(\tau)\bigr)\,\mathrm{d}\tau\right| \le 4L2^{-n}\mathrm{e}^{4L(2^{-p}T-1)}\Delta t, \tag{4.37}$$

because

$$\begin{aligned}
&\bigl|[\![\varphi(0^q,u,v)]\!] - g\bigl(\tau,h(\tau)\bigr)\bigr| \\
&\quad\le \bigl|[\![\varphi(0^q,u,v)]\!] - g([\![u]\!],[\![v]\!])\bigr| + \bigl|g([\![u]\!],[\![v]\!]) - g(\tau,[\![v]\!])\bigr| + \bigl|g(\tau,[\![v]\!]) - g\bigl(\tau,h(\tau)\bigr)\bigr| \\
&\quad\le 2^{-q} + 2^{-n-8L} + L|[\![v]\!] - h(\tau)| \\
&\quad\le 2^{-n-8L} + 2^{-n-8L} + L\bigl(|[\![v]\!] - h(2^{-p}T)| + |h(2^{-p}T) - h(\tau)|\bigr) \\
&\quad\le 2^{-n-8L} + 2^{-n-8L} + L\bigl(2^{-n}\mathrm{e}^{4L(2^{-p}T-1)} + 2^{-p}2^M\bigr) \\
&\quad\le L\bigl(2^{-n-8L} + 2^{-n-8L} + 2^{-n}\mathrm{e}^{4L(2^{-p}T-1)} + 2^{-n-8L}\bigr) \le 4L2^{-n}\mathrm{e}^{4L(2^{-p}T-1)},
\end{aligned} \tag{4.38}$$

where the second, third and fifth inequalities come from $p \ge \mu(n+8L)$, $q \ge n+8L$, $p \ge M+n+8L$, respectively. Using (4.37) and the induction hypothesis, we compare (4.34) and (4.36) to obtain

$$\begin{aligned}
\bigl|\tilde{h}_{p,q}(2^{-p}T+\Delta t) - h(2^{-p}T+\Delta t)\bigr| &\le 2^{-n}\mathrm{e}^{4L(2^{-p}T-1)} + 4L2^{-n}\mathrm{e}^{4L(2^{-p}T-1)}\Delta t \\
&= 2^{-n}\mathrm{e}^{4L(2^{-p}T-1)}\bigl(1+4L\Delta t\bigr) \le 2^{-n}\mathrm{e}^{4L(2^{-p}T-1)}\mathrm{e}^{4L\Delta t} = 2^{-n}\mathrm{e}^{4L(2^{-p}T-1+\Delta t)},
\end{aligned} \tag{4.39}$$

as desired. □

For the hardness, we use the following lemma. This is an effectivized version of Lemma 4.28, which we will prove in Section 4.4.2 using the **PSPACE**-$\le^1_{\mathrm{mF}}$-complete problem QBF. The effectivized version below is obtained by replacing in this proof the problem QBF with the relativized version QBF^2 from Lemma 2.16.

Lemma 4.27. *Let $L \in$ **PSPACE**. Then there are a second-order polynomial P and a function $G \in$ **FP** such that the following holds for each $\varphi \in \operatorname{dom} L$, $\lambda\colon \mathbb{N}\to\mathbb{N}$ and $u \in \Sigma^*$. Consider the partial function from **Reg** to **Reg** whose domain is $\operatorname{dom}[\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}}|^{[-1,1]}]$ and which maps each θ in it to the function that maps each string v to $G(\varphi,\overline{\lambda},\theta)(u,v)$. Then this function $([\rho_{\mathbb{R}}|^{[0,1]}, \rho_{\mathbb{R}}|^{[-1,1]}], \rho_{\mathbb{R}})$-computes a real function $g^{\varphi}_u\colon [0,1]\times[-1,1]\to\mathbb{R}$ such that*

(a) *$g^{\varphi}_u(0,y) = g^{\varphi}_u(1,y) = 0$ for all $y \in [-1,1]$;*

(b) *$|g^{\varphi}_u(t,y_0) - g^{\varphi}_u(t,y_1)| \le 2^{-\lambda(|u|)}|y_0-y_1|$ for any $t\in[0,1]$ and $y_0,y_1\in[-1,1]$;*

(c) *g^{φ}_u belongs to* dom LIVP, *and $h^{\varphi}_u :=$ LIVP(g^{φ}_u) satisfies $h^{\varphi}_u(1) = 2^{-P(\lambda)(|u|)}\cdot L(\varphi)(u)$.*

Thus G encodes a family of functions g^{φ}_u, indexed by φ, λ and u, such that $h^{\varphi}_u :=$ LIVP(g^{φ}_u) contains the information on $L(\varphi)(u)$.

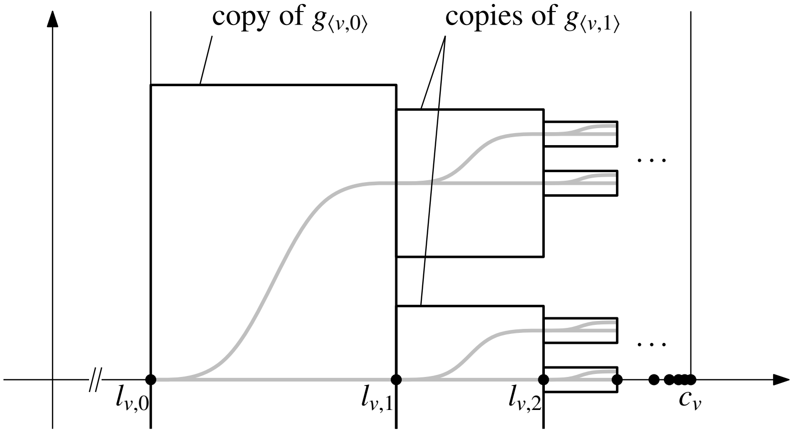

copy of $g_{\langle v,0\rangle}$
copies of $g_{\langle v,1\rangle}$
. . .
. . .
$l_{v,0}$
$l_{v,1}$
$l_{v,2}$
c_v

Figure 4.3: Widget for reducing **FPSPACE** to LIVP.

Proof of the hardness part of Theorem 4.25. Let $F \in$ **FPSPACE**. We need to show that $F \leq^2_{\mathrm{mF}} \delta_\square^{-1} \circ \textsc{livp} \circ \delta_{\square\mathrm{L}}$. We may assume that F is a total function and that there is a second-order polynomial Q such that $F(\varphi)(v)$ has length exactly $Q(|\varphi|)(|v|)$ for all $\varphi \in \mathbf{Reg}$ and $v \in \Sigma^*$. There is $L \in$ **PSPACE** such that $L(\varphi)(v, 0^i)$ equals the ith symbol of $F(\varphi)(v)$ for any $\varphi \in \mathbf{Reg}$, $v \in \Sigma^*$ and $i \in \{0, 1, \ldots, Q(|\varphi|)(|v|) - 1\}$.

Apply Lemma 4.27 to this L to obtain the P and G. Let g^φ_u and h^φ_u be as in the Lemma, corresponding to $\lambda(k) = 3k + 2$.

We define s (of Definition 2.12) by describing the real function $g = \delta_{\square\mathrm{L}}(s(\varphi)) \in \mathrm{CL}[[0,1] \times [-1,1]]$ for a given φ. It has Lipschitz constant 1. It will be straightforward to check that, by the **FP**-computability of G, a $\delta_\square$-name (and hence a $\delta_{\square\mathrm{L}}$-name) of g can be **FP**-computed from φ.

For each binary string v, let $\Lambda_v = 2^{-2|v|-2}$ and

$$c_v = 1 - \frac{1}{2^{|v|}} + \frac{2\overline{v} + 1}{\Lambda_v}, \qquad l^\mp_v = c_v \mp \frac{1}{\Lambda_v}, \tag{4.40}$$

where $\overline{v} \in \{0, \ldots, 2^{|v|} - 1\}$ means v read as an integer in binary notation. This divides $[0, 1)$ into intervals $[l^-_v, l^+_v]$ indexed by $v \in \{0, 1\}^*$. We further divide the left half $[l^-_v, c_v]$ into $Q(|\varphi|)(|v|) + 1$ subintervals $[l_{v,0}, l_{v,1}]$, $[l_{v,1}, l_{v,2}]$, . . . , $[l_{v,Q(|\varphi|)(|v|)-1}, l_{Q(|\varphi|)(|v|)}]$, $[l_{v,Q(|\varphi|)(|v|)}, c_v]$, where

$$l_{v,i} = c_v - \frac{1}{2^i \Lambda_v}, \qquad i = 0, 1, \ldots, Q(|\varphi|)(|v|). \tag{4.41}$$

On each strip $[l_{v,i}, l_{v,i+1}] \times [-1, 1]$, we define g by putting the copies of $g^\varphi_{(v,0^i)}$ as in Figure 4.3. Precisely, on the strip $[l_{v,i}, l_{v,i+1}] \times \mathbb{R}$, we define g by

$$g\left(l_{v,i} + \frac{t}{2^{i+1}\Lambda_v}, \frac{2m + (-1)^m y}{2^{i+1}\Lambda_v \Gamma_{v,i}}\right) = \frac{g^\varphi_{(v,0^i)}(t, y)}{\Gamma_{v,i}} \tag{4.42}$$

for each $t \in [0,1]$ and $m \in \mathbb{N}$, $y \in [-1,1]$, where $\varGamma_{v,i}$ is an appropriate factor (of vertical shrinkage) that can be written exponentially in second-order polynomial in $|\varphi|$ and $|v|$, i. On the last strip $[l_{v,Q(|\varphi|)(|v|)}, c_v] \times [-1,1]$, we define g to be constantly 0. On the right half $[c_v, l_v^+]$, we define g symmetrically: $g(l^+ - t, y) = -g(l^- + t, y)$ for $0 \le t \le 1/\varLambda_v$.

Recall that $g^{\varphi}_{(v,0^i)}$ is a vector field such that $h^{\varphi}_{(v,0^i)} := \text{LIVP}(g^{\varphi}_{(v,0^i)})$ tells us whether $L(\varphi)(v,i) = 1$ in the way described in (c) of Lemma 4.27. Thus by arranging the copies of $g^{\varphi}_{(v,0^i)}$ in this way, we can see the values $L(\varphi)(\langle v,0\rangle)$, …, $L(\varphi)(\langle v, Q(|\varphi|)(|v|) - 1)$ by looking at $h(c_v)$. The reducing functions r and t (of Definition 2.12) do this lookup. That is, $t(\varphi)(v)$ is the encoding of the rational number c_v with appropriate precision, and $r(\varphi)$ is the function that, given the encoding of (an approximation of) $h(c_v)$, extracts the values $L(\varphi)(v,i)$. □

In [Kaw10], the ineffective version of Lemma 4.27 was used to construct a function that proved the ineffective Theorem 4.23 (Theorem 3.2 of [Kaw10]). We needed to use a different construction, because for our Theorem 4.25 (with the reduction $\le^2_{\mathrm{mF}}$), we needed to get the values $L(\varphi)(\langle v,0\rangle)$, …, $L(\varphi)(\langle v, Q(|\varphi|)(|v|) - 1)$ in one query. For Corollary 4.26 (with the reduction $\le^2_{\mathrm{T}}$), we are allowed to make many queries, so the straightforward effectivization of a slightly simpler construction used for Theorems 3.2 of [Kaw10] would have worked (we would not have to stack the copies of $g_{\langle v,i\rangle}$ vertically).

4.4.2 The Lipschitz continuous case: main part of the hardness proof

To keep the notation simple, we state and prove the non-constructive version of Lemma 4.27 here.

Lemma 4.28 ([Kaw10, Lemma 4.1])**.** *Let $L \in \mathsf{PSPACE}$ and let $\lambda\colon \mathbb{N} \to \mathbb{N}$ be a polynomial. Then there exist a polynomial $\rho\colon \mathbb{N} \to \mathbb{N}$ and families of functions $g_u\colon [0,1] \times [-1,1] \to \mathbb{R}$ and $h_u\colon [0,1] \to \mathbb{R}$ indexed by binary strings u such that the family $(g_u)_u$ is polynomial-time computable and for each u we have*

(a) $h_u(t) \in [-1,1]$ *for all* $t \in [0,1]$*;*

(b) $g_u(0,y) = g_u(1,y) = 0$ *for all* $y \in [-1,1]$*;*

(c) $h_u(0) = 0$ *and* $h'_u(t) = g_u(t, h_u(t))$ *for all* $t \in [0,1]$*;*

(d) $|g_u(t,y_0) - g_u(t,y_1)| \le 2^{-\lambda(|u|)}|y_0 - y_1|$ *for any* $t \in [0,1]$ *and* $y_0, y_1 \in [-1,1]$*;*

(e) $h_u(1) = 2^{-\rho(|u|)} L(u)$.

Discrete initial value problem and the Lipschitz condition

A first attempt to prove 4.28 would be as follows. Consider a polynomial-space Turing machine that decides whether a given string u belongs to L. Its configuration at each time

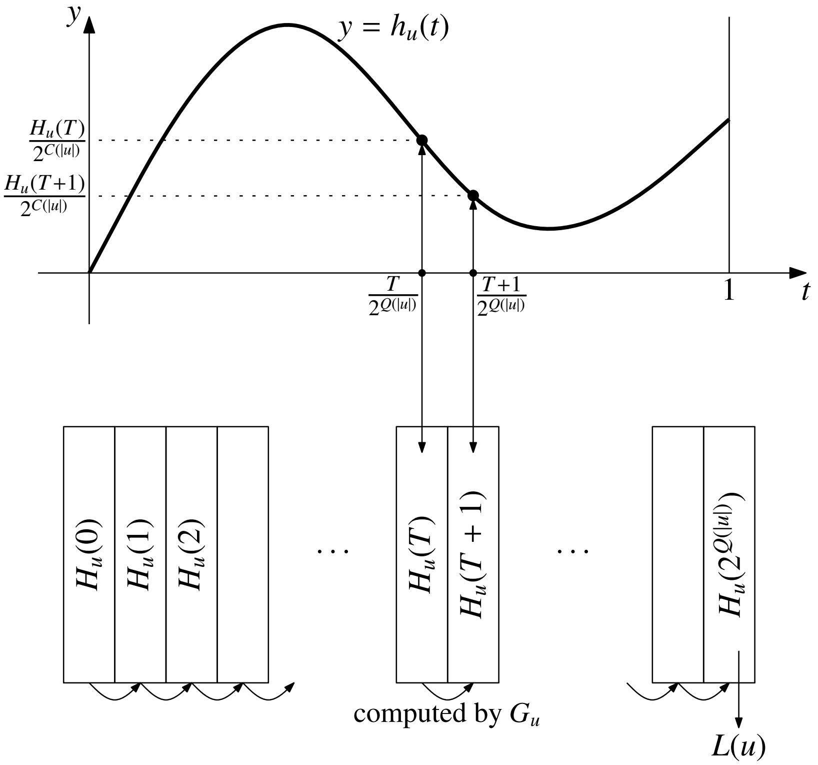


Figure 4.4: An attempt to simulate a polynomial-space Turing machine by an initial value problem is to encode the machine configuration $H_u(T)$ at each time T into the value $h_u(t) = H_u(T)/2^{C(|u|)}$ at time $t = T/2^{Q(|u|)}$.

can be encoded into a nonnegative integer less than $2^{C(|u|)}$, where C is a polynomial. There is a simple rule that maps u (the input), T (time) and d (the current configuration) to a number $G_u(T, d)$ (the next configuration) such that the recurrence

$$H_u(0) = 0, \qquad H_u(T+1) = G_u\bigl(T, H_u(T)\bigr) \tag{4.43}$$

leads to $H_u(2^{Q(|u|)}) = L(u)$ for some polynomial Q. Now this situation looks similar to the one in 4.28: starting at 0, the value of H_u (or h_u) changes over time according to a simpler function G_u (or g_u), to reach a value eventually that indicates the answer $L(u)$. Thus we are tempted to simulate the "discrete initial value problem" (4.43) by embedding each value $H_u(T)$ as real number $H_u(T)/2^{C(|u|)}$ (4.4).

The obstacle to this attempt is that the differential equation of the form (c) of 4.28 cannot express all discrete recurrences of form (4.43): continuous trajectories cannot branch or cross one another; besides, we have the Lipschitz condition (d) that puts restriction on how strong the feedback of h to itself can be. We thus need to restrict the discrete problem (4.43) so that it can be simulated by the continuous version.

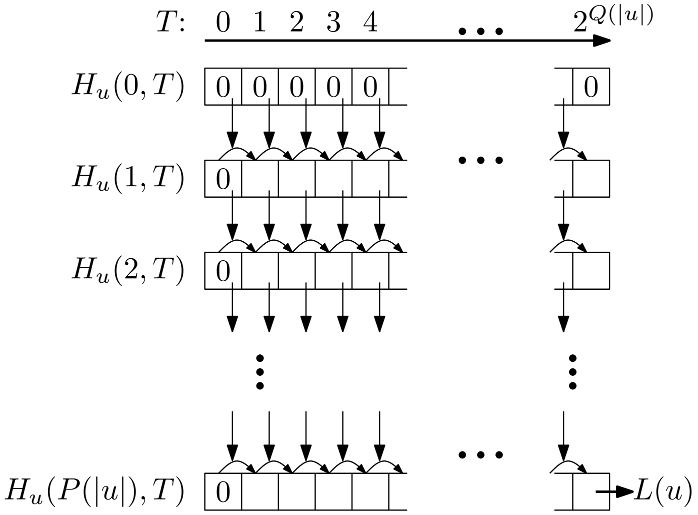


Figure 4.5: The discrete initial value problem (4.44)–(4.47). Each cell $H_u(T)$ in Figure 4.4 is now divided into $H_u(0,T)$, …, $H_u(P(|u|),T)$; the increment from $H_u(i{+}1,T)$ to $H_u(i+1,T+1)$ is computed by G_u using the upper left cell $H_u(i,T)$.

To do so, let us reflect on what the Lipschitz condition (d) means. A rough calculation shows that if two trajectories differ by ε at time t, then they can differ at time $t+2^{-Q(|u|)}$ by at most $\varepsilon \exp(2^{-\lambda(|u|)}2^{-Q(|u|)}) \approx \varepsilon(1+2^{-\lambda(|u|)-Q(|u|)})$. Thus, the gap can only widen (or narrow) by a factor of $\pm 2^{-\lambda(|u|)-Q(|u|)}$ during each time interval of length $2^{-Q(|u|)}$. In other words, the feedback caused by equation (c) is so weak that each digit of h_u can only affect far lower digits of h_u in the next step.

Now we define a discrete problem that reflects this restriction. Let P and Q be polynomials and let

$$G_u\colon\ [P(|u|)] \times [2^{Q(|u|)}] \times [4] \to \{-1,0,1\}, \tag{4.44}$$

$$H_u\colon\ [P(|u|)+1] \times [2^{Q(|u|)}+1] \to [4], \tag{4.45}$$

where we write $[N] = \{0,\dots,N-1\}$ for $N \in \mathbb{N}$. Our restricted discrete initial value problem is as follows:

$$H_u(i,0) = H_u(0,T) = 0, \tag{4.46}$$

$$H_u(i+1,T+1) = H_u(i+1,T) + G_u\bigl(i,T,H_u(i,T)\bigr). \tag{4.47}$$

Thus, $H_u(T)$ of (4.43) is now divided into polynomially many (in $|u|$) components $H_u(0,T)$, …, $H_u(P(|u|),T)$; compare Figures 4.4 (bottom) and 4.5. We have added the restriction that G_u sees only the component $H_u(i,T)$, which in 4.5 means the upper left of the current cell. The following lemma states that, despite this restriction, we still have **PSPACE**-completeness. Note that making G_u completely oblivious to its last argument would be an overkill, because then H_u would just add up the values of G_u, resulting in the complexity merely of #P.

Lemma 4.29. *Let $L \in \mathsf{PSPACE}$. Then there are polynomials P, Q and families $(G_u)_u$, $(H_u)_u$ satisfying (4.44)–(4.47) such that $(G_u)_u$ is polynomial-time computable and $H_u(P(|u|), 2^{Q(|u|)}) = L(u)$ for each string u.*

Before proving this, we will reduce Lemma 4.28 to Lemma 4.29 by simulating the new system (4.44)–(4.47) by the differential equation. Using G_u and H_u of 4.29, we will construct g_u and h_u of 4.28 such that $h_u(T/2^{Q(|u|)}) = \sum_i H_u(i,T)/B^i$ for each T, where B is a big number. Thus, each column in Figure 4.5 will be encoded in one real number so that upper/lower cells in the column correspond to upper/lower bits of the real number. Thanks to the restriction that G_u sees only the upper row, the differential equation $h'_u(t) = g_u(t, h_u(t))$ only needs to cause a weak feedback on h_u where each bit of the value of h_u affects only much lower bits of its next value. This keeps g_u Lipschitz continuous. Now we fill in the details.

Proof of 4.28. Let P, Q, $(G_u)_u$, $(H_u)_u$ be as in 4.29. By "dividing each unit time into $P(|u|)$ steps," we may assume that for each T, there is at most one i such that $G_u(i,T,Y) \neq 0$ for some Y. Write $j_u(T)$ for this unique i (define $j_u(T)$ arbitrarily if there is no such i). We may further assume that

$$H_u\bigl(i, 2^{Q(|u|)}\bigr) = \begin{cases} L(u) & \text{if } i = P(|u|), \\ 0 & \text{if } i < P(|u|). \end{cases} \tag{4.48}$$

Thus, not only does the bottom right corner of Figure 4.5 equal $L(u)$, as stated already in 4.29, but we also claim that the other cells in the rightmost column are all 0. This can be achieved by doubling the time frame and extending G symmetrically so that in the second half it cancels out what it has done. Precisely, we extend G_u by

$$G_u(i, 2 \cdot 2^{Q(|u|)} - 1 - T, Y) = \begin{cases} 0 & \text{if } i = P(|u|) - 1, \\ -G_u(i,T,Y) & \text{if } i < P(|u|) - 1 \end{cases} \tag{4.49}$$

for $(i,T,Y) \in [P(|u|)] \times [2^{Q(|u|)}] \times [4]$, and H_u by

$$H_u(i, 2 \cdot 2^{Q(|u|)} - T) = \begin{cases} H_u\bigl(P(|u|), 2^{Q(|u|)}\bigr) & \text{if } i = P(|u|), \\ H_u(i,T) & \text{if } i < P(|u|) \end{cases} \tag{4.50}$$

for $(i,T) \in [P(|u|) + 1] \times [2^{Q(|u|)} + 1]$, and then add 1 to $Q(|u|)$. It is easy to verify that the equations (4.46) and (4.47) are still satisfied.

Now, assuming (4.48), we construct the families of functions $(g_u)_u$ and $(h_u)_u$ of 4.28. For each string u and each $(t,y) \in [0,1] \times [-1,1]$, let $T \in \mathbb{N}$, $\theta \in [0,1]$, $Y \in \mathbb{Z}$, $\eta \in [-1/4, 3/4]$

be such that $t = (T+\theta)2^{-Q(|u|)}$ and $y = (Y+\eta)B^{-j_u(T)}$, and define

$$g_u(t,y) = \begin{cases} \dfrac{2^{Q(|u|)}\pi\sin(\theta\pi)}{2B^{j_u(T)+1}} G_u\bigl(j_u(T), T, Y \bmod 4\bigr) & \text{if } \eta \le \dfrac{1}{4}, \\ \dfrac{3-4\eta}{2} g_u\left(t, \dfrac{Y}{B^{j_u(T)}}\right) + \dfrac{4\eta-1}{2} g_u\left(t, \dfrac{Y+1}{B^{j_u(T)}}\right) & \text{if } \eta \ge \dfrac{1}{4}, \end{cases} \tag{4.51}$$

$$h_u(t) = \frac{1-\cos(\theta\pi)}{2} \cdot \frac{G_u\bigl(j_u(T), T, H_u(j_u(T), T)\bigr)}{B^{j_u(T)+1}} + \sum_{i=0}^{P(|u|)} \frac{H_u(i,T)}{B^i}, \tag{4.52}$$

where $B = 2^{\lambda(|u|)+Q(|u|)+5}$. Note that the second branch of (4.51) says that when $\eta \in [1/4, 3/4]$, we define $g_u(t,y)$ by interpolating between the nearest two y at which g_u is already defined by the first branch. Equation (4.52) says that, when $\theta = 0$ (i.e., t is a multiple of $2^{-Q(|u|)}$), the value $h_u(t)$ is the real number that encodes the Tth column of Figure 4.5; as θ goes from 0 to 1, it moves to the next value along a cosine curve, whose slope, as we will verify below, matches the sine function in the first branch of (4.51). It is easy to verify that the definition is consistent; in particular, we use (4.47) to show that (4.52) stays the same for the two choices of (T, θ) when t is a multiple of $2^{-Q(|u|)}$.

Conditions (a) and (b) of 4.28 are easy to verify. We have (e) with $\rho(k) = P(k)(\lambda(k) + Q(k) + 5)$, since $h_u(1) = H_u(P(|u|), 2^{Q(|u|)})/B^{P(|u|)} = L(u)/B^{P(|u|)} = L(u)/2^{\rho(|u|)}$ by (4.52) and (4.48). Polynomial-time computability of $(g_u)_u$ can be verified using Theorem 3.24.

To see (c), observe that in the right-hand side of (4.52),

- the absolute value of the first term is bounded by $B^{-j_u(T)-1} \le B^{-j_u(T)}/32$,
- the summands corresponding to $i \le j_u(T)$ are multiples of $B^{-j_u(T)}$, and
- the summands corresponding to $i > j_u(T)$ are nonnegative numbers, each bounded by $3/B^i = 3B^{-j_u(T)}/B^{i-j_u(T)} \le 3B^{-j_u(T)}/32^{i-j_u(T)}$, and thus altogether by $3B^{-j_u(T)}/31$.

Hence, we can write $h_u(t) = (Y+\eta)B^{-j_u(T)}$ for some $\eta \in [-1/4, 1/4]$, where

$$Y = \sum_{i=0}^{j_u(T)} H_u(i,T) \cdot B^{j_u(T)-i}. \tag{4.53}$$

Since B is a multiple of 4, we have $Y \bmod 4 = H_u(j_u(T), T)$. Substituting these Y and η into (the first branch of) (4.51), we get

$$g_u\bigl(t, h_u(t)\bigr) = \frac{2^{Q(|u|)}\pi\sin(\theta\pi)}{2B^{j_u(T)+1}} G_u\bigl(j_u(T), T, H_u(j_u(T), T)\bigr). \tag{4.54}$$

This equals $h'_u(t)$ calculated from (4.52).

For the Lipschitz condition (d), note that since the value of G_u in the first branch of (4.51) is in $\{-1, 0, 1\}$, the difference between the two values of g_u in the second branch is bounded by $2 \times 2^{Q(|u|)}\pi \sin(\theta\pi)/(2B^{j_u(T)+1}) < 2^{Q(|u|)+2}/B^{j_u(T)+1}$. Thus, the slope of g_u along the second argument is at most

$$2B^{j_u(T)} \cdot \frac{2^{Q(|u|)+2}}{B^{j_u(T)+1}} = \frac{2^{Q(|u|)+3}}{B} \leq 2^{-\lambda(|u|)} \tag{4.55}$$

by our choice of B. □

The discrete initial value problem is hard

It remains to prove 4.29. At first sight, our system (4.44)–(4.47) (Figure 4.5) looks too weak to simulate a polynomial-space computation: although we have polynomial amount of memory (rows) and exponential amount of time (columns), the "chains of dependence" of values must run from top to bottom and thus are polynomially bounded in length.

Thus, we give up embedding a general PSPACE computation into Figure 4.5. Instead, we embed another PSPACE-complete problem, QBF, which asks for the truth value of the given formula u of form

$$Q_n x_n \dots Q_1 x_1 .\, \psi(x_1, \dots, x_n), \tag{4.56}$$

where ψ is a boolean formula and $Q_i \in \{\forall, \exists\}$ for each $i = 1, \dots, n$.

The truth value of (4.56) is obtained by evaluating a binary tree of depth n whose 2^n leaves each correspond to an assignment to $(x_1, \dots, x_n)$ and whose internal nodes at level i are labeled Q_i. This is roughly why it can be simulated by the tableau in Figure 4.5 despite the restriction that the dependence of values must run from top to bottom. We give a formal proof and then an example (Figure 4.6).

Proof of 4.29. We may assume that $L =$ QBF. We will construct the polynomials P, Q and families $(G_u)_u$ and $(H_u)_u$ in 4.29. Let u be of form (4.56). For each $i = 0, \dots, n$ and $n - i$ bits $b_{i+1}, \dots, b_n \in \{0, 1\}$, we write $\psi_i(b_{i+1}, \dots, b_n) \in \{0, 1\}$ for the truth value (1 for true) of the subformula $Q_i x_i \dots Q_1 x_1 .\, \psi(x_1, \dots, x_i, b_{i+1}, \dots, b_n)$, so that $\psi_0 = \psi$ and $\psi_n(\,) = \text{QBF}(u)$. We regard quantifiers as functions from $\{0, 1, 2, 3\}$ to $\{0, 1\}$:

$$\forall(2) = \exists(2) = \exists(1) = 1, \qquad \exists(0) = \forall(0) = \forall(1) = 0 \tag{4.57}$$

(the values $\forall(3)$ and $\exists(3)$ do not matter). These correspond to the meaning of the quantifiers: $\forall x$ (resp. $\exists x$) means that the subsequent formula is satisfied by 2 (resp. 1 or 2) of the two possible assignments to x. Thus,

$$Q_{i+1}\big(\psi_i(0, b_{i+2}, \dots, b_n) + \psi_i(1, b_{i+2}, \dots, b_n)\big) = \psi_{i+1}(b_{i+2}, \dots, b_n) \tag{4.58}$$

for each $i = 0, \dots, n - 1$. For $2n + 1$ bits $b_0, \dots, b_{2n} \in \{0, 1\}$, we write $\overline{b_{2n} \dots b_0}$ for the number $b_0 + 2b_1 + 2^2 b_2 + \dots + 2^{2n} b_{2n}$.

To define G_u, let

$$G_u(i, \overline{T_{2n}T_{2n-1}\dots T_{2i+2}T_{2i+1}100\dots 0}, Y) \\ = (-1)^{T_{2i+2}} \times \begin{cases} \psi_0(T_1 \oplus T_2, T_3 \oplus T_4, \dots, T_{2n-1} \oplus T_{2n}) & \text{if } i = 0, \\ Q_i(Y) & \text{otherwise,} \end{cases} \tag{4.59}$$

where $\oplus$ denotes the exclusive or; let $G_u(i, T, Y) = 0$ for other T (that is, when T is not an odd multiple of 2^{2i}). Define H_u from G_u by (4.46) and (4.47).

We prove by induction on $i = 0, \dots, n$ that $H_u(i, T) \in \{0, 1, 2\}$ for all T, as we mentioned earlier, and that

$$G_u(i, S, H_u(i, S)) = (-1)^{S_{2i+2}}\psi_i(S_{2i+1} \oplus S_{2i+2}, \dots, S_{2n-1} \oplus S_{2n}) \tag{4.60}$$

for all S of form $\overline{S_{2n}S_{2n-1}\dots S_{2i+1}100\dots 0}$ (it is immediate from the definition of G_u that $G_u(i, S, H_u(i, S)) = 0$ for other S). Once we have proved this, the case $i = n$ yields $G_u(n, 2^{2n}, H_u(n, 2^{2n})) = \psi_n() = \text{QBF}(u)$, and hence $H_u(n+1, 2^{2n}+1) = \text{QBF}(u)$. Since $n < |u|$, we can add dummy rows and columns so that $H_u(P(|u|), 2^{Q(|u|)}) = \text{QBF}(u)$ for some polynomials P and Q, as required.

The claims for $i = 0$ follow immediately from (4.46) and (4.59). Now suppose (4.60) as the induction hypothesis and fix $T = \overline{T_{2n}T_{2n-1}\dots T_0}$. Let $Y = H_u(i+1, T)$. By (4.46) and (4.47), we have

$$Y = \sum_{S=0}^{T-1} G_u\bigl(i, S, H_u(i, S)\bigr). \tag{4.61}$$

Since the assumption (4.60) implies that flipping the two bits S_{2i+2} and S_{2i+1} of any $S = \overline{S_{2n}S_{2n-1}\dots S_0}$ reverses the sign of $G_u(i, S, H_u(i, S))$, most of the nonzero summands in (4.61) cancel out. The only terms that can survive are those that correspond to $S = \overline{T_{2n}T_{2n-1}\dots T_{2i+3}00100\dots 0}$ and $S = \overline{T_{2n}T_{2n-1}\dots T_{2i+3}01100\dots 0}$. This proves $Y \in \{0, 1, 2\}$.

When $T = \overline{T_{2n}T_{2n-1}\dots T_{2i+3}100\dots 0}$, both of these terms survive, so that

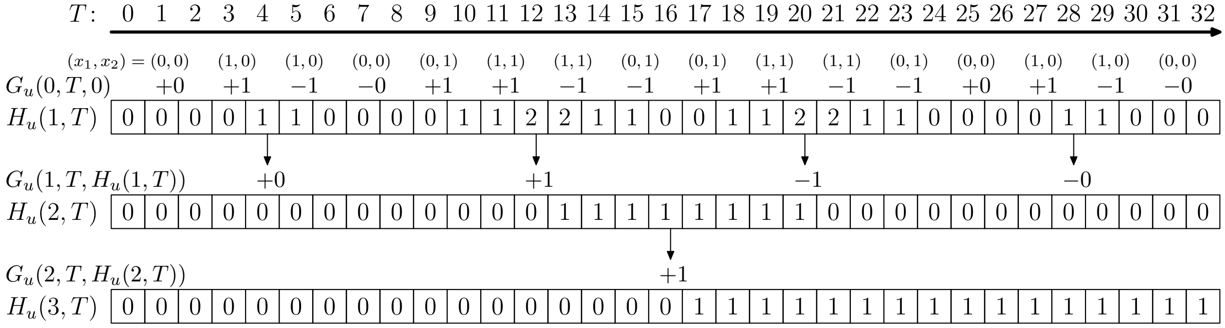
$$Y = \psi_i(0, T_{2i+3} \oplus T_{2i+4}, \dots, T_{2n-1} \oplus T_{2n}) + \psi_i(1, T_{2i+3} \oplus T_{2i+4}, \dots, T_{2n-1} \oplus T_{2n}). \tag{4.62}$$

Therefore, $Q_{i+1}(Y) = \psi_{i+1}(T_{2i+3} \oplus T_{2i+4}, \dots, T_{2n-1} \oplus T_{2n})$ by (4.58). Thus,

$$G_u(i+1, T, Y) = (-1)^{T_{2i+4}}\psi_{i+1}(T_{2i+3} \oplus T_{2i+4}, \dots, T_{2n-1} \oplus T_{2n}) \tag{4.63}$$

by (4.59), completing the induction step. □

Figure 4.6 shows the table when u be the formula $\exists x_2. \forall x_1. (x_1 \vee x_2)$. The values $G_u(0, T, 0)$ encode (redundantly) the truth table of the matrix $x_1 \vee x_2$ (first branch of (4.59)). For example, $G_u(0, T, 0) = \pm 1$ (resp. 0) for $T = 3, 5, 27, 29$ (resp. $1, 7, 25, 31$) because $(x_1, x_2) = (1, 0)$ (resp. $(0, 0)$) makes $x_1 \vee x_2$ true (resp. false). Also observe



T:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$(x_1,x_2) =$		$(0,0)$		$(1,0)$		$(1,0)$		$(0,0)$		$(0,1)$		$(1,1)$		$(1,1)$		$(0,1)$		$(0,1)$		$(1,1)$		$(1,1)$		$(0,1)$		$(0,0)$		$(1,0)$		$(1,0)$		$(0,0)$	
$G_u(0,T,0)$		+0		+1		−1		−0		+1		+1		−1		−1		+1		+1		−1		−1		+0		+1		−1		−0	
$H_u(1,T)$	0	0	0	0	1	1	0	0	0	0	1	1	2	2	1	1	0	0	1	1	2	2	1	1	0	0	0	0	1	1	0	0	0
$G_u(1,T,H_u(1,T))$					+0								+1								−1								−0				
$H_u(2,T)$	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
$G_u(2,T,H_u(2,T))$																	+1																
$H_u(3,T)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 4.6: The discrete initial value problem corresponding to the formula $u = \exists x_2.\,\forall x_1.\,(x_1 \vee x_2)$. We follow the convention in Figure 4.5 (but omit the top cells $H_u(0,T)$ which are always 0): the cells contain $H_u(i,T)$, and the signed number above the cell $H_u(i+1,T)$ indicates the increment $G_u(i,T,H_u(i,T))$ (which is 0 when omitted). The increments $G_u(0,T,0)$ for the top row are determined by the truth values of $x_1 \vee x_2$ for various assignments to (x_1,x_2).

that $H_u(1,T)$ returns to 0 every eight cells. As a result, the cell $H_u(1,4) = 1$ (resp. $H_u(1,12) = 2$) represents the fact that when x_2 is false (resp. true), $x_1 \vee x_2$ is satisfied by one (resp. two) of the assignments to x_1. Now look at the next row. The second branch of (4.59) says that for odd multiples T of 4, the values $G_u(1,T,H_u(1,T))$ are ± 1 or 0 according to whether the upper left cell has a 2 or not. Thus, they encode the smaller truth table for the subformula $\forall x_1.\,(x_1 \vee x_2)$ under each assignment to x_2. As a result, the cell $H_u(2,16) = 1$ indicates that this subformula is satisfied by one of the assignments to x_2, which causes the last row to get incremented at $T = 17$. Observe that the final cell $H_u(3,32)$ has a 1, exactly because u is true.

4.4.3 The analytic case

If g in the initial value problem (4.31) is analytic (and hence Lipschitz continuous), then so is the solution h (which is known to be unique) by the Cauchy–Kowalewsky theorem [KP02, Section 2.4].

As we mentioned in Section 4.1.2, polynomial-time computability of an analytic function is equivalent to that of its Taylor sequence. Therefore, an operator on analytic functions preserves polynomial-time computability if it is polynomial-time when viewed as an operator on Taylor series. This gives us some operators, including the initial value problem, which are not in polynomial-time with respect $\delta_\square$ but still preserve polynomial-time computability of analytic functions [KF88, MM93, Kaw]:

Theorem 4.30. *Let g: $[0,1]\times[-1,1]\to\mathbb{R}$ and h: $[0,1]\to\mathbb{R}$ be analytic functions. Assume that h takes values in $[-1,1]$ and satisfies* (4.31)*. If g is polynomial-time computable, then so is h.*

Proof. Let g and h be as assumed. There are sequences $(a_{i,j})_{(i,j)\in\mathbb{N}^2}$ and $(b_k)_{k\in\mathbb{N}}$ such that

$$g(t,y)=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}a_{i,j}\cdot t^i\cdot y^j, \qquad h(t)=\sum_{i=0}^{\infty}b_i\cdot t^i \tag{4.64}$$

for all t and y sufficiently close to the origin. By Theorem 4.6, it suffices to prove that if $(a_{i,j})_{(i,j)\in\mathbb{N}^2}$ is polynomial-time computable, then so is $(b_k)_{k\in\mathbb{N}}$.

Substituting (4.64) into (4.31), we get $b_0=0$ and

$$\sum_{k=0}^{\infty}(k+1)\cdot b_{k+1}\cdot t^k=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}a_{i,j}\cdot t^i\cdot\big(\sum_{k=0}^{\infty}b_k\cdot t^k\big)^j=\sum_{k=0}^{\infty}\sum_{i=0}^{k}\sum_{j=0}^{k-i}a_{i,j}\cdot B_{k-i,j}\cdot t^k, \tag{4.65}$$

where we put

$$B_{s,j}=\sum_{\substack{(k_1,\dots,k_j)\in\mathbb{N}^j\\ k_1+\dots+k_j=s}}b_{k_1}\cdots b_{k_j}=\begin{cases}1 & \text{if } j=0 \text{ and } s=0,\\ 0 & \text{if } j=0 \text{ and } s>0,\\ \displaystyle\sum_{k=0}^{s}b_k\cdot B_{s-k,j-1} & \text{if } j>0\end{cases} \tag{4.66}$$

for integers $j\ge 0$ and $s\ge j$. Comparing the coefficients of t^k in (4.65), we get

$$b_{k+1}=\frac{1}{k+1}\cdot\sum_{i=0}^{k}\sum_{j=0}^{k-i}a_{i,j}\cdot B_{k-i,j}. \tag{4.67}$$

Note that (4.66) and (4.67) give a mutual recurrence defining $B_{s,j}$ and b_j in the order

$$b_0, B_{0,0}, b_1, B_{1,0}, B_{1,1}, b_2, B_{2,0}, B_{2,1}, B_{2,2}, b_3, B_{3,0}, \dots. \tag{4.68}$$

In fact, they allow us to compute easily the rth term of this list with precision $2^{p(r)}\cdot\varepsilon$, for some polynomial p, if all the preceding $r-1$ terms are known to within $\varepsilon>0$, since $|a_{i,j}|$ and $|b_k|$ are bounded exponentially in their subscripts by (4.7). Using this inductively, we obtain the approximation of the rth term to precision 2^{-m} in polynomial time in $r+m$ by computing the r'th term, for $r'=1,\dots,r$, to precision $2^{-m-p(r)-p(r-1)-\dots-p(r'+1)}$. □

In fact, the proof gives a constructive method to convert the sequence $(a_{i,j})_{(i,j)\in\mathbb{N}^2}$ to $(b_k)_{k\in\mathbb{N}}$. However, since Theorem 4.6 is not constructive, this does not give us a way to $(\delta_\square,\delta_\square)$-compute h from g.

4.4.4 Related results

Table 4.1 summarizes what is known about the computability and complexity of the initial value problem (4.31) in our sense. Complexity of some other forms of differential equations

Table 4.1: Assuming that g is polynomial-time computable, how complex can the solution h of (4.31) be?

Assumptions	Upper bounds	Lower bounds
None	——	can be (non-unique and) all non-computable: [Abe71], [PR79], [Ko83]
h is the unique solution	computable: [Osg98], [PR79]	can take arbitrarily long time: [Mil70], [Ko83]
the weak Lipschitz condition in [Ko92]	exponential-space: [Ko92]	can be EXPSPACE-hard: [Kaw10, Theorem 5.5]
the Lipschitz condition (4.32)	polynomial-space: [Ko83]	can be PSPACE-hard (our Theorem 4.23)
g is analytic	polynomial-time (our Theorem 4.30)	——

with the Lipschitz condition is also discussed in [Kaw10]. Computability of other aspects of the solution is discussed by [CR04], [GBC08] and [Kaw09]. A domain-theoretic account for the problem is given in [EP04].

Computability (or not) of other classes of differential equations is studied by [PR81], [PZ97], [GZZ01], [WZ02], [WZ06] and [Zho07]. Less is known about their computational complexity.

Bibliography

[Abe71] Oliver Aberth. The failure in computable analysis of a classical existence theorem for differential equations. *Proceedings of the American Mathematical Society*, 30:151–156, 1971.

[AMT07] Tetsuo Asano, Jiří Matoušek, and Takeshi Tokuyama. The distance trisector curve. *Advances in Mathematics*, 212(1):338–360, 2007.

[BCE$^+$98] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998.

[BG11] Vasco Brattka and Guido Gherardi. Weihrauch degrees, omniscience principles and weak computability. *Journal of Symbolic Logic*, 76(1):143–176, 2011.

[BHW08] Vasco Brattka, Peter Hertling, and Klaus Weihrauch. A tutorial on computable analysis. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *New Computational Paradigms: Changing Conceptions of What is Computable*, pages 425–491. Springer, 2008.

[BJL04] Josef M. Breutzmann, David W. Juedes, and Jack H. Lutz. Baire category and nowhere differentiability for feasible real functions. *Mathematical Logic Quarterly*, 50(4–5):460–472, 2004.

[Bra03] Vasco Brattka. Computability over topological structures. In S. Barry Cooper and Sergey S. Goncharov, editors, *Computability and Models: Perspectives East and West*, pages 93–136. Kluwer Academic Publishers, New York, 2003.

[Bra05] Mark Braverman. On the complexity of real functions. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 155–164, October 2005.

[CR04] Douglas Cenzer and Jeffrey B. Remmel. Index sets for computable differential equations. *Mathematical Logic Quarterly*, 50(4-5):329–344, 2004.

[EP04] Abbas Edalat and Dirk Pattinson. A domain theoretic account of Picard's theorem. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 494–505, 2004.

[Fri84] Harvey Friedman. The computational complexity of maximization and integration. *Advances in Mathematics*, 53:80–98, 1984.

[GBC08] Daniel S. Graça, Jorge Buescu, and Manuel L. Campagnolo. Boundedness of the domain of definition is undecidable for polynomial ODEs. In *Proceedings of the Fourth International Conference on Computability and Complexity in Analysis*, volume 202 of *Electronic Notes in Theoretical Computer Science*, pages 49–57, March 2008.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[GZZ01] William Gay, Bing-Yu Zhang, and Ning Zhong. Computability of solutions of the Korteweg–de Vries equation. *Mathematical Logic Quarterly*, 47(1):93–110, 2001.

[Hoo90] H. James Hoover. Feasible real functions and arithmetic circuits. *SIAM Journal on Computing*, 19(1):182–204, 1990.

[Hoo91] H. James Hoover. Real functions, contraction mappings, and P-completeness. *Information and Computation*, 93(2):333–349, 1991.

[Kaw] Akitoshi Kawamura. Complexity of initial value problems. Submitted.

[Kaw09] Akitoshi Kawamura. Differential recursion. *ACM Transactions on Computational Logic*, 10(3):22:1–22, 2009.

[Kaw10] Akitoshi Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 2010.

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

[KC10] Akitoshi Kawamura and Stephen Cook. Complexity theory for operators in analysis. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC 2010)*, pages 495–502, June 2010.

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

[KF88] Ker-I Ko and Harvey Friedman. Computing power series in polynomial time. *Advances in Applied Mathematics*, 9:40–50, 1988.

[KMT10] Akitoshi Kawamura, Jiří Matoušek, and Takeshi Tokuyama. Zone diagrams in Euclidean spaces and in other normed spaces. In *Proceedings of the 26th Annual ACM Symposium on Computational Geometry*, June 2010.

[Ko83] Ker-I Ko. On the computational complexity of ordinary differential equations. *Information and Control*, 58:157–194, 1983.

[Ko91] Ker-I Ko. *Complexity Theory of Real Functions*. Birkhäuser Boston, 1991.

[Ko92] Ker-I Ko. On the computational complexity of integral equations. *Annals of Pure and Applied Logic*, 58(3):201–228, November 1992.

[Ko98] Ker-I Ko. Polynomial-time computability in analysis. In Iurii Leonidovich Ershov et al., editors, *Handbook of Recursive Mathematics: Volume 2: Recursive Algebra, Analysis and Combinatorics*, volume 139 of *Studies in Logic and the Foundations of Mathematics*, pages 1271–1317. North-Holland, December 1998.

[KP02] Steven G. Krantz and Harold R. Parks. *A Primer of Real Analytic Functions*. Birkhäuser Advanced Texts. Birkhäuser Boston, second edition, June 2002.

[KY08] Ker-I Ko and Fuxiang Yu. On the complexity of convex hulls of subsets of the two-dimensional plane. In *Proceedings of the Fourth International Conference on Computability and Complexity in Analysis (CCA 2007)*, volume 202 of *Electronic Notes in Theoretical Computer Science*, pages 121–135, March 2008.

[Lam06] Branimir Lambov. The basic feasible functionals in computable analysis. *Journal of Complexity*, 22(6):909–917, December 2006.

[Meh76] Kurt Mehlhorn. Polynomial and abstract subrecursive classes. *Journal of Computer and System Sciences*, 12(2):147–178, April 1976.

[Mil70] Webb Miller. Recursive function theory and numerical analysis. *Journal of Computer and System Sciences*, 4:465–472, 1970.

[MM93] N. Th. Müller and B. Moiske. Solving initial value problems in polynomial time. In *22as Jornadas Argentinas de Informática e Investigación Operativa (JAIIO 1993)*, 1993.

[Mül86] N. Th. Müller. Subpolynomial complexity classes of real functions and real numbers. In *Proceedings of the 13th International Colloquium on Automata, Languages and Programming (ICALP 1986)*, volume 226 of *Lecture Notes in Computer Science*, pages 284–293, 1986.

[Mül87] N. Th. Müller. Uniform computational complexity of Taylor series. In *Proceedings of the 14th International Colloquium on Automata, Languages and Programming (ICALP 1987)*, volume 267 of *Lecture Notes in Computer Science*, pages 435–444, 1987.

[Osg98] W. F. Osgood. Beweis der Existenz einer Lösung der Differentialgleichung $\frac{dy}{dx} = f(x, y)$ ohne Hinzunahme der Cauchy-Lipschitz'schen Bedingung. *Monatshefte für Mathematik*, 9(1):331–345, 1898.

[PR79] Marian Boykan Pour-El and Ian Richards. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17(1–2):61–90, 1979.

[PR81] Marian Boykan Pour-El and Ian Richards. The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics*, 39(3):215–239, 1981.

[PR89] Marian B. Pour-El and Jonathan I. Richards. *Computability in Analysis and Physics*. Perspectives in Mathematical Logic. Springer, 1989.

[PZ97] Marian Boykan Pour-El and Ning Zhong. The wave equation with computable initial data whose unique solution is nowhere computable. *Mathematical Logic Quarterly*, 43:499–509, 1997.

[Sch82] Arnold Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical Report, University of Tübingen, 1982.

[Sch02] Matthias Schröder. Extended admissibility. *Theoretical Computer Science*, 284(2):519–538, July 2002.

[Set92] Anil Seth. There is no recursive axiomatization for feasible functionals of type 2. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science (LICS 1992)*, June 1992.

[Set93] Anil Seth. Some desirable conditions for feasible functionals of type 2. In *Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS 1993)*, July 1993.

[Tak01] Izumi Takeuti. Effective fixed point theorem over a non-computably separable metric space. In Jens Blanck, Vasco Brattka, and Peter Hertling, editors, *Computability and Complexity in Analysis*, volume 2064 of *Lecture Notes in Computer Science*, pages 310–322. Springer, 2001.

[Wei92] Klaus Weihrauch. The degrees of discontinuity of some translators between representations of the real numbers. Technical Report TR-92-050, International Computer Science Institute, Berkeley, July 1992.

[Wei00] Klaus Weihrauch. *Computable Analysis: An Introduction.* Texts in Theoretical Computer Science. Springer, 2000.

[Wei03] Klaus Weihrauch. Computational complexity on computable metric spaces. *Mathematical Logic Quarterly*, 49(1):3–21, 2003.

[WZ02] Klaus Weihrauch and Ning Zhong. Is wave propagation computable or can wave computers beat the Turing machine? *Proceedings of the London Mathematical Society*, 85(2):312–332, 2002.

[WZ06] Klaus Weihrauch and Ning Zhong. Computing Schrödinger propagators on type-2 Turing machines. *Journal of Complexity*, 22(6):918–935, 2006.

[Zho07] Ning Zhong. Computable analysis of a boundary-value problem for the Korteweg–de Vries equation. *Theory of Computing Systems*, 41(1):155–175, 2007.

[ZM08] Xishun Zhao and Norbert Müller. Complexity of operators on compact sets. In *Proceedings of the Fourth International Conference on Computability and Complexity in Analysis (CCA 2007)*, volume 202 of *Electronic Notes in Theoretical Computer Science*, pages 101–119, March 2008.

Index