TECHNISCHE
UNIVERSITÄT
DARMSTADT

Parametrised Complexity of Functionals on Spaces of Real Functions

Eike Neumann

Bachelor-Thesis

October 2012

Parametrisierte Komplexität von Funktionalen auf Räumen reeller Funktionen
Parametrised Complexity of Functionals on Spaces of Real Functions

Vorgelegte Bachelor-Thesis von Eike Neumann

1. Gutachten: Prof. Dr. Martin Ziegler
2. Gutachten: Prof. Dr. Ulrich Kohlenbach

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 29.10.12

____

(Eike Neumann)

Abstract.
Using Kawamura's and Cook's extension of Weihrauch's TTE, we study the uniform complexity of integration, maximisation and function evaluation on the space of continuous- and the space of real analytic functions over a compact interval. Building on results due to Kawamura, Müller, Rösnick and Ziegler we introduce a notion of 'efficiently constructible functions', which covers a wide range of real functions encountered in practical numerical analysis and prove that maximisation restricted to the class of efficiently constructible functions and integration restricted to a smaller subclass are uniformly polynomial-time computable, thus providing a more differentiated view on the classic result due to Ko and Friedman which asserts that maximisation and integration of polynomial-time computable functions are computationally hard unless $\mathbf{P} = \mathbf{NP}$. We also give a quantitative improvement of the aforementioned results of Kawamura, Müller, Rösnick and Ziegler.

Contents

1. Introduction and Summary

Subject of the present work is the study of the computational complexity of integration, maximisation and function evaluation on the space $C([0,1])$ of continuous functions over the compact interval $[0,1]$. More precisely, we will consider the functionals

$$\begin{array}{llclll}
\mathbf{EVAL}\colon & C([0,1]) & \times & [0,1] & \to \mathbb{R}, & (f,x) \mapsto f(x) \\
\mathbf{MAX}\colon & C([0,1]) & \times & [0,1]^2 & \to \mathbb{R}, & (f,a,b) \mapsto \max\{f(x) \mid \min\{a,b\} \leq x \leq \max\{a,b\}\} \\
\mathbf{INT}\colon & C([0,1]) & \times & [0,1]^2 & \to \mathbb{R}, & (f,a,b) \mapsto \int_a^b f(x)dx
\end{array}$$

Classic results concerning the general difficulty of computing these functionals are due to Ko and Friedman, who used the following notion of polynomial-time computability (cf. [Ko91])

Definition 1.1 (Real Complexity due to Ko and Friedman)**.** (i) Let $x \in \mathbb{R}$ be a real number. Let $\mathbb{D} = \bigcup_{n\in\mathbb{N}} \mathbb{D}_n$ denote the set of dyadic rational numbers, where $\mathbb{D}_n := \{\frac{c}{2^n} \mid c \in \mathbb{Z}\}$. A function $\phi\colon \mathbb{N} \to \mathbb{D}$ is said to **binary converge** to x, if for all $n \in \mathbb{N}$ we have $\phi(n) \in \mathbb{D}_n$ and $|\phi(n) - x| \leq 2^{-n}$. The set of all functions binary converging to f is denoted by $\mathbf{CF}_x$.

(ii) Let $x \in \mathbb{R}$ be a real number. We say that x is **computable** if there exists a computable $\phi \in \mathbf{CF}_x$. It is **polynomial-time computable** if there exists a polynomial-time computable $\phi \in \mathbf{CF}_x$.

(iii) Let I be a compact interval in $\mathbb{R}$. A function $f\colon I \to \mathbb{R}$ is **computable** if there exists a function-oracle Turing Machine such that for each $x \in I$ and every $\phi \in \mathbf{CF}_x$, the function ψ computed by M with oracle ϕ is in $\mathbf{CF}_{f(x)}$.

(iv) If $f\colon I \to \mathbb{R}$ is a computable function, then we say that **the uniform time complexity of f is bounded by** $T\colon \mathbb{N} \to \mathbb{N}$ if there exists a function-oracle Turing Machine M which computes f, such that for every $x \in I$ and every $\phi \in \mathbf{CF}_x$ the number of computational steps that M performs on input n with oracle ϕ is smaller or equal than $T(n)$. Consequently, f is said to be **computable in polynomial time** if its uniform time complexity is bounded by some polynomial $P \in \mathbb{N}[x]$.

One can then show that the following relations hold

Theorem 1.2 ([Ko91])**.** *(i) For each polynomial-time computable $f \in C([0,1])$, the function $g(x) := \max_{y\leq x} f(y)$ is again polynomial-time computable if and only if* $\mathbf{P} = \mathbf{NP}$. *The same result holds true even if one assumes f to be infinitely-often differentiable.*

(ii) For each polynomial-time computable $f \in C([0,1]$, the function $g(x) := \int_0^x f(t)dt$ is again polynomial-time computable if and only if $\mathbf{FP} = \#\mathbf{P}$. *The same result holds true even if one assumes f to be infinitely-often differentiable.*

Since it is widely believed that $\mathbf{P} \neq \mathbf{NP}$, these results are generally considered negative. The proof of these results is based on a construction, which assigns to each $\mathbf{NP}$-language

$$L(p,A) = \{u \in \{0,1\}^* \mid \exists v.|v| \leq p(|u|),\ (u,v) \in A, \}, \quad A \in \mathbf{P},\ \ p \in \mathbb{N}[x]$$

a polynomial-time computable function f_L such that maximising f_L is equivalent to deciding for a given $u \in \{0,1\}^*$, whether there exists a certificate v such that $(u,v) \in A$ and integrating f_L is equivalent to counting the number of such certificates. The thus constructed functions f_L are, however, highly artificial and not representative of functions encountered in practical applications.

In fact, most 'efficiently computable' functions encountered in every-day mathematics are not only computable in polynomial time, but can be efficiently constructed from basic functions, such as polynomials, rational functions, functions such as $|x|$ or $\sqrt{x}$ or analytic functions such as $\sin, \cos$ or $\exp$ by means of composition and elementary algebraic operations. This motivates the following

Definition 1.3. The class of **efficiently constructible real functions** is defined as follows:

- If f is a polynomial-time computable real analytic function over a compact interval $[a,b] \subseteq \mathbb{R}$, then f is efficiently constructible.
- f is an efficiently constructible function over the interval I, such that $f(x) \geq 0$ for all $x \in I$, then $\sqrt{f}$ is efficiently constructible as well. If $f(x) \geq 1$ for all $x \in I$, then $1/f$ is efficiently constructible.
- If f and g are efficiently constructible functions defined on the same interval I then so are their sum $f+g$ and their product $f \cdot g$. If $a \in I$ is a polynomial-time computable number such that $f(a) = g(a)$ then
$$h_a(x) := \begin{cases} f(x) & \text{if } x \leq a \\ g(x) & \text{if } x \geq a \end{cases}$$
is efficiently constructible as well.
- If f, g are efficiently constructible functions such that the range of g is contained in the domain of f, then the composition $f \circ g$ is efficiently constructible.

This is of course a completely 'ad-hoc' definition not designed to capture the general intuitive notion of what it means for a function to be 'efficiently constructible' - it does, however, seem to cover a large class of functions encountered in real analysis. We will show that these 'efficiently constructible' functions are indeed efficiently constructible in a more precise sense and that the following theorem holds

Theorem 1.4. *Let $f \in C([0,1])$ be an efficiently constructible function. Then f and*
$$\mathbf{MAX}(f,\cdot,\cdot)\colon [0,1] \times [0,1] \to \mathbb{R},\ (a,b) \mapsto \max\{f(x) \mid x \in \min\{a,b\} \leq x \leq \max\{a,b\}\}$$
are polynomial-time computable functions. If the construction of f does not involve divisions or square-roots, then
$$\mathbf{INT}(f,\cdot,\cdot)\colon [0,1] \times [0,1] \to \mathbb{R}\ (a,b) \mapsto \int_a^b f(x)dx$$
is polynomial-time computable as well.

In fact, we will not only show that for these functions maximisation is non-uniformly computable in polynomial-time, but also provide a polynomial-time algorithm which uniformly computes the maximum of an efficiently constructible function, given an appropriate 'encoding' of the basic functions. To state this more precisely, we need to introduce the notion of *representations*.

In [KC10], Kawamura and Cook have introduced an extension of Weihrauch's Type Two Theory of Effectivity (TTE) [Wei00], introducing a notion of 'size' of represented objects. In [Wei00], uncountable sets are represented as 'infinite strings' (i.e. elements of Cantor-Space) via partial mappings

$$\delta\colon \subseteq \Sigma^\omega \to X,$$

where Σ is some finite alphabet. A Turing Machine is then said to compute a function $f\colon X \to Y$ (with respect to some representation α of X and some representation β of Y) if, while reading an infinite string[†] s representing some element $x \in X$ (in the sense that $\alpha(s) = x$) it produces some infinite string t representing $f(x) \in Y$ (in the sense that $\beta(t) = y$).

In [KC10], uncountable sets are represented as string functions (i.e. elements of Baire-Space), essentially via partial mappings

$$\delta\colon \subseteq (\Sigma^*)^{\Sigma^*} \to X.$$

The size a name $\psi \in \operatorname{dom}\delta$ can then be defined as $|\psi|(n) = \max_{|u|=n} |\psi(u)|$ [‡]. Computability is then defined essentially like in Definition 1.1: a machine computes a function $f\colon X \to Y$ with respect to some representation α of X and some representation β of Y if on input $u \in \Sigma^*$ and with oracle $\psi \in \operatorname{dom}\alpha$, where $\alpha(\psi) = x$ it outputs $\phi(u)$ where $\beta(\phi) = y$. In this definition we deal with two different kinds of inputs: the type-zero input u, given as the 'usual' input, and the type-one input ϕ, given as an oracle. The notion of 'polynomial-time computability' can then be generalised by considering so called 'second-order polynomials': intuitively speaking, polynomials in the type-zero- and in the type one input, e.g. $P(x,\ell) = 23x^3 + 42x^2\ell(x) + 1729x\ell(\ell(x))^2$, which evaluate to a positive integer $P(n, |\psi|) = 23n^3 + 42n^2|\psi|(n) + 1729n|\psi|(|\psi|(n))^2$. A machine is said to compute $f\colon X \to Y$ in second-order-polynomial time, if its running time on input u and with oracle ψ is bounded by $P(|u|, |\psi|)$ for some second-order polynomial P.

This approach allows for treating real functions (and other uncountable structures) as inputs, just like real numbers, and thus for stating *uniform* complexity results for operators and functionals, explicitly specifying what information is needed on the input function in order to perform a particular computational task and how the resources required during the computation depend on said function. Furthermore, such results yield explicit and well-defined algorithms for solving computational problems in numerical analysis.

In order to establish Theorem 1.4, we define representations $\delta_{C([0,1]),\mathfrak{pp}}$ and $\delta_{C([0,1]),\mathfrak{pq}}$ of the space $C([0,1])$ of continuous functions over the compact interval $[0,1]$ via fast converging sequences of piecewise polynomial- and rational functions with dyadic coefficients and prove maximisation, integration and evaluation, as well as the operations mentioned in Theorem 1.4 second-order polynomial time computable with respect to these representations. It then suffices to show that every polynomial-time computable real analytic

[†]A Turing Machine reading and writing infinite strings is usually referred to as a *Type Two Turing Machine*

[‡]we will have to refine this definition slightly, since, like in discrete complexity, one wants to obtain time-constructible complexity-bounds, which require the size of ψ to be 'efficiently computable' in terms of ψ

function has a polynomial-time computable $\delta_{C([0,1]),\mathsf{pp}}$-name. This is established using a result due to Kawamura, Müller, Rösnick and Ziegler [KMRZ12], who have given several natural (and equivalent) representations of the space of real-analytic functions, rendering various basic operators and functionals (including the functionals **MAX**, **EVAL** and **INT**) uniformly second-order polynomial-time computable, and have shown that every polynomial-time computable real-analytic function has a polynomial-time computable name with respect to these representations. We then show that a $\delta_{C([0,1]),\mathsf{pp}}$-name of a real analytic function is computable in second-order polynomial time, given one of these representations.

In Section 2 we give a brief introduction to the computational model due to Kawamura and Cook. Section 3 contains some general considerations of how to construct representations of separable metric spaces which render a given functional computable in second-order polynomial-time, requiring as few information on the represented function as possible. As mentioned before, we will represent continuous functions by fast converging Cauchy-sequences of piecewise polynomial- and rational functions with dyadic rational coefficients. In Section 4 we study the computational complexity of basic algorithms on these functions, which will then in Section 5 be used to prove uniform complexity results for operators and functionals on the space of continuous functions over a compact interval. This will be the most important step in proving Theorem 1.4. In Section 6 we discuss the results of Kawamura, Müller, Rösnick and Ziegler from a quantitative viewpoint and conclude the proof of Theorem 1.4.

Acknowledgement. I would like to thank my advisor, Martin Ziegler, for his helpful remarks and suggestions.

2. The Computational Model

In the following, fix some finite alphabet Σ containing 0 and 1.

Definition 2.1. A function $\psi\colon \Sigma^* \to \Sigma^*$ is called **regular** if

$$|\psi(1^n)| = \max\{|\psi(u)| \mid |u| \leq n\}.$$

We write $\mathfrak{Reg}$ for the class of all regular functions. For all $\psi \in \mathfrak{Reg}$ we define the function

$$|\psi|\colon \mathbb{N} \to \mathbb{N},\ n \mapsto |\psi(1^n)|.$$

It is called the **size** or **length** of ψ.

We are going to use $\mathfrak{Reg}$ as the domain for representations. Observe, that for every $\psi \in \mathfrak{Reg}$, the size-function $|\psi|$ is monotone and a Turing-machine given ψ as an Oracle can compute $|\psi|(u)$ in time $\mathcal{O}(|u| + |\psi|(u))$. These properties are crucial for obtaining a sensible notion of computational complexity. For our purpose, the restriction of ${\Sigma^*}^{\Sigma^*}$ to $\mathfrak{Reg}$ entails no loss of generality, since every function $\varphi\colon\ \Sigma^* \to \Sigma^*$ induces a function $\check{\varphi} \in \mathfrak{Reg}$, from which φ can be efficiently recovered, via

$$\check{\varphi}(0u) = \varphi(u),\ \check{\varphi}(1u) = 1^{\max\{|\varphi(v)| \mid |v| \leq |u|+1\}}.$$

For a regular function ψ we denote by $\hat{\psi}$ the function

$$\psi(0\cdot)\colon\ \Sigma^* \to \Sigma^*\ u \mapsto \psi(0u).$$

By convention, we will 'store' all the information relevant to the representation in $\hat{\psi}$ and use $\psi(1\cdot)$ to encode the size of ψ.

Remark 2.2. Kawamura and Cook define the notion of size slightly differently. In [KC10] a function $\psi\colon \Sigma^* \to \Sigma^*$ is called regular if for all $u, v \in \Sigma^*$ we have

$$|\psi(u)| \leq |\psi(v)| \text{ whenever } |u| \leq |v|.$$

The function $|\psi|$ is then simply defined by putting $|\psi|(n) = |\psi(u)|$ for some u with $|u| = n$. Similarly to our approach, any function can be 'made regular' - in this case by 'padding out' the function values by redundant information. The disadvantage of this approach is that recovering the original function becomes somewhat less straightforward.

Definition 2.3. (i) Let X be some non-empty, at most countable set. A surjective partial mapping $\nu\colon\ \subseteq \Sigma^* \to X$ is called a **notation** of X. If $x \in X$ and $u \in \Sigma^*$ satisfies $\nu(u) = x$ we call u a **ν-name** of x.

(ii) Let X be some non-empty set of cardinality at most $2^{\aleph_0}$. A surjective partial mapping $\alpha\colon\ \subseteq \mathfrak{Reg} \to X$ is called a **representation** of X. If $x \in X$ and $\psi \in \mathfrak{Reg}$ satisfies $\alpha(\psi) = x$ we call ψ an **α-name** of x.

Remark 2.4. Let $\nu\colon\ \subseteq \Sigma^* \to X$ be a notation of X. Then ν induces a representation $\tilde{\nu}$ of X with

$$\operatorname{dom} \tilde{\nu} = \{\psi \mid \exists a \in \operatorname{dom} \nu.\forall u \in \Sigma^*.\psi(u) = a\}$$

and $\tilde{\nu}(\psi) = \nu(\psi(0))$. We will always simply write ν for both the notation and the induced representation $\tilde{\nu}$ as from the context no confusion may arise.

Example 2.5. (i) A $\nu_{\mathbb{N}}$-name of $n \in \mathbb{N}$ is a string $b_k \cdot \cdots \cdot b_0$ over the alphabet $\{0, 1\}$ such that $n = \sum_{i=0}^{k} b_i 2^i$.

(ii) A ν_1-name of $n \in \mathbb{N}$ is the string 1^n.

(iii) A $\nu_{\mathbb{D}}$-name of $c \in \mathbb{D}$ is a string over the alphabet $\{0, 1, +, -, .\}$ of the form $\sigma b_{-N} \dots b_0.b_1 \dots b_M$, where $\sigma \in \{+, -\}$ and $b_i \in \{0, 1\}$ satisfying $c = \sigma \sum_{k=-N}^{M} b_k 2^{-k}$. If $b_{-N} = b_M = 1$, we write $prec(c)$ for the number M. Unconditionally, we denote $N + M$ by $\ell(c)$ [†].

(iv) A $\delta_{\mathbb{N}\to\mathbb{N}}$-name of $f \in \mathbb{N}^{\mathbb{N}}$ is a function $\psi \in \mathfrak{Reg}$ such that for all $u \in \{0, 1\}^*$, $\psi(0u)$ is a $\nu_{\mathbb{N}}$-name of $f(\nu_{\mathbb{N}}(u))$.

(v) A ρ-name of $x \in \mathbb{R}$ is a function $\psi \in \mathfrak{Reg}$ such that $\psi(u) = \psi(1^n)$ whenever $|u| = n$ and $\psi(1^n)$ is a $\nu_{\mathbb{D}}$-name of some $c \in \mathbb{D}$ with $|c - x| \leq 2^{-n}$.

Definition 2.6. (i) Let $(\alpha_i)_{i\in I}$ be an at most countable family of representations of sets $(X_i)_{i\in I}$, I being notated by ν_I. The **product representation** $\prod_I \alpha_i$ of the Cartesian product $\prod_I X_i$ with respect to ν_I is defined as follows: ψ is a $\prod_I \alpha_i$-name of $\vec{x} \in \prod_I X_i$ if $\hat{\psi}(\bar{u}\#\bar{\cdot}) = \phi(\cdot)$, where

$$\hat{\psi}(\bar{u}\#\bar{\cdot}) \colon \Sigma^* \to \Sigma^*,\ v \mapsto \hat{\psi}(\bar{u}\#\bar{v}),$$

and ϕ is an $\alpha_{\nu_I(u)}$-name of x_u, where for $e \in \Sigma$, $\bar{e} := e \cdot e$, for $u, v \in \Sigma^*$, $\overline{u \cdot v} := \bar{u} \cdot \bar{v}$ and $\# := 10$.

(ii) Let $\alpha_1, \dots, \alpha_m$ be a finite family of representations of $X_1, \dots, X_m$. The **product representation** $\alpha_1 \times \dots \times \alpha_m$ of $X_1 \times \dots \times X_m$ is the product $\prod_{\mathbb{N}} \alpha_i$ of the family $(\alpha_i)_{i\in\mathbb{N}}$ with respect to ν_1, where $\alpha_i := \alpha_1$ for $i > m$.

(iii) Let α be a representation of X. The **star representation** α^* of the free monoid X^* over X is defined as follows: ϕ is an α^*-name of $x = x_1 \cdot \dots \cdot x_m \in X^*$ if it is a $\nu_1 \times \alpha \times \dots \times \alpha$-name of $m \in \mathbb{N}$ and $(x_1, \dots, x_m) \in X^m$.

(iv) Let α be a representation of X. The representation α^ω of X^ω is the product representation $\prod_{\mathbb{N}} \alpha$ with respect to ν_1.

Definition 2.7. Let $F\colon \subseteq X \rightrightarrows Y$ be a partial multifunction, let α and β be representations of X and Y respectively. A function $f\colon \operatorname{dom}\alpha \to \operatorname{dom}\beta$ is called an (α, β)**-realiser** of F if for every $\psi \in \operatorname{dom}\alpha$ such that $\alpha(\psi) \in \operatorname{dom} F$ we have

$$\beta(f(\psi)) \in F[\alpha(\psi)]$$

i.e. if the following diagram commutes.

$$\begin{tikzcd} \operatorname{dom}\alpha \arrow[r, "f"] \arrow[d, "\alpha"] & \operatorname{dom}\beta \arrow[d, "\beta"] \\ X \arrow[r, "F"] & Y \end{tikzcd}$$

Definition 2.8. The set of **second-order-polynomials** is recursively defined as follows

(i) 1 and x are second-order-polynomials.

(ii) if P and Q are second-order-polynomials, then so are $P + Q$, $P \cdot Q$ and $\ell(P)$.

If P is a second-order-polynomial, then the **evaluation of** P in $n \in \mathbb{N}$ and $f \in \mathbb{N}^{\mathbb{N}}$, denoted $P(n, f)$, is defined as the value of the expression obtained by replacing everywhere in P the term $\ell(\cdot)$ by $f(\cdot)$ and x by n.

[†]note that this is an abuse of notation, since it suggests independence of the underlying name, which is clearly not the case. However, it will always be clear from the context to which name we refer

Definition 2.9. A function $f\colon \subseteq \mathfrak{Reg} \to \mathfrak{Reg}$ is **computable** if there exists an Oracle-Turing-machine which on input $u \in \Sigma^*$ and with oracle ψ outputs $f(\psi)(u)$. It is **computable in second order polynomial-time** or $\mathbf{FP}^2$-computable, if there exists some second-order polynomial $P(x, \ell)$ such that $P(|u|, |\psi|)$ bounds the running time of the machine computing f on input u and with oracle ψ. A partial multifunction $F\colon \subseteq X \rightrightarrows Y$ is (α, β)**-computable** resp. **second-order polynomial-time** (α, β)**-computable** (or (α, β)-$\mathbf{FP}^2$) if it has a computable resp. second-order polynomial time computable realiser.

Definition 2.10. Let α, β be representations of the same set X. We say that α **reduces** to β and write $\alpha \preceq \beta$ if the identity id_X has a second-order polynomial time computable (α, β)-realiser. We say that α and β are **equivalent** and write $\alpha \equiv \beta$ if $\alpha \preceq \beta$ and $\beta \preceq \alpha$. We say that the reduction $\alpha \preceq \beta$ is **strict**, if α and β are not equivalent.

Proposition 2.11. *(i) If $F\colon \subseteq X \rightrightarrows \operatorname{dom} G \subseteq Y$ is (α, β)-$\mathbf{FP}^2$ and $G\colon \subseteq Y \rightrightarrows Z$ is (β, γ)-$\mathbf{FP}^2$, then their composition $F \circ G \subseteq X \rightrightarrows Z$ is (α, γ)-$\mathbf{FP}^2$.*
(ii) If $\alpha \preceq \beta$ and $F\colon \subseteq X \rightrightarrows Y$ is (β, γ)-$\mathbf{FP}^2$, then F is (α, γ)-$\mathbf{FP}^2$.

3. Representations of Separable Metric Spaces

Consider a function

$$\mathfrak{F}\colon X \times A \to Y,$$

where X, A and Y are metric spaces, A is compact and separable, Y is separable and representations γ of A and β of Y have been fixed. One might ask whether it is possible to find some canonical representation α of X, such that $\mathfrak{F}$ is $(\alpha \times \gamma, \beta)$-$\mathbf{FP}^2$ and every representation of X rendering $\mathfrak{F}$ second-order polynomial time computable reduces to α. Let us fix this notion for further reference.

Definition 3.1. Let α be a representation of X. We say that α **uniformly characterises $\mathfrak{F}$ with respect to β and with parameters represented by γ** if $\mathfrak{F}$ is $(\alpha \times \gamma, \beta)$-$\mathbf{FP}^2$ and if every representation δ of X, such that $\mathfrak{F}$ is $(\delta \times \gamma, \beta)$-$\mathbf{FP}^2$ reduces to α.

The function $\mathfrak{F}$ induces an equivalence relation on X via

$$x \sim_{\mathfrak{F}} y \Leftrightarrow \forall a \in A.(\mathfrak{F}(x, a) = \mathfrak{F}(y, a)).$$

A representation α of X induces a representation $\alpha/\sim_{\mathfrak{F}}$ of $X/\sim_{\mathfrak{F}}$ via

$$\alpha/\sim_{\mathfrak{F}} := \pi_{\mathfrak{F}} \circ \alpha,$$

where $\pi_{\mathfrak{F}}\colon X \to X/\sim_{\mathfrak{F}}$ is the canonical projection. Note that the function

$$\bar{\mathfrak{F}}\colon (X/\sim_{\mathfrak{F}}) \times A \to Y,\ \bar{\mathfrak{F}}(\pi_{\mathfrak{F}}(x), a) := \mathfrak{F}(x, a)$$

is well-defined. Our next goal is to show, that there exists a representation α of $X/\sim_{\mathfrak{F}}$ uniformly characterising $\bar{\mathfrak{F}}$ under fairly general assumptions, so long the representations of A and Y reflect the separability of A and Y and the compactness of A.

Definition 3.2. Let X be a separable metric space, and let S be a dense, countable subset of X, notated by $\sigma\colon \subseteq \Sigma^* \to S$. The **Cauchy representation of X induced by** σ, denoted $\delta(X, \sigma)$ is defined as follows. A function $\psi \in \mathfrak{Reg}$ is a $\delta(X, \sigma)$-name of $x \in X$ if for all $n \in \mathbb{N}$ $\psi(u) = \psi(1^n)$, whenever $|u| = n$ and $\psi(1^n)$ is a σ-name of some $a \in S$ satisfying $d_X(a, x) \leq 2^{-n}$. A representation of X is called **a Cauchy representation** if it is equivalent to a Cauchy representation induced by some notation of a dense, countable subset of X.

Remark 3.3. A similar notion of 'Cauchy representations' is can be found in [Wei00] (Definition 8.1.2).

Example 3.4. $\rho\colon \mathfrak{Reg} \to \mathbb{R}$, defined in example 2.5, is the Cauchy representation induced by $\nu_{\mathbb{D}}$.

Definition 3.5. Let α be a Cauchy representation of a complete separable metric space X. We say that α is **compact** if there exists $M : \mathbb{N} \to \mathbb{N}$ such that every $x \in X$, has an α-name whose size is bounded by M, i.e. for all x there exists $\psi \in \operatorname{dom} \alpha$ such that $\alpha(\psi) = x$ and $|\psi|(n) \leq M(n)$ for all n. The representation α is called **efficiently compact** if M is (ν_1, ν_1)-computable in polynomial time. More generally, α is called **locally compact**, if the co-restriction $\alpha|^K$ to any compact set $K \subseteq X$ is compact and **locally efficiently compact** if the co-restriction to any compact set is efficiently compact.

Example 3.6. Every real number x has a ρ-name of size $\lfloor\log(|x|+1)\rfloor + n + 3$. In particular, every number in a compact interval $I = [a, b]$ has a ρ-name of size

$$\max\{\lfloor\log(|a|+1)\rfloor, \lfloor\log(|b|+1)\rfloor\} + n + 3.$$

So the co-restriction $\rho|^I$ of ρ to any compact interval is efficiently compact and ρ is locally efficiently compact.

The above example motivates the definition of a new representation ρ_c of $\mathbb{R}$ as follows: regular ψ is a ρ_c-name of $x \in \mathbb{R}$ if it is a ρ name of x of size at most $\lfloor\log(|x|+1)\rfloor + n + 3$. This new representation is also a Cauchy representation: it is equivalent to ρ. In fact, we have

Lemma 3.7 (truncation lemma)**.** *Let $x \in \mathbb{R}$ and $c \in \mathbb{D}$ with $prec(c) = n + m$, $m \geq 1$, such that*

$$|x - c| \leq 2^{-n-1}.$$

Given a $\nu_{\mathbb{D}}$-name of c we can compute a $\nu_{\mathbb{D}}$-name of $c' \in \mathbb{D}$ of size $\lfloor\log(x)\rfloor + n$ satisfying

$$|x - c'| \leq 2^{-n}$$

in time $\mathcal{O}(n + m + \lfloor\log(x)\rfloor)$.

Proof. Compute the unique $\bar{c} \in [-2^{-m-1}, 2^{-m-1}]$ satisfying

$$\bar{c} \equiv 2^{n+m}c \pmod{2^m},$$

then put $c' := c - \frac{\bar{c}}{2^{n+m}}$. Note that $prec(c') = n$ and that

$$|c' - x| \leq |c' - c| + |c - x| \leq 2^{-n}.$$

In the worst case this procedure requires one addition of numbers of size smaller or equal than $n + m + \lfloor\log(|x|+1)\rfloor + 1$. □

We immediately deduce

Proposition 3.8. *$\rho \equiv \rho_c$.*

The notion 'compact' is motivated by the following proposition.

Proposition 3.9. *Let X be a complete separable metric space. Then the following are equivalent*

(i) X has a compact Cauchy representation.
(ii) X is compact.

Proof. (i) ⇒ (ii). Let $\alpha = \delta(X, \sigma)$ be a compact Cauchy representation of X. Consider the mapping

$$\alpha' \colon \subseteq (\Sigma^*)^\omega \to X$$

with $\operatorname{dom}\alpha' = \{(a_n)_n \in (\Sigma^*)^\omega \mid (\sigma(a_n))_n \text{converges in } X\}$ and $\alpha'((a_n)_n) = \lim_n \sigma(a_n)$. Note that if $(\sigma(a_n)_n)$ converges with order 2^{-n} then $\alpha'((a_n)_n) = \alpha(\psi[(a_n)_n])$, where $\psi[(a_n)_n](u) := a_{|u|}$. On $(\Sigma^*)^\omega$ we introduce the Baire-metric

$$d((a_n)_n, (b_n)_n) := 2^{-\min\{n | a_n \neq b_n\}}$$

(with the convention that $\min\emptyset = \infty$ and $2^{-\infty} = 0$). Note that α' is continuous with respect to this metric and that the limit of a convergent sequence in $\operatorname{dom}\alpha'$ corresponds to a Cauchy-sequence in X. Since X is complete, $\operatorname{dom}\alpha'$ is closed. Now, since every $x \in X$ has an α-name of size $M(n)$, we have

$$X = \alpha'\left(\prod_{n\in\mathbb{N}}\{u \in \Sigma^* \mid |u| \leq M(n)\} \cap \operatorname{dom}\alpha'\right).$$

Now, the set $\{u \in \Sigma^* \mid |u| \leq M(n)\}$ is finite and thus compact for every $n \in \mathbb{N}$, so the product $\prod_{n\in\mathbb{N}}\{u \in \Sigma^* \mid |u| \leq M(n)\}$ is compact by Tychonoff's Theorem and since $\operatorname{dom}\alpha'$ is closed, $\prod_{n\in\mathbb{N}}\{u \in \Sigma^* \mid |u| \leq M(n)\} \cap \operatorname{dom}\alpha'$ is compact. So, X is compact, since α' is continuous.

(ii) ⇒ (i). Let $A \subseteq X$ be countable and dense in X. Since X is compact, for every n there exist $s_1^n, \dots, s_{k_n}^n \in A$ such that

$$X \subseteq \bigcup_{i=1}^{k_n} B(s_i^n, 2^{-n}).$$

There exists a representation $\sigma\colon \subseteq \Sigma^* \to A$ of A such that for every n and $k \leq k_n$ the element s_k^n of A has a σ-name of size at most $\sum_{i=1}^n \log k_i$. Let α be the Cauchy representation of X induced by σ. For every $x \in X$ there exists a sequence $(s_{i_n}^n)_n$ such that $x \in B(s_{i_n}^n, 2^{-n})$, which yields an α-name of x with size bounded by $\sum_{i=1}^n \log k_i$. □

Remark 3.10. One may argue that the term 'compact' is not entirely appropriate, since the completeness of the represented space is not a priori necessary to formulate the definition. It is easy to see that one can find representations of non-compact spaces (e.g. $\rho|^{\mathbb{D}\cap[0,1]}$) such that the size of every name is uniformly bounded. It might thus be more natural to drop the condition on the completeness of the represented space and refer to such representations as **precompact** (or 'totally bounded') rather than compact. One can then show that a separable metric space X has a precompact representation if and only if it is precompact, since from a precompact representation of X one can easily construct a compact representation of its Cauchy-Completion $\tilde{X}$ (which by the above proposition entails that the Cauchy-Completion is compact, i.e. X is precompact). A 'compact' representation could then be defined either like it is done here, by restricting the co-domain, or - maybe more naturally - by requiring the representation's domain to be closed with respect to a Baire-metric on $\mathfrak{Reg}$, similar to the one in the above proof. However, since we are only going to study complete spaces, we may content ourselves with the definition given here.

From now on, we consider the function

$$\mathfrak{F}\colon\ X \times A \to Y$$

where X is some metric space of cardinality at most $2^{\aleph_0}$, A is a separable and compact metric space, which will be referred to as the **parameter space** represented by an efficiently compact Cauchy representation γ, induced from the notation σ of $S \subseteq A$ and Y is a separable metric space represented by a Cauchy representation β induced by a notation ν of $T \subseteq Y$, called the **image space**.

There are also some noteworthy relationships between first- and second order complexity classes. Let $\mathbf{DTIME}(T(n))$ be the class of regular functions computable in time $\mathcal{O}(T(n))$, $\mathbf{SPACE}(S(n))$ be the class of regular functions computable in space $\mathcal{O}(S(n))$

and $\mathbf{SIZE}(M(n))$ be the class of regular functions with size in $\mathcal{O}(M(n))$ (for more detailed definitions of **DTIME** and **SPACE**, see e.g. [AB09]). We may then define the following complexity classes over $\mathfrak{Reg}$.

Definition 3.11. Let

- $\mathbf{FP} = \bigcup_{c\in\mathbb{N}} \mathbf{DTIME}(n^c)$.
- $\mathbf{PSIZE} = \bigcup_{c\in\mathbb{N}} \mathbf{SIZE}(n^c)$.
- $\mathbf{FPSPACE} = \bigcup_{c\in\mathbb{N}} \mathbf{SPACE}(n^c)$.

Definition 3.12. (i) Let $\mathbf{C}$ be any of the complexity classes listed in definition 3.11. The set of $\mathbf{C}$-points of X with respect to α, denoted $\mathbf{C}_\alpha(X)$ is the set of all $x \in X$ such that there exists a $\psi \in \mathfrak{Reg} \cap \mathbf{C}$ with $\alpha(\psi) = x$.

(ii) The set $\mathbf{FP}^2$-points of $\mathfrak{F}$ with respect to β and parameters represented by γ, denoted $(\mathbf{FP}^2)^\beta_{\mathfrak{F},\gamma}(X)$ is the set of all $x \in X$ such that the function

$$A \ni a \mapsto \mathfrak{F}(x, a)$$

is (γ, β)-$\mathbf{FP}^2$.

Example 3.13. On the space $C([0,1])$ of continuous functions, consider the functional

$$\mathbf{EVAL}\colon\ C([0,1]) \times [0,1] \to \mathbb{R},\ \ (f, x) \mapsto f(x).$$

One can easily verify that a function $f \in C([0,1])$ is in $\left(\mathbf{FP}^2\right)^\rho_{\mathbf{EVAL},\rho_c|^{[0,1]}}(C([0,1]))$ if and only if it is computable in polynomial-time in the sense of Ko and Friedman (Definition 1.1). Since $\rho \equiv \rho_c$ (Proposition 3.8), it follows that $f \in \left(\mathbf{FP}^2\right)^\rho_{\mathbf{EVAL},\rho|^{[0,1]}}(C([0,1]))$ if and only if it is polynomial-time computable in the sense of Ko and Friedman.

Proposition 3.14. *Let* $\mathbf{C}$ *be any of the complexity classes listed in Definition 3.11.*

(i) If $f\colon \mathfrak{Reg} \to \mathfrak{Reg}$ *is second-order polynomial time computable and* $\psi \in \mathbf{C}$ *then* $f(\psi) \in \mathbf{C}$.

(ii) Let $F\colon X \to Y$, α *be a representation of* X *and* β *be a representation of* Y. *If* F *is* (α, β)-$\mathbf{FP}^2$, *then* $F(\mathbf{C}_\alpha(X)) \subseteq \mathbf{C}_\beta(Y)$.

(iii) Let α, γ *be representations of the same set* X. *If* $\alpha \preceq \gamma$ *then* $\mathbf{C}_\alpha(X) \subseteq \mathbf{C}_\gamma(X)$.

(iv) Let α *be a representation of* X. *If* $\mathfrak{F}$ *is* $(\alpha\times\gamma, \beta)$-$\mathbf{FP}^2$, *then* $\mathbf{FP}_\alpha(X) \subseteq \left(\mathbf{FP}^2\right)^\beta_{\mathfrak{F},\gamma}(X)$.

Definition 3.15. Let α be a representation of X. We say that α **non-uniformly characterises $\mathfrak{F}$ with respect to β with parameters represented by** γ, if

$$\mathbf{FP}_\alpha(X) = (\mathbf{FP}^2)^\beta_{\mathfrak{F},\gamma}(X).$$

The notion of 'non-uniform characterisation' is motivated by the following observation.

Example 3.16. Let α be representation of $C([0,1])$. If α non-uniformly characterises **EVAL** with respect to ρ with parameters represented by $\rho|^{[0,1]}$ then a function $f \in C([0,1])$ is polynomial-time computable in the sense of Ko and Friedman if and only if it is an **FP**-point of α, i.e. if and only if it has a polynomial-time computable α-name.

Definition 3.17 (modulus of continuity)**.** Let X and Y be metric spaces and $f\colon\ X \to Y$ be a uniformly continuous function. A function $\mu\colon\ \mathbb{N} \to \mathbb{N}$ is called a **(uniform) modulus of continuity** of f if for all $x, y \in X$ we have

$$d_X(x,y) \le 2^{-\mu(n)} \Rightarrow d_Y(f(x), f(y)) \le 2^{-n}$$

We now have introduced the terminology necessary to state our main result.

Theorem 3.18. *If for every $x \in X$ the function $\mathfrak{F}(x,\cdot)\colon\ A \to Y,\ a \mapsto \mathfrak{F}(x,a)$ is continuous, then there exists a canonical representation α of $X/\sim_{\mathfrak{F}}$, uniformly characterising $\bar{\mathfrak{F}}$ with respect to β and parameters represented by γ, which is defined as follows: regular ψ is an α-name of $\bar{x} \in X/\sim_{\mathfrak{F}}$ if for every $a \in S$ with σ-name s, $\hat{\psi}(\bar{s}\#1^n)$ (cf. Definition 2.6) is a ν-name of some $y \in T$ satisfying $d_Y(\mathfrak{F}(x,a), y) \le 2^{-n}$ and $\hat{\psi}(1^n) = \mu(n)$ where μ is a modulus of continuity of $\mathfrak{F}(x,\cdot)$.*

Moreover, the mapping

$$(X/\sim_{\mathfrak{F}}) \times A \ni (\bar{x}, a) \mapsto \mathfrak{F}(x,a) \in Y$$

is $(\alpha \times \gamma, \beta)$ computable in time $\mathcal{O}(n + \mu(n+1))$.

Proof. It is easy to see that the relation given is functional and surjective. We now show, that $\bar{\mathfrak{F}}$ is $(\alpha \times \gamma, \beta)$-$\mathbf{FP}^2$. Given an $\alpha \times \gamma$ name of $(\bar{x}, a) \in (X/\sim_{\mathfrak{F}}) \times A$ and $1^n, n \in \mathbb{N}$ we need to compute a ν-name of $y \in T$ satisfying $d_Y(\mathfrak{F}(x,a), y) \le 2^{-n}$. In order to do so, we first query the oracle for $\mu(n+1)$ and then for an approximation ξ to a of order $2^{-\mu(n+1)}$. Finally, we query the oracle for an approximation to $\mathfrak{F}(x,\xi)$ up to error 2^{-n-1} and return it. It remains to show that α uniformly characterises $\bar{\mathfrak{F}}$. Let δ be some representation of $X/\sim_{\mathfrak{F}}$ such that $\bar{\mathfrak{F}}$ is $(\delta \times \gamma, \beta)$-computable in second order polynomial time, and let $P(x, \ell)$ be a second-order bound on the running time. Given 1^n, $n \in \mathbb{N}$, some δ-name ψ of $\bar{x} \in X/\sim_{\mathfrak{F}}$ and some γ-name ξ of $a \in A$, the machine computes a ν-name of $y \in T$ satisfying $d_Y(\mathfrak{F}(x,a), y) \le 2^{-n}$ in time $P(n, |\psi \times \xi|)$. Now, since every $a \in A$ has a γ-name with length majorised by some function $M(n)$, the running time of the machine on input of such a name is bounded by $P(n, \max\{|\psi|, M\})$. From this it follows, that $P(\cdot, \max\{|\psi|, M\})$ is a modulus of continuity for $\mathfrak{F}(x,\cdot)$, since in order to compute the value of $\mathfrak{F}(x,a)$ up to error 2^{-n}, the machine only reads an approximation to a up to error $P(n, \max\{|\psi|, M\})$. Observe that $(\psi, n) \mapsto P(n, \max\{|\psi|, M\})$ is computable in second-order polynomial time. It is obvious, that given a δ-name of $x \in X$, a σ-name of $s \in S$ and 1^n, $n \in \mathbb{N}$, we can compute a ν-name of $y \in T$ satisfying $d_Y(\mathfrak{F}(x,s), y) \le 2^{-n}$ in second-order polynomial time. □

Remark 3.19.
(i) Theorem 3.18 is essentially a uniform and somewhat generalised version of a classic characterisation of polynomial-time computable functions (cf. [Ko91], Corollary 2.21). A similar fact has also been stated in [KC10] and proven in [Kaw11].
(ii) One can generalise Theorem 3.18 to the case where A is not necessarily compact, if one requires the representation of A to be locally efficiently compact (this is for instance the case for ρ) and if there exists some covering of A by compact sets such that one can decide in second-order polynomial time to which compact set a given element of A belongs (this is for instance the case for the covering $\{[-n,n] \mid n \in \mathbb{N}\}$ of $\mathbb{R}$ represented by ρ). Instead of a global modulus of continuity, the characterising

representation will then encode a local modulus of continuity for every compact set of this covering.

(iii) Theorem 3.18 suggests a canonical way of computing a $(\delta \times \gamma, \beta)$-realiser of $\mathfrak{F}$, where δ is some representation of X. One first computes a $(\delta, \delta/\sim_{\mathfrak{F}})$-realiser of the canonical projection $\pi\colon X \to X/\sim_{\mathfrak{F}}$ and then computes a realiser of the reduction $(\delta/\sim_{\mathfrak{F}}) \preceq \alpha$, where α is the uniformly characterising representation of $\bar{\mathfrak{F}}$ according to Theorem 3.18. Now, the given α-name allows for evaluating $\mathfrak{F}$ in second-order polynomial time. More explicitly, given a δ-name ψ of $x \in X$, in order to compute $\mathfrak{F}(x,\cdot)$ we need to be able to compute from ψ a modulus of continuity of $\mathfrak{F}(x,\cdot)$ and evaluations of $\mathfrak{F}(x,\cdot)$ on 'rational sample points'.

Theorem 3.20. *If α uniformly characterises $\mathfrak{F}$ then α non-uniformly characterises $\mathfrak{F}$.*

Proof. Suppose that α uniformly characterises $\mathfrak{F}$ and let $x_0 \in (\mathbf{FP}^2)^{\beta}_{\mathfrak{F},\gamma}(X)$. We define a new representation α' of X as follows: regular ψ is an α'-name of $x \in X$ if $\hat{\psi}(0) = 1$ and $\hat{\psi}(1\cdot)$ is an α-name of x or if $x = x_0$ and $\hat{\psi}(0) = 0$ and $\hat{\psi}(1\cdot)$ is a δ-name of $\bar{x} \in X/\sim_{\mathfrak{F}}$, where δ is the uniformly characterising representation of $\bar{\mathfrak{F}}$ according to Theorem 3.18. It is clear that $\mathfrak{F}$ is $(\alpha' \times \gamma, \beta)$-$\mathbf{FP}^2$. By the characterising property of α it follows that $\alpha \equiv \alpha'$, so by Proposition 2.11 we have $\mathbf{FP}_\alpha(X) = \mathbf{FP}_{\alpha'}(X)$. We now show that $x_0 \in \mathbf{FP}_{\alpha'}(X)$. Since $\mathfrak{F}(x_0,\cdot)$ is (γ,β)-computable in second-order-polynomial time, $\mathfrak{F}(x_0,\cdot)|_S$ is (σ,β)-computable in polynomial time and the (polynomial) bound on the running time of some (γ,β)-realiser of $\mathfrak{F}(x_0,\cdot)$ constitutes a modulus of continuity of $\mathfrak{F}$. This means that some δ-name of $\bar{x}_0$ is in $\mathbf{FP}$, i.e. $x_0 \in \mathbf{FP}_{\alpha'}(X)$ and hence $x_0 \in \mathbf{FP}_\alpha(X)$. It follows that α nonuniformly characterises $\mathfrak{F}$. $\square$

4. Computing with Discrete Objects

4.1. **Algorithms on Polynomials and Rational Functions.** In this section we study the computational complexity of well-known algorithms on polynomials and rational functions with dyadic rational coefficients, which will serve as a basis for the algorithms on real functions developed in Sections 5 and 6.

Proposition 4.1. *(i) There exists an algorithm, which on input of a $\nu_{\mathbb{D}} \times \nu_{\mathbb{D}}$ name of $(a, b) \in \mathbb{D} \times \mathbb{D}$ outputs a $\nu_{\mathbb{D}}$-name of $a + b \in \mathbb{D}$ of size at most* $\max\{\ell(a), \ell(b)\} + 1$ *in time $\mathcal{O}(\max\{\ell(a), \ell(b)\})$.*

(ii) There exists an algorithm, which on input of a $\nu_{\mathbb{D}} \times \nu_{\mathbb{D}}$ name of $(a, b) \in \mathbb{D} \times \mathbb{D}$ outputs a $\nu_{\mathbb{D}}$ name of $a \cdot b \in \mathbb{D}$ of size at most $\ell(a) + \ell(b)$ in time $O(\ell(a)\ell(b))$.

(iii) There exists an algorithm, which on input of a $\nu_{\mathbb{D}} \times \nu_{\mathbb{D}}$ name of $(a, b) \in \mathbb{D} \times \mathbb{D}$ and 1^n, $n \in \mathbb{N}$, outputs a $\nu_{\mathbb{D}}$ name of some number $c \in \mathbb{D}$ of size $n + \max\{\lfloor \log(a) \rfloor - \lfloor \log(b) \rfloor, 0\}$ *satisfying $|c - \frac{a}{b}| \leq 2^{-n}$ in time $\mathcal{O}((\ell(a) + n)\ell(b))$.*

(iv) There exists an algorithm, which on input of a $\nu_{\mathbb{D}} \times \nu_{\mathbb{D}}$ name of $(a, b) \in \mathbb{D} \times \mathbb{D}$ with $\frac{a}{b} \in \mathbb{D}$, outputs a $\nu_{\mathbb{D}}$ name of $\frac{a}{b}$ of size $\max\{\lfloor \log(a) \rfloor - \lfloor \log(b) \rfloor, 0\}$ *in time $\mathcal{O}(\ell(a)\ell(b))$.*

Remark 4.2. The complexity of Multiplication given in Proposition 4.1 is the complexity underlying the 'naive' multiplication-algorithm and not the asymptotically best result known (see e.g. [Fü09] and [SS71]). Similarly, there exist generic division algorithms whose complexity is majorised by the complexity of multiplication. This will systematically increase the asymptotic time complexity of virtually all the following algorithms, since almost all of them rely on multiplication.

We denote by $\mathbb{D}[x]$ the ring of univariate polynomials over $\mathbb{D}$ and by $\mathbb{D}_m(x)$ the ring of rational functions over $\mathbb{D}$ which are bounded on the compact interval $[0, 1]$.

Definition 4.3. (i) A $\nu_{\mathbb{D}[x]}$-name of $P \in \mathbb{D}[x]$ is a $\nu_{\mathbb{D}}^*$ name of the list of coefficients of P in the monomial basis.

(ii) A $\nu_{\mathbb{D}_m(x)}$ name of a rational function $\frac{P}{Q} \in \mathbb{D}_m(x)$ with $\min_{x \in [0,1]} |Q(x)| \geq 1$ is a $\mathbb{D}[x] \times \mathbb{D}[x]$-name of its numerator and denominator.

Remark 4.4. Requiring the represented function in (ii) to be minorised by 1 is equivalent to encoding a rational function bounded on $[0, 1]$ by a similar name and a lower bound (in binary) on its minimum.

As $\nu_{\mathbb{D}[x]}$ and $\nu_{\mathbb{D}_m(x)}$ are product notations, the size of a name of a function can be expressed in terms of the size of the individual components.

Definition 4.5. Let $P := \sum_{k=0}^{\deg P} a_k x^k \in \mathbb{D}[x]$, $R = \frac{P}{Q} \in \mathbb{D}_m(x)$ be given as $\nu_{\mathbb{D}[x]}$- and $\nu_{\mathbb{D}_m(x)}$-names respectively. Define $d_P := \deg P$, $d_R := \max\{d_P, d_Q\}$, $c_P := \max_{k=0,\dots,d_P}\{\lfloor \log(|a_k| + 1) \rfloor\}$, $c_R := \max\{c_P, c_Q\}$, $b_P := \max_{k=0,\dots,d_P}\{\ell(a_k)\}$ and $b_R := \max\{b_P, b_Q\}$.

Note that this is an abuse of notation, since b_P and b_R actually depend on the chosen representation. For $P = \sum_{k=0}^{d_P} a_k x^k$, the number c_P is a bound on the logarithm of the *height* $H(P) = \max_{k \leq d_P}\{|a_k|\}$ of P and $b_P \geq c_P$. The size $\ell(P)$ of P is bounded by $d_P b_P$. Analogously, the size of a rational function R is bounded by $2 d_R b_R$.

Proposition 4.6. *There exists an algorithm, which, given a $\nu_{\mathbb{D}[X]}$-name of $P \in \mathbb{D}[x]$ and a $\nu_{\mathbb{D}}$-name of $\xi \in \mathbb{D}$ outputs a $\nu_{\mathbb{D}}$-name of $P(\xi) \in \mathbb{D}$ in time $\mathcal{O}((b_P + d_P\ell(\xi))\ell(\xi)d_P)$ with output size at most $b_P + d_P(\ell(\xi) + 1)$.*

Proof. Let $a_0, \dots a_{d_P}$ denote the coefficients of P. Put $E_0 := a_{d_P}$ and compute the result iteratively using the Horner schema $E_{k+1} := \xi E_k + a_{d_P-(k+1)}$. Output E_{d_P}. By induction one verifies $\ell(E_k) \leq b_P + k(\ell(\xi) + 1)$, so in the k^{th} iterative step the algorithm performs one multiplication of a number of size $\ell(\xi)$ and a number of size $b_P + k(\ell(\xi)+1)$ and one addition of a number of size b_P and a number of size $b_P + k(\ell(\xi) + 2)$. The complexity is thus dominated by the complexity of d_P multiplications of numbers of size at most $b_P + d_P(\ell(\xi) + 1)$ with numbers of size at most $\ell(\xi)$. $\square$

Proposition 4.7. *There exists an algorithm, which, given a $\nu_{\mathbb{D}[x]}$-name of $P \in \mathbb{D}[x]$ and a $\nu_{\mathbb{D}}$-name of $c \in \mathbb{D}$ outputs a $\nu_{\mathbb{D}[x]}$-name of $P(x-c) \in \mathbb{D}[x]$ in time $\mathcal{O}(d_P^2 b_P \ell(c) + d_P^3 \ell(c)^2)$ and the size of the coefficients of the output polynomial is bounded by $b_P + d_P(\ell(c) + 1)$.*

Proof. Let $a_0, \dots a_{d_P}$ denote the coefficients of P. Put $T_0 := a_{d_P}$ and $T_{k+1} := (x - c)T_k + a_{d_P-k}$. One verifies by induction that the size of the coefficients of T_k is bounded by $b_P + k(\ell(c) + 1)$, so in the k^{th} iterative step we perform k multiplications of numbers of size $\ell(c)$ and $b_P + k(\ell(c) + 1)$ and k additions of numbers of size $b_P + k(\ell(c) + 2)$, so the complexity is bounded by the complexity of $\frac{d_P(d_P+1)}{2}$ multiplications of numbers of size at most $b_P + d_P(\ell(c) + 1)$ with numbers of size $\ell(c)$. $\square$

Proposition 4.8. *Evaluation of $R := \frac{P}{Q} \in \mathbb{D}_m(x)$ in $\xi \in \mathbb{D}$ is $(\nu_{\mathbb{D}_m(x)} \times \nu_{\mathbb{D}} \times \nu_1, \nu_{\mathbb{D}})$-computable in time*

$$\mathcal{O}\left(d_P(b_P + d_P\ell(\xi))^2 + d_Q(b_Q + d_Q\ell(\xi))^2 + n(b_Q + d_Q\ell(\xi))\right).$$

In particular, it is computable in time

$$\mathcal{O}\left(d_R(b_R + d_R\ell(\xi))^2 + n(b_R + d_R\ell(\xi))\right)$$

Proof. Proposition 4.6 allows us to compute $P(\xi)$ and $Q(\xi)$ in time $\mathcal{O}(d_P(b_P + d_P(\ell(\xi) + 1))^2 + d_Q(b_Q + d_Q(\ell(\xi) + 1))^2)$ with output sizes $b_P + d_P(\ell(\xi) + 1)$ and $b_Q + d_Q(\ell(\xi) + 1)$ respectively. Now, we apply Proposition 4.1, which yields the desired output in time $\mathcal{O}((b_P + d_P(\ell(\xi) + 1) + n)(b_Q + d_Q(\ell(\xi) + 1)))$ $\square$

Proposition 4.9. *(i) There exists an algorithm, which, given a $\nu^2_{\mathbb{D}[X]}$-name of $(P, Q) \in \mathbb{D}[x]^2$ outputs a $\nu_{\mathbb{D}[x]}$-name of $P + Q \in \mathbb{D}[x]$ in time $\mathcal{O}((d_P + d_Q)(b_P + b_Q))$. The output polynomial has degree at most $\max\{d_P, d_Q\}$ and the size of its coefficients is bounded by $\max\{b_P, b_Q\} + 1$.*

(ii) There exists an algorithm, which, given a $\nu^2_{\mathbb{D}[X]}$-name of $(P, Q) \in \mathbb{D}[X]^2$ outputs a $\nu_{\mathbb{D}[x]}$-name of $P \cdot Q \in \mathbb{D}[x]$ in time $\mathcal{O}(d_P d_Q b_P b_Q)$. The output polynomial has degree at most $d_P + d_Q$ and the size of its coefficients is bounded by $b_P + b_Q + \min\{d_P, d_Q\}$.

Proof. (i) obvious.

(ii) Let $P := \sum_{k=0}^{d_P} a_k x^k$, $Q := \sum_{k=0}^{d_Q} b_k x^k$. Then

$$P \cdot Q = \sum_{k=0}^{d_P+d_Q} \left(\sum_{i+j=k} a_i b_j \right) x^k.$$

The inner sum has at most $\min\{d_P, d_Q\}$ summands, so in order to compute the k^{th} coefficient, we need to perform at most $\min\{d_P, d_Q\}$ multiplications of numbers of size b_P with numbers of size b_Q and $\min\{d_P, d_Q\}$ additions. The given bound on the coefficients then follows from 4.1 and the complexity estimate is $\mathcal{O}((d_P + d_Q)\min\{d_P, d_Q\}b_P b_Q) = \mathcal{O}(d_P d_Q b_P b_Q)$.

□

Proposition 4.10. *Given a $\nu_{\mathbb{D}[x]} \times \nu_{\mathbb{D}[x]}$ name of $(P, Q) \in \mathbb{D}[x] \times \mathbb{D}[x]$ we can compute a $\nu_{\mathbb{D}[x]}$-name of $P \circ Q$ in time $\mathcal{O}(d_P^3 d_Q^2(d_P b_Q + d_P d_Q + b_P)^2)$. The output-polynomial has degree at most $d_P d_Q$ and its coefficients have size $d_P b_Q + d_P(d_Q + 1) + b_P$.*

Proof. Let $P = \sum_{k=0}^{d_P} a_k x_k$. Define $C_0 := a_{d_P}$ and for $0 \le k < d_P$, $C_{k+1} := QC_k + a_{d_P-(k+1)}$. By induction one verifies that the polynomial C_k has degree at most kd_Q and coefficient size $kb_Q + kd_Q + b_P + k$, so the total complexity is dominated by d_P multiplications of polynomials of degree smaller than $d_P d_Q$ and coefficients of size $d_P b_Q + d_P(d_Q + 1) + b_P$. The bound given then follows from Proposition 4.9. □

Proposition 4.11. *(i) There exists an algorithm, which, given a $\nu_{\mathbb{D}_m(x)} \times \nu_{\mathbb{D}_m(x)}$-name of $(R, S) \in \mathbb{D}_m(x) \times \mathbb{D}_m(x)$ outputs a $\nu_{\mathbb{D}_m(x)}$-name of $R+S \in \mathbb{D}_m(x)$ in time $\mathcal{O}(d_R d_S b_R b_S)$. The output function has degree at most $d_R + d_S$ and the size of its coefficients is bounded by $b_R + b_S + \min\{d_R, d_S\} + 1$.*

(ii) There exists an algorithm, which, given a $\nu_{\mathbb{D}_m(x)} \times \nu_{\mathbb{D}_m(x)}$-name of $(R, S) \in \mathbb{D}_m(x) \times \mathbb{D}_m(x)$ outputs a $\nu_{\mathbb{D}_m(x)}$-name of $R \cdot S \in \mathbb{D}_m(x)$ in time $\mathcal{O}(d_R d_S b_R b_S)$. The output function has degree at most $d_R + d_S$ and the size of its coefficients is bounded by $b_R + b_S + \min\{d_R, d_S\}$.

Proposition 4.12. *Given a $\nu_{\mathbb{D}_m(x)} \times \nu_{\mathbb{D}_m(x)}$ name of $(R, S) \in \mathbb{D}_m(x) \times \mathbb{D}_m(x)$ with $S([0, 1]) \subseteq [0, 1]$ we can compute a $\nu_{\mathbb{D}_m(x)}$-name of $R \circ S$ in time $\mathcal{O}(d_R^3 d_S^2(d_R b_S + d_R d_S + b_R)^2)$. The output-function has degree at most $2d_R d_S$ and its coefficients have size $2d_P b_Q + 2d_P(d_Q + 1) + 2b_P$.*

Proof. Let $R := P/Q$ with $P := \sum_{k=0}^{d_P} a_k x^k$, $Q := \sum_{k=0}^{d_Q} b_k x^k$. We compute $P(S)$ and $Q(S)$ using a schema like in 4.10. Let $C_0 := a_{d_P}$ and $C_{k+1} := SC_k + a_{d_P-(k+1)}$. By induction one verifies that C_k has degree at most kd_S and the size of its coefficients is bounded by $kb_S + kd_S + b_R + k$, so the total complexity of this computation is $\mathcal{O}(d_R^2 d_S^2 b_S(d_R b_S + d_R d_S + b_R))$ by 4.11. Similarly, we obtain $Q(S)$ in the same running time. Now, we may output $P(S)/Q(S)$ (note that $Q(S) \ge 1$ throughout $[0, 1]$), which can be obtained according to 4.11 in time $\mathcal{O}(d_R^2 d_S^2(d_R b_S + d_R d_S + b_R)^2)$. The output size follows analogously. □

Proposition 4.13. *There exists an algorithm, which, given a $\nu_{\mathbb{D}[x]}$ name of some polynomial $P \in \mathbb{D}[x]$ outputs a $\nu_{\mathbb{D}[x]}$-name of its derivative $P'(x) \in \mathbb{D}[x]$ in time $\mathcal{O}(d_P b_P \log d_P)$. The output polynomial has degree $d_P - 1$ and coefficient size at most $b_P + \log d_P$. More generally, given a $\nu_{\mathbb{D}[x]} \times \nu_1$ of $P \in \mathbb{D}[x]$ and $m \in \mathbb{N}$ we can compute a $\nu_{\mathbb{D}[x]}$-name of $P^{(m)}$ in time $\mathcal{O}(d_P m^2(\log d_P)^2 + md_P \log d_P b_P)$*

Proof. We perform $m - 1$ multiplications of numbers of size at most $\log d_P$, which takes time $\mathcal{O}(m^2(\log d_P)^2)$. Then we multiply a number of size $m \log d_P$ with the coefficients, which takes time $\mathcal{O}(m \log d_P b_P)$. Since there are $\mathcal{O}(d_P)$ such multiplications to perform the total complexity is $\mathcal{O}(d_P m^2 (\log d_P)^2 + m d_P \log d_P b_P)$. □

Theorem 4.14. *(i) There exists an algorithm, which, given a $\nu_{\mathbb{D}[x]}$-name of $P \in \mathbb{D}[x]$ and 1^n, $n \in \mathbb{N}$ outputs a $\nu_{\mathbb{D}[x]}$-name of some polynomial $Q \in \mathbb{D}[x]$ whose coefficients approximate the coefficients of an antiderivative of P up to error 2^{-n} in time $\mathcal{O}(d_P(b_P + n) \log d_P)$.*

(ii) There exists an algorithm, which, given a $\nu_{\mathbb{D}[x]}$-name of $P \in \mathbb{D}[x]$, 1^m, $m \in \mathbb{N}$ and 1^n, $n \in \mathbb{N}$ outputs a $\nu_{\mathbb{D}[x]}$-name of some polynomial $Q \in \mathbb{D}[x]$ whose coefficients approximate the coefficients of an m-fold anti-derivative of P, i.e. a polynomial $R \in \mathbb{D}[x]$ satisfying $R^{(m)} = P$, up to error 2^{-n} in time $\mathcal{O}(d_P m^2 (\log(d_P + m))^2 + d_P(b_P + n) m \log(d_P + m))$.

(iii) There exists an algorithm, which, given a $\nu_{\mathbb{D}[x]}$-name of $P \in \mathbb{D}[x]$, a $\nu^2_{\mathbb{D}[x]}$-name of $(a, b) \in \mathbb{D}^2 \cap [0, 1]$ and 1^n, $n \in \mathbb{N}$ outputs a $\nu_{\mathbb{D}}$ name of some $c \in \mathbb{D}$ satisfying

$$|c - \int_a^b P(x)dx| \leq 2^{-n}$$

in time

$$\mathcal{O}(d_P(b_P + n + d_P(\ell(a) + \ell(b)))(\ell(a) + \ell(b)) + d_P \log d_P (b_P + \log d_P + n)),$$

with output size bounded by $b_P + n + d_P(\max\{\ell(a), \ell(b)\} + 1) + 1$.

Proof. (i) Let $a_0, \ldots, a_d$ denote the coefficients of P. An antiderivative of P is given by

$$\sum_{k=1}^{d+1} \frac{a_{k-1}}{k} x^k.$$

The coefficients of this polynomial can be approximated up to error 2^{-n} following Proposition 4.1 in time $\mathcal{O}(d_P(b_P + n) \log d_P)$.

(ii) Picking up the notation in (i), an m-fold anti-derivative is given by

$$\sum_{k=m}^{d+m} \frac{a_{k-m}}{k(k-1) \cdot \cdots \cdot (k - m + 1)} x^k.$$

Its coefficients can be approximated up to error 2^{-n} in time $\mathcal{O}(d_P m^2 (\log(d_P + m))^2 + d_P(b_P + n) m \log(d_P + m))$.

(iii) Using item (i) we compute an approximate antiderivative Q of P up to error 2^{-n-1} in time $\mathcal{O}(d_P \log d_P (b_P + \log d_P + n))$. Note that for all $x \in [0, 1]$ we have

$$|Q(x) - \int_0^x P(x)dx| \leq 2^{-n-1}.$$

So $Q(b) - Q(a)$ approximates $\int_a^b P(x)dx$ up to error 2^{-n}. We can compute this quantity using 4.6 and 4.1 in time $\mathcal{O}(d_P(b_P + n + d_P(\ell(a) + \ell(b)))(\ell(a) + \ell(b))$ and the result has size $b_P + n + d_P(\max\{\ell(a), \ell(b)\} + 1) + 1$.

□

Proposition 4.15. *Given a $\nu_{\mathbb{D}[X]}$ name of $P \in \mathbb{D}[x]$ we can compute a ν_1-name of the logarithm of some Lipschitz-constant of P in time $\mathcal{O}(d_P c_P)$ with output size $2 \log(d_P + 1) + c_P$*

Proof. A Lipschitz-constant of P is given by $\max_{x\in[0,1]}|P'(x)|$, which is majorised by $\frac{d_P(d_P+1)}{2}2^{c_P}$, so its logarithm is majorised by $2\log(d_P+1)+c_P$, where c_P can be computed in time $\mathcal{O}(d_P c_P)$ by reading the coefficients of P and determining the maximum length. □

Proposition 4.16. *Given a $\nu_{\mathbb{D}_m(x)}$-name of $R := \frac{P}{Q} \in \mathbb{D}_m(x)$ we can compute a ν_1 name of the logarithm of some Lipschitz-constant of R in time $\mathcal{O}(d_R c_R)$ with output size $3\log(d_R+1)+2c_R+1$.*

Proof. A Lipschitz-constant of R is given by

$$\max_{x\in[0,1]}\left|\frac{P(x)Q'(x)-P'(x)Q(x)}{Q^2(x)}\right|.$$

Its logarithm is majorised by $2c_R+3\log(d_R+1)+1$, where c_R can be computed in time $\mathcal{O}(d_R c_R)$. □

4.2. **Root Finding and Maximisation.** In the following we develop the techniques necessary for finding the maximum of a function. The algorithm will rely on finding and comparing the local extrema of an approximating polynomial, thus reducing the problem to the problem of *root finding*. Our root finding algorithm will be based on a root isolating algorithm found in [BPR08], which there is credited to [Vin36] and [Usp48] - in fact, our algorithm is merely a variant of Algorithm 10.5 in [BPR08], slightly adapted to better match our framework (in particular, it will be formulated over the domain $\mathbb{D}$ rather than over $\mathbb{Z}$). The root isolating algorithm we are going to use works only for polynomials which do not have multiple roots, so given a polynomial we will first need to extract its *separable part*, which can be done efficiently using *signed subresultant sequences*. This is carried out in detail for the ring $\mathbb{Z}$ in [BPR08] and the results immediately carry over to the ring $\mathbb{D}$. It should be re-emphasised that this whole section follows the presentation in [BPR08] *very closely* and all adaptations made in order to fit our framework are *very minor*.

Definition 4.17. Let $P \in \mathbb{C}[x]$ be some polynomial. The **separable part** $\bar{P}$ of P is a separable polynomial whose complex roots coincide with the roots of P in $\mathbb{C}$.

Remark 4.18. Actually, 'the' separable part is only defined up to a multiplicative constant.

Proposition 4.19. *There exists an algorithm, which on input $P \in \mathbb{D}[x]$ outputs the separable part of P in time $\mathcal{O}(d_P^3(b+\log d_P))$. The coefficients of the output polynomial have size at most $d_P + b_p + \log(d_P+1)$.*

Proof. As mentioned before, this follows from the proof of the correctness of Algorithm 10.1 in [BPR08]. □

We may now proceed to developing an algorithm, which on input of a separable polynomial P outputs a list isolating the roots of P. It is based on **Descartes' Rule of Signs**. Let $\mathrm{Var}(P)$ denote the number of sign variations in the list of the polynomial's coefficients, e.g. $\mathrm{Var}(x^2-x+1)=2$ and $\mathrm{Var}(x^2-x-1)=1$. Let *pos*(P) denote the number of positive real roots of P, counted with multiplicity.

Theorem 4.20 (Descartes' Rule of Signs)**.** *Let P be a polynomial in $\mathbb{R}[x]$. Then $Var(P) \geq pos(P)$ and$Var(P) - pos(P)$ is even.*

Proof. Without loss of generality we may assume that $P(x) = x^d + a_d x^{d-1} + \cdots + a_0$ with $a_0 \neq 0$. Observe that if we change the sign of a coefficient a_i with $1 \leq i \leq d-1$, then the number of sign variations changes by either zero or two. Since the sign of the leading coefficient is one, the number of sign variations is odd if and only if $a_0 < 0$. The sign of P at zero is the sign of a_0 and the sign of P at $+\infty$ is one. From this it follows immediately, that if P has only simple positive roots, the number of positive roots is odd if and only if $a_0 < 0$, i.e. the number of sign variations and the number of positive roots coincide modulo two. Now, if P has multiple positive roots, P is obtained from a polynomial with simple positive roots by successive multiplication of terms of the form $(x-a)$, where a is positive. Observe that each of these multiplications alters the sign of the constant coefficient and thus alters the number of sign variations modulo two. It remains to show that the number of positive roots is always smaller than the number of sign variations. For this observe, that multiplying a polynomial with $(x-a)$ for positive a will add at least one sign variation to the coefficients. □

We also have two converse propositions.

Proposition 4.21. *Let P be a monic polynomial in $\mathbb{R}[x]$. If all the roots of P have non-positive real part, then $Var(P) = 0$.*

Proof. Write

$$P = (x - a_0) \cdot \cdots \cdot (x - a_s) \cdot (x^2 + \alpha_0 x + \beta_0) \cdot \cdots \cdot (x^2 + \alpha_t x + \beta_t),$$

with real numbers $a_i, \alpha_i, \beta_i \in \mathbb{R}$ such that

$$x^2 + \alpha_i x + \beta_i = (x - z_i)(x - \bar{z}_i)$$

for some complex number $z_i \in \mathbb{C}$. Observe, that the coefficients of the polynomials $(x - a_i)$ and $(x^2 + \alpha_t x + \beta_t)$ are non-negative and that the product of two polynomials with non-negative coefficients has non-negative coefficients. □

Definition 4.22. Let $A \in \mathbb{R}[x]$ be a polynomial with non-negative coefficients, $A = a_d x^d + \cdots + a_0$. A is called **normal** if

(1) $a_d > 0$
(2) $a_k^2 \geq a_{k-1}a_{k+1}$ for all $1 \leq k \leq d-1$
(3) If $a_j > 0$ and $a_h > 0$ then $a_i > 0$ for all $i \in \{j \dots h\}$.

Proposition 4.23. *Let $P \in \mathbb{R}[x]$ be a monic polynomial. If all the roots of P belong to the cone $\mathfrak{B}$ in the complex plane defined by*

$$\mathfrak{B} := \{a + ib \mid b \leq -\sqrt{3}a\}$$

then P is normal.

Proof. See [BPR08], Proposition 2.40. □

Proposition 4.24. *If $A \in \mathbb{R}[x]$ is normal and $a > 0$ then $Var((x-a)A) = 1$.*

Proof. We may suppose without loss of generality that 0 is not a root of A. It then follows from condition (iii) of Definition 4.22 that all coefficients of A are positive. Condition (ii) can then be written as

$$\frac{a_d}{a_{d-1}} \leq \frac{a_{d-1}}{a_{d-2}} \leq \dots$$

or equivalently

$$\frac{a_{d-1}}{a_d} \geq \frac{a_{d-2}}{a_{d-1}} \geq \dots$$

It follows that

$$A(x-a) = a_d x^{d+1} + a_d \left(\frac{a_{d-1}}{a_d} - x\right) x^d + \dots + a_1 \left(\frac{a_0}{a_1} - x\right) x - a_0 x$$

has exactly one sign variation. □

Given a polynomial $P \in \mathbb{R}[x]$ we define

$$\operatorname{Rec}_p(P(x)) := x^p P(1/x)$$

The non-zero roots of $\operatorname{Rec}_{d_P} P$ are of the form $1/a$, where a is a root of P

$$\operatorname{Co}_\lambda(P(x)) := P(\lambda x)$$

The roots of $\operatorname{Co}_\lambda P$ are of the form a/λ, where a is a root of P.

$$\operatorname{T}_c(P(x)) := P(x-c).$$

The roots of $\operatorname{T}_c P$ are of the form $a+c$, where a is a root of P.

Descartes' rule of signs (Theorem 4.20) yields a sufficient criterion for determining whether a polynomial has a root in a given interval.

Theorem 4.25. *Let $P \in \mathbb{R}[x]$ and let $\hat{P} := \operatorname{T}_{-1} \operatorname{Rec}_{d_P} \operatorname{Co}_{r-l} \operatorname{T}_{-l}$. If $Var(\hat{P}) = 0$, then P has no root in (l,r), if $Var(\hat{P}) = 1$, then P has exactly one simple root in (l,r).*

Proof. The roots of $\hat{P}$ in $(0,\infty)$ correspond to the roots of P in (l,r). By Theorem 4.20, $\operatorname{Var}(\hat{P}) \geq pos(\hat{P})$, so if $\operatorname{Var}(\hat{P}) = 0$, then $\hat{P}$ has no roots in $(0,\infty)$, i.e. P has no roots in (l,r) and if $\operatorname{Var}(\hat{P}) = 1$, then $pos(\hat{P}) = 1$, so $\hat{P}$ has exactly one simple root in $(0,\infty)$, i.e. P has exactly one simple root in (l,r). □

Let $\mathcal{C}(l,r)$ denote the open disk in $\mathbb{C}$ with $[l,r]$ as diameter. Let $\mathcal{C}_1(l,r)$ denote the open disk circumscribing the equilateral triangle based in $[l,r]$ and $\mathcal{C}_2(l,r)$ be the open disk symmetric to $\mathcal{C}_1(l,r)$ with respect to the x-axis.

Theorem 4.26 ('Theorem of Three Circles')**.** *Let $P \in \mathbb{R}[x]$ and let*

$$\hat{P} := \operatorname{T}_{-1} \operatorname{Rec}_{d_P} \operatorname{Co}_{r-l} \operatorname{T}_{-l} P$$

If P has no root in $\mathcal{C}(l,r)$ then $Var(\hat{P}) = 0$. If P has exactly one simple root in $\mathcal{C}_1(l,r) \cup \mathcal{C}_2(l,r)$ then $Var(\hat{P}) = 1$.

Proof. The roots of P in the complement of $\mathcal{C}(l,r)$ correspond to the roots of $\hat{P}$ in the half-plane $\{x \in \mathbb{C} | \operatorname{Re} x \leq 0\}$. Hence, if P has no roots in $\mathcal{C}(l,r)$ then by Proposition 4.21 we have $\operatorname{Var}(\hat{P}) = 0$.

The roots of P in the complement of $\mathcal{C}_1(l,r) \cup \mathcal{C}_2(l,r)$ correspond to the roots of $\hat{P}$ in the cone $\mathfrak{B} = \{a+ib \mid b \leq -\sqrt{3}a\}$. If P has exactly one simple root $\alpha \in \mathcal{C}_1(l,r) \cup \mathcal{C}_2(l,r)$, then α is a real number in (l, r) and $\hat{P}$ has exactly one real root $\alpha' \in (0, \infty)$ and all its other roots lie in $\mathfrak{B}$. According to Proposition 4.23, $A(x) := \hat{P}(x)/(x - \alpha')$ is normal, so $\operatorname{Var}(\hat{P}) = \operatorname{Var}(A(x - \alpha')) = 1$ by 4.24. ☐

Proposition 4.27 (Cauchy Bound)**.** *Let $P \in \mathbb{R}[x]$, $P = a_d x^d + \cdots + a_0$. Let*

$$C(P) := \sum_{i=0}^{d} \left| \frac{a_i}{a_d} \right| .$$

Then the absolute value of any real root of P is smaller than $C(P)$.

Proposition 4.28. *Given a $\nu_{\mathbb{D}[x]} \times \nu_{\mathbb{D}}$-name of $P \in \mathbb{D}[x]$ and $c \in \mathbb{D}$ we can compute a $\nu_{\mathbb{D}[x]}$-name of $\operatorname{Co}_c P$ in time $\mathcal{O}(d_P^2 \ell(c)(b_P + \ell(c)))$. The output polynomial's coefficients are bounded by $b_P + d_P \ell(c)$.*

Proof. Let $c_k := c^k$. Then $\ell(c_k) = k\ell(c)$. We compute for $k = 1, \ldots, d_P$, $\hat{a}_k := c_k a_k$ and $c_{k+1} := k c_k$ and output the list $(\hat{a}_0, \ldots, \hat{a}_{d_P})$. Visibly, $\hat{a}_k$ has size $b_P + k\ell(c)$. The complexity of computing $c_1, \ldots, c_{d_P}$ is $\mathcal{O}(d_P^2 \ell(c))$ and the complexity of computing $\hat{a}_1, \ldots,$ given $c_1, \ldots, c_{d_P}$ is $\mathcal{O}(d_P^2 b_P \ell(c))$. ☐

A binary tree T whose nodes are labelled by open intervals with dyadic rational endpoints such that for every node labelled by (l, r), its children are labelled by (l, m) and (m, r) respectively with $m = \frac{r-l}{2}$ is called an **isolating tree** for a polynomial $P \in \mathbb{R}[x]$ if

- The interval labelling the root of the tree contains all the roots of P in $\mathbb{R}$.
- If the interval (l, r) labels a leaf of the tree then it contains either no root or exactly one (possibly multiple) root of P.
- If the interval (l, r) labels a node of the tree which is not a leaf then either P has at least two roots in $\mathcal{C}(l, r)$ or the interval (l, r) contains exactly one root of P and $\mathcal{C}_1(l, r) \cup \mathcal{C}_2(l, r)$ contains a pair of conjugate roots.

If T is an isolating tree for P, a leaf which is labelled by an interval containing a root of P will be referred to as a **leaf of type one** and a leaf which is labelled by an interval containing no root of P will be called a **leaf of type zero**.

Proposition 4.29. *Let T be an isolating tree for a polynomial P. Let U denote the number of nodes of T. Let T' be a subtree of T obtained by pruning certain leaves from T: we prune every leaf that has a non-leaf for a sibling and for every pair of leaves we prune exactly one, where a leaf of type zero is pruned preferably to a leaf of type one. Let L' denote the set of T''s leaves and for a leaf $u \in L$ denote by h_u its depth. Then*

$$U \leq 2 \sum_{u \in L'} h_u \leq (2b_P + 3 \log d_P + 3) d_P$$

We are now able to formulate the root isolation algorithm. The main idea is to use Theorems 4.26 and 4.25 to decide whether a given separable polynomial $P \in \mathbb{D}[x]$ has a real root in a given interval with dyadic rational endpoints (r, l). This leads to the task of computing $\hat{P} := \mathrm{T}_{-1} \operatorname{Rec}_{d_P} \operatorname{Co}_{r-l} \mathrm{T}_{-l} P \in \mathbb{D}[x]$. If we have already computed this transform on the interval (l, r), we can recursively compute the transform on subintervals. For dyadic rational numbers $r, l \in \mathbb{D}$, let $P[l, r] := \operatorname{Co}_{r-l} \mathrm{T}_{-l} P$. If $m = \frac{r-l}{2}$, we have

$$P[l, m] = \operatorname{Co}_{1/2} P[l, r]$$

and

$$P[m,r] = \mathrm{T}_{-1}\, P[l,m]$$

Lemma 4.30. *(i) Computing* T_{-1} *takes time* $\mathcal{O}(d_P^2(b_P + d_P))$ *and has output size* $\mathcal{O}(b_P + d_P)$*.*

(ii) Computing $\mathrm{Co}_{1/2}$ *takes time* $\mathcal{O}(d_P(b_P + d_P))$ *and has output size* $\mathcal{O}(b_P + d_P)$*.*

Theorem 4.31. *There exists an algorithm which, given a* $\nu_{\mathbb{D}[x]}$*-name of some non-zero polynomial* $P \in \mathbb{D}[x]$ *returns a* $(\nu_{\mathbb{D}} \times \nu_{\mathbb{D}} \times \nu_{\mathbb{D}[x]})^*$*-name of a list isolating the zeroes of* P *in* $\mathbb{R}$ *with corresponding polynomials* $\bar{P}[l,r]$*, where* $\bar{P}$ *is the separable part of* P*. Its complexity is bounded by* $\mathcal{O}(d_P^5(b_P + \log d_P)^2)$*. The resulting list has at most* d_P *elements whose size is bounded by* $b_P + \log(d_P + 1) + d_P(4b_P + 6\log d_P + 6)$*.*

Proof. We first compute the logarithm of the Cauchy-bound B of P. Now, compute the separable part of P and denote it by Q. Let $L := \{((2^{-\log B}, 2^{\log B}), Q[2^{-\log B}, 2^{\log B}])\}$, let R be the empty list. While L is nonempty, remove an element $((l,r), Q[l,r])$ from L and compute

$$\hat{Q} := \mathrm{T}_{-1}\, \mathrm{Rec}_{d_P}\, Q[l,r] \in \mathbb{D}[x].$$

If $\mathrm{Var}(\hat{Q}) = 1$, add $((l,r), Q[l,r])$ to R. If $\mathrm{Var}(\hat{Q}) > 1$, let $m := \frac{r-l}{2}$, compute

$$Q[l,m] = \mathrm{Co}_{1/2}\, Q[l,r]$$

and

$$Q[m,r] = \mathrm{T}_{-1}\, Q[l,m]$$

and add $((l,m), Q[l,m])$ and $((m,r), Q[m,r])$ to L. If $Q(m) = 0$, add $(\{m\}, Q[l,r])$ to R.

The fact that this algorithm terminates and its correctness follow immediately from Theorem 4.26 and Theorem 4.25. Note that the algorithm produces an isolating tree for P, so by 4.29 the algorithm produces at most $\mathcal{O}(d_P(b_P + \log d_P))$ intervals. By the above lemma, the coefficient size of the polynomial occurring in the computation is in $\mathcal{O}(d_P^2(b_P + \log d_P))$, so at each algorithmic step the computation takes time $\mathcal{O}(d_P^4(b_P + \log d_P))$, which yields a total complexity of $\mathcal{O}(d_P^5(b_P + \log d_P)^2)$. As for the size, $C(P) \leq (d_P + 1)2^{b_P}$ and the height of the tree produced by the algorithm is bounded by $\sum_{u \in L'} h_u \leq d_P(2b_P + 3\log d_P + 3)$. At each step, the endpoints of the intervals are added and divided by 2, so their size increases by at most two - each interval is treated at most $d_P(2b_P + 3\log d_P + 3)$ times, so the size of the endpoints can be estimated by $b_P + \log(d_P + 1) + d_P(4b_P + 6\log d_P + 6)$. □

Theorem 4.32. *There exists an algorithm which, given a* $\nu_{\mathbb{D}[x]} \times \nu_1$*-name of some nonzero polynomial* $P \in \mathbb{D}[x]$ *and* $1^n, n \in \mathbb{N}$ *returns a* $\nu_{\mathbb{D}}^*$*-name of a list of dyadic approximations to the roots of* P *in* $\mathbb{R}$ *up to error* 2^{-n}*, which has running time*

$$\mathcal{O}(d_P^5(b_P + \log d_P)^2 + nd_P^4(b_P + \log d_P + n) + nd_P^5(b_P + \log d_P)).$$

The output list has at most d_P *elements and the endpoints of the intervals have size at most*

$$\mathcal{O}(d_P(b_P + \log d_P) + n)$$

Proof. Find a root isolating list L for P and the polynomials $\bar{P}[l,r]$ according to Theorem 4.31. Then for every element $((l,r),P[l,r])$ in L, while $r-l>2^{-n}$ compute

$$P[l,m] := \mathrm{Co}_{1/2}\,P[l,r]$$
$$P[m,r] := \mathrm{T}_{-1}\,P[l,m]$$

and

$$\hat{P} := \mathrm{T}_{-1}\,\mathrm{Rec}_{d_P}\,P[l,m].$$

If we have $\mathrm{Var}(\hat{P})=1$, we replace $((l,r),P[l,r])$ by $((l,m),P[l,m])$, otherwise we replace it by $((m,r),P[m,r])$. The polynomial $\bar{P}$ has coefficient size $d_P+b_P+\log(d_P+1)$, so by Theorem 4.31, the computation of the isolating list will take time at most $\mathcal{O}(d_P^5(b_P+\log d_P)^2)$, the polynomials in the output list have coefficient size $\mathcal{O}(d_P^2(b_P+\log d_P))$ and the endpoints of the intervals have size $\mathcal{O}(d_P(b_P+\log d_P))$. Since the endpoints of the intervals are contained in the compact interval $[-2^{C(P)},2^{C(P)}]$ it follows, that the algorithm terminates after at most $\mathcal{O}(d_P(b_P+\log d_P+n))$ steps, taking $\mathcal{O}(b_P+\log d_P+n)$ steps for the refinement of each interval in the root isolation. It then follows from 4.30 that the polynomials occurring in the computation have coefficient size

$$\mathcal{O}(d_P^2(b_P+\log d_P)+d_P(b_P+\log d_P+n)) = O(d_P^2(b_P+\log d_P)+d_Pn),$$

so at each step the computation takes time

$$\mathcal{O}(d_P^2(d_P^2(b_P+\log d_P)+d_Pn+d_P) = \mathcal{O}(d_P^4(b_P+\log d_P)+d_P^3n).$$

Since there are $\mathcal{O}(d_P(b_P+\log d_P+n))$ steps to perform, the total complexity of the operations on the polynomials is given by

$$\mathcal{O}(d_P^5(b_P+\log d_P)^2+nd_P^4(b_P+\log d_P+n)+nd_P^5(b_P+\log d_P)).$$

Similarly, the size of the endpoints of the intervals increases by at most two in every algorithmic step, and for each interval there are $\mathcal{O}(b_P+\log d_P+n)$ steps performed, so the size of the endpoints of the output intervals is at most $\mathcal{O}(d_P(b_P+\log d_P)+n)$. The total computational complexity of the operations on the intervals is thus bounded by $\mathcal{O}(d_P(b_P+\log d_P)^2+n(d_P(b_P+\log d_P)+n))$. □

Theorem 4.33. *Given a $\nu_{\mathbb{D}[x]}\times\nu_{\mathbb{D}}\times\nu_{\mathbb{D}}$-name of some polynomial $P\in\mathbb{D}[x]$ and $a,b\in\mathbb{D}$, we can compute a $\nu_{\mathbb{D}}$-name of a dyadic number approximating $\max_{x\in[a,b]}P(x)$ up to error 2^{-n} in time*

$$\mathcal{O}(d_P^5(b_P+d_P)(b_P+d_P+n)+d_P^4(b_P+d_P+n)^2+d_P^2M^2),$$

where M is a bound on the bitsize of a and b. The output has size at most

$$\mathcal{O}(d_PM+d_Pn+d_P^2(b_P+d_P)).$$

Proof. First compute the the derivative of P and denote its separable part by Q. Then approximate its zeroes $\{x_1,\dots,x_k\}$ in $[a,b]$ by dyadic numbers $\{\xi_1,\dots,\xi_k\}$ up to error $2^{-n-\log C}$ where C is the Lipschitz-constant found in 4.16. We then have the inequality $|P(x_i)-P(\xi_i)|\le 2^{-n}$, so $\max\{P(a),P(\xi_1),\dots,P(\xi_k),P(b)\}$ approximates the maximum of P on $[a,b]$ up to error 2^{-n}. Computing the derivative can be done in time $\mathcal{O}(d_Pb_P\log d_P)$ with output size $b_P+\log d_P$ (cf. 4.13), computing the separable part takes time $\mathcal{O}(d_P^3(b_P+\log d_P))$ and the resulting polynomial Q has degree at most d_P and coefficient size $d_P+b_P+\log d_P+\log(d_P+1)$ according to 4.19. For the bounds found in 4.16, we may use the estimate $c_P \leq b_P$, so computing the logarithm of some Lipschitz constant takes time $\mathcal{O}(d_P(d_P + b_P + \log d_P))$ and has output size $3\log(d_P + 1) + d_P + b_P + \log d_P$. So Proposition 4.32 yields the desired approximation to the roots of Q in $\mathbb{R}$ in time $\mathcal{O}(d_P^4(b_P + d_P + n)^2 + d_P^5(b_P + d_P)(b_P + d_P + n))$, and the endpoints of the intervals have size at most $\mathcal{O}(d_P(b_P + d_P) + n)$. To obtain the roots in $[a, b]$, we drop the intervals whose right endpoint is smaller than a and those whose left endpoint is bigger than b. If there is an interval $[c, d]$ such that $a \in [c, d]$, we evaluate Q in a and in d to decide whether the root of Q in $[c, d]$ is contained in (a, d), in which case we add the interval (a, d) to the list (note that since Q is separable, it will change its sign in the root). We do the same for b. According to Proposition 4.6, this step takes time $\mathcal{O}(d_P^2(d_P(b_P + d_P) + M + n)^2)$. Finally, we perform at most d_P evaluations of Q in numbers of size $\mathcal{O}(d_P(b_P + d_P) + n)$, and since Q has coefficient size $\mathcal{O}(d_P + b_P)$, each of these evaluations is computable in time $\mathcal{O}(d_P(d_P + b_P + d_P(d_P(b_P + d_P) + n))^2)$ according to Proposition 4.6, which simplifies to $\mathcal{O}(d_P^5(b_P + d_P)^2 + d_P^3 n^2)$. So the overall complexity is bounded by

$$\mathcal{O}(d_P^5(b_P + d_P)(b_P + d_P + n) + d_P^4(b_P + d_P + n)^2 + d_P^2 M^2).$$

The output size is given by Proposition 4.6 as $\mathcal{O}(d_P M + d_P n + d_P^2(b_P + d_P))$. □

Theorem 4.34. *Given a $\nu_{\mathbb{D}_m(x)} \times \nu_{\mathbb{D}} \times \nu_{\mathbb{D}}$-name of some rational function $R \in \mathbb{D}_m(x)$ and $a, b \in \mathbb{D}$, we can compute a $\nu_{\mathbb{D}}$-name of a dyadic approximation to $\max_{x \in [a,b]} R(x)$ up to error 2^{-n} in time*

$$\mathcal{O}(d_P^5(b_P + d_P)(b_P + d_P + n) + d_P^4(b_P + d_P + n)^2 + d_P^2 M^2).$$

Proof. Let $R := P/Q$. We first compute the separable part of $P'Q - Q'P$ and then proceed as in 4.33, replacing in the last step polynomial evaluation by rational function evaluation (4.8). The coefficients of $P'Q - Q'P$ are bounded by $2b_R + 2\log d_R + d_R + 2 \in \mathcal{O}(b_R + d_R)$ according to 4.9 and the degree of the polynomial is at most $2d_R \in \mathcal{O}(d_R)$, so we obtain the same bound for the overall complexity as in 4.33. □

4.3. **Algorithms on Piecewise Polynomials and Rational Functions.** Let $\mathbb{PP}$ and $\mathbb{PQ}$ denote the subsets of $C[0, 1]$ consisting of piecewise polynomial and rational functions bounded on $[0, 1]$ with dyadic rational breakpoints and dyadic rational coefficients respectively. In this section we introduce notations $\nu_{\mathfrak{pp}}$ and $\nu_{\mathfrak{pq}}$ of $\mathbb{PP}$ and $\mathbb{PQ}$ and generalise the algorithms on polynomials and rational functions developed so far to the piecewise case. These representations will induce Cauchy representations of the space of continuous functions, which will be studied in Section 5. Approximations by piecewise polynomial functions are used for the Finite Element Method in numerical analysis of partial differential equations (often with further conditions to the smoothness of the approximating functions) and are supported by standard computer algebra libraries.

Definition 4.35.
(i) A $\nu_{\mathfrak{pp}}$-name of $f \in \mathbb{PP}$ is a $\nu_{\mathbb{D}}^* \times \nu_{\mathbb{D}[x]}^*$-name of a tuple of dyadic rational points $\vec{a}$, $0 = a_0 < a_1 < \cdots < a_N = 1$ and a tuple of polynomials $\vec{P}$ in $\mathbb{D}[x]$, $|\vec{P}| = |\vec{a}| - 1$ such that $f = P_i$ on $[a_i, a_{i+1}]$.
(ii) A $\nu_{\mathfrak{pq}}$-name of $f \in \mathbb{PQ}$ is a $\nu_{\mathbb{D}}^* \times \nu_{\mathbb{D}_m(x)}^*$-name of a tuple of dyadic rational points $\vec{a}$, $0 = a_0 < a_1 < \cdots < a_N = 1$ and a tuple of rational functions $\vec{R}$ in $\mathbb{D}_m(x)$, $|\vec{R}| = |\vec{a}| - 1$ such that $f = R_i$ on $[a_i, a_{i+1}]$.

Definition 4.36. If f is in $\mathbb{PP}$ or $\mathbb{PQ}$ we write B_f for the number of breakpoints and p_f for a bound on their bit-size and define the numbers d_f, c_f, b_f in the obvious manner.

The size $\ell(f)$ of a piecewise polynomial function f is bounded by $B_f d_f b_f + B_f p_f$ and the size of a piecewise rational function g is bounded by $2B_g d_g b_g + B_g p_g$.

Theorem 4.37. *(i) Addition,*

$$\mathbb{PP} \times \mathbb{PP} \ni (f,g) \mapsto f+g \in \mathbb{PP}$$

is $(\nu_{\mathsf{pp}} \times \nu_{\mathsf{pp}}, \nu_{\mathsf{pp}})$-computable in time

$$\mathcal{O}((B_f + B_g)((d_f + d_g)(b_f + b_g) + p_f + p_g)).$$

The resulting function has at most $B_f + B_g$ breakpoints of size $\max\{p_f, p_g\}$, its degree is bounded by $\max\{d_f, d_g\}$ and its coefficients are bounded by $\max\{b_f, b_g\}+1$.

(ii) Multiplication,

$$\mathbb{PP} \times \mathbb{PP} \ni (f,g) \mapsto f \cdot g \in \mathbb{PP}$$

is $(\nu_{\mathsf{pp}} \times \nu_{\mathsf{pp}}, \nu_{\mathsf{pp}})$-computable in time

$$\mathcal{O}((B_f + B_g)((d_f + d_g)^2(b_f + b_g)^2 + p_f + p_g).$$

The resulting function has at most $B_f + B_g$ breakpoints of size $\max\{p_f, p_g\}$, its degree is bounded by $d_f + d_g$ and its coefficients are bounded by $b_f + b_g + \min\{d_f, d_g\}$.

Proof. (i) We arrange the breakpoints of f and g in a common sorted list, which takes time $\mathcal{O}(\max\{B_f, B_g\}\max\{p_f, p_g\})$ and on each interval defined by this new partition of $[0,1]$ we add the polynomials defined on said interval using 4.9.

(ii) We proceed as in (i), replacing addition by multiplication.

□

Proposition 4.38. *Given a $\nu_{\mathsf{pp}} \times \nu_{\mathsf{pp}} \times \nu_1$-name of $(f,g) \in \mathbb{PP}^2$ and $1^n, n \in \mathbb{N}$ with $g([0,1]) \subseteq [0,1]$ we can compute a ν_{pp}-name of $\hat{f} \in \mathbb{PP}$ uniformly approximating $f \circ g$ up to error 2^{-n} in time*

$$\mathcal{O}\left(B_g B_f d_g^5 (b_g + p_f + \log d_g)^2 + B_g B_f N d_g^4 (b_g + p_f + \log d_g + N)\right)$$
$$+\mathcal{O}\left(B_g B_f N d_g^5 (b_g + p_f + \log d_g) + B_g B_f d_f^3 d_g^3 (d_f b_g + d_f d_g + b_f)^2\right),$$

where $N := n + c_g + c_f + \log d_f + \log d_g$.

Proof. We first refine the partition of $[0,1]$ defined by the breakpoints of g, such that the interval is divided into subintervals on which $f \circ g$ is a polynomial and then apply the polynomial composition algorithm. Let $\vec{b}$ denote the breakpoints of f and $\vec{a}$ the breakpoints of g. For every $1 \leq k \leq B_P - 1$ we compute $g(a_k)$ and $g(a_{k+1})$ and find i_1 and i_2 such that $b_{i_1} \leq g(a_k) \leq g(a_{k+1}) \leq b_{i_2}$. We now need to find the preimages of b_j for $i_1 \leq j \leq i_2$ under g and add them to the breakpoints of g. In order to do so we need to solve the equation $g(x) = b_j$, where $x \in [a_k, a_{k+1}]$. We can do this approximatively up to error $2^{-\mu(n)}$, where $\mu(n)$ is a modulus of continuity of $f \circ g$ using 4.32. This yields points l_j^i, r_j^i such that for every $x \in [a_k, a_{k+1}]$ the equation $g(x) = b_j$ implies that there exists an i such that $x \in [l_j^i, r_j^i]$. We add these points to the breakpoints of g. Now, for every pair of successive breakpoints a, b with $|a - b| > 2^{-\mu(n)}$ we can determine an interval

$[b_{j_1}, b_{j_2}]$ such that $g([a,b]) \subseteq [b_{j_1}, b_{j_2}]$, so we can compute $f \circ g$ on the interval $[a,b]$ using 4.10. If the pair of successive breakpoints has distance smaller than $2^{-\mu(n)}$, then for all $x \in [a,b]$ we have $|f(g(x)) - f(g(a))| \leq 2^{-n}$, so the linear function interpolating $f \circ g$ in a and b approximates $f \circ g$ on $[a,b]$ up to error 2^{-n}. A modulus of continuity μ of $f \circ g$ is given by $\mu(n) := n + L_f + L_g$, where L_f and L_g are logarithms of Lipschitz-constants of f and g respectively, which can be found in time $\mathcal{O}(B_f d_f c_f + B_g d_g c_g)$ with output size $c_f + \log d_f$ and $c_g + \log d_g$ according to 4.43. The approximation of the preimages can be done in time

$$\mathcal{O}\left(B_f\left(d_g^5(b_g+p_f+\log d_g)^2 + N d_g^4(b_g+p_f+\log d_g+N) + N d_g^5(b_g+p_f+\log d_g)\right)\right)$$

according to 4.32, where $N := n + c_g + c_f + \log d_f + \log d_g$. Since on every interval defined by the breakpoints of g we introduce at most $2B_f d_g$ new breakpoints, the 'polynomial composition step' on one interval takes time

$$\mathcal{O}(B_f d_f^3 d_g^3 (d_f b_g + d_f d_g + b_f)^2).$$

Since these steps are performed B_g times, the total complexity is given as

$$\mathcal{O}\left(B_g B_f d_g^5(b_g+p_f+\log d_g)^2 + B_g B_f N d_g^4(b_g+p_f+\log d_g+N)\right)$$
$$+\mathcal{O}\left(B_g B_f N d_g^5(b_g+p_f+\log d_g) + B_g B_f d_f^3 d_g^3(d_f b_g + d_f d_g + b_f)^2\right).$$

□

Proposition 4.39. *Given a $\nu_{\mathfrak{pq}} \times \nu_{\mathfrak{pq}} \times \nu_1$-name of $(f,g) \in \mathbb{PQ}^2$ and $1^n, n \in \mathbb{N}$ with $g([0,1]) \subseteq [0,1]$ we can compute a $\nu_{\mathfrak{pq}}$-name of $\hat{f} \in \mathbb{PQ}$ uniformly approximating $f \circ g$ up to error 2^{-n} in time*

$$\mathcal{O}\left(B_g B_f d_g^5(b_g+p_f+\log d_g)^2 + B_g B_f N d_g^4(b_g+p_f+\log d_g+N)\right)$$
$$+\mathcal{O}\left(B_g B_f N d_g^5(b_g+p_f+\log d_g) + B_g B_f d_f^3 d_g^3(d_f b_g + d_f d_g + b_f)^2\right)$$

where $N := n + c_g + c_f + \log d_f + \log d_g$.

Proof. The proof is almost exactly identical as in the case of piecewise polynomial functions. The only subtlety we have to be aware of, is that we cannot compute the linear interpolating functions in the same way as in the above proof, since we cannot evaluate rational functions exactly. Instead, we represent a linear function of the type

$$\frac{P(a)}{Q(a)} + \frac{x-a}{b-a}\left(\frac{P(b)}{Q(b)} - \frac{P(a)}{Q(a)}\right)$$

as a rational function with denominator $(b-a)Q(a)Q(b)$ and numerator

$$(P(b)Q(a) - P(a)Q(b))\,x - aP(b)Q(a) + bP(a)Q(b).$$

Note that in order to compute the preimages of the breakpoints of f we have to solve equations of the form $P(x)/Q(x) = b$, which can be written as $P(x) - bQ(x) = 0$ and thus lead to polynomial equations of the same magnitude as in Proposition 4.38. □

Theorem 4.40. *(i) There exists an algorithm, which, given a $\nu_{\mathfrak{pq}} \times \nu_{\mathfrak{pq}}$-name of $(f,g) \in \mathbb{PQ}^2$ outputs a ν_{pfrac}-name of $f+g$ in time $\mathcal{O}((B_f + B_g)((d_f+d_g)^2(b_f+b_g)^2 + p_f + p_g))$.*

(ii) *There exists an algorithm, which, given a $\nu_{\mathsf{pq}} \times \nu_{\mathsf{pq}}$-name of $(f,g) \in \mathbb{PQ} \times \mathbb{PQ}$ outputs a ν_{pfrac}-name of $f \cdot g \in \mathbb{PQ}$ in time $\mathcal{O}((B_f + B_g)((d_f + d_g)^2(b_f + b_g)^2 + p_f + p_g))$.*

Theorem 4.41. *Evaluation of f in $\xi \in \mathbb{D}$ is $(\nu_{\mathsf{pp}} \times \nu_{\mathbb{D}}, \nu_{\mathbb{D}})$-computable in time*

$$\mathcal{O}(B_f(\ell(\xi) + p_f) + d_f(b_f + d_f\ell(\xi))\ell(\xi))$$

with output size at most $b_f + d_f(\ell(\xi) + 1)$ and $(\nu_{\mathsf{pq}} \times \nu_{\mathbb{D}} \times \nu_1, \nu_{\mathbb{D}})$-computable in time

$$\mathcal{O}\left(B_f(\ell(\xi) + p_f) + d_f(b_f + d_f\ell(\xi))\ell(\xi) + n(b_f + d_f\ell(\xi))\right).$$

Proof. Let $\vec{a}$ denote the breakpoints of f. We first determine $0 < k < |\vec{a}|$ such that $\xi \in [a_k, a_{k+1}]$ and then apply Propositions 4.6 or 4.8 respectively. □

Theorem 4.42. (i) *Given a $\nu_{\mathsf{pp}} \times \nu_{\mathbb{D}} \times \nu_{\mathbb{D}} \times \nu_1$-name of $f \in \mathbb{PP}$ and $a, b \in \mathbb{D}$, $\ell(a), \ell(b) \leq M$ and $n \in \mathbb{N}$ we can compute a $\nu_{\mathbb{D}}$-name of an approximation to $\max_{x\in[a,b]} f(x)$ up to error 2^{-n} in time*

$$\mathcal{O}(B_f d_f^5(b_f + d_f)(b_f + d_f + n) + B_f d_f^4(b_f + d_f + n)^2 + B_f d_f^2(M + p_f)^2).$$

(ii) *Given a $\nu_{\mathsf{pp}} \times \nu_{\mathbb{D}} \times \nu_{\mathbb{D}} \times \nu_1$-name of $f \in \mathbb{PQ}$ and $a, b \in \mathbb{D}$, $\ell(a), \ell(b) \leq M$ and $n \in \mathbb{N}$ we can compute a $\nu_{\mathbb{D}}$-name of an approximation to $\max_{x\in[a,b]} f(x)$ up to error 2^{-n} in time*

$$\mathcal{O}(B_f d_f^5(b_f + d_f)(b_f + d_f + n) + B_f d_f^4(b_f + d_f + n)^2 + B_f d_f^2(M + p_f)^2).$$

Proof. We first determine $j \leq k$ such that $a \in [a_j, a_{j+1}]$ and $b \in [a_k, a_{k+1}]$. This takes time $\mathcal{O}(B_f(p_f + M))$ for both representations. We may assume without loss of generality that $j < k$ (otherwise we apply 4.33, or 4.34 right away). We then compute the numbers $m_j := \max\{f(x)|x \in [a, a_{j+1}]\}$, $m_k := \max\{f(x)|x \in [a_k, b]\}$ and for $i = j+1, \ldots k-1$ the numbers $m_i := \max\{f(x)|x \in [a_i, a_{i+}]$ up to precision 2^{-n} using 4.33 or 4.34 respectively. Then the maximum of f on $[a, b]$ is given by $\max\{m_j, \ldots, m_k\}$. This computation takes time

$$\mathcal{O}(B_f d_f^5(b_f + d_f)(b_f + d_f + n) + B_f d_f^4(b_f + d_f + n)^2 + B_f d_f^2(M + p_f)^2)$$

which clearly dominates the overall complexity. □

Theorem 4.43. (i) *Given a ν_{pp}-name of $f \in \mathbb{PP}$ we can compute a ν_1-name of the logarithm of some Lipschitz-constant of f in time $O(B_f d_f c_f)$ with output-size $2\log(d_f + 1) + c_f$. Furthermore, we can compute a ν_1-name of the logarithm of some number majorising f on $[0, 1]$ in time $O(B_f d_f c_f)$ with output-size $c_f + \log(d_f)$.*

(ii) *Given a ν_{pq}-name of $f \in \mathbb{PQ}$ we can compute a ν_1 name of the logarithm of some Lipschitz-constant of f in time $O(B_f d_f c_f)$ with output size $3\log(d_f + 1) + 2c_f + 1$. Furthermore, we can compute a ν_1-name of the logarithm of some number majorising f on $[0, 1]$ in time $O(B_f d_f c_f)$ with output-size $c_f + \log(d_f)$.*

Theorem 4.44. *Given a $\nu_{pp} \times \nu_{\mathbb{D}} \times \nu_{\mathbb{D}} \times \nu_1$-name of $f \in \mathbb{PP}$ and $a, b \in \mathbb{D}, a < b$, and $n \in \mathbb{N}$ we can compute a $\nu_{\mathbb{D}}$ name of an approximation to $\int_a^b f(x)dx$ up to error 2^{-n} in time*

$$\mathcal{O}(B_f d_f(b_f + n + d_f(\ell(a) + \ell(b) + p_f))(\ell(a) + \ell(b) + p_f) + B_f d_f \log d_f(b_f + \log d_f + n)).$$

Proof. We may suppose that there is at least one breakpoint between a and b. Let $0 = g_1 < \cdots < g_m = 1$ denote the breakpoints of f. First we need to determine k and l such that $g_k \leq a \leq g_{k+1}$ and $g_l \leq b \leq g_{l+1}$. This can be done in time $\mathcal{O}(B_f(\ell(a) + p_f))$. Then we compute

$$\int_a^{g_{k+1}} f(x)dx + \cdots + \int_{g_l}^b f(x)dx$$

according to 4.14, which can be done in time

$$\mathcal{O}(B_f d_f(b_f + n + d_f(\ell(a) + \ell(b) + p_f))(\ell(a) + \ell(b) + p_f) + B_f d_f \log d_f(b_f + \log d_f + n)).$$

□

5. Representations of The Space of Continuous Functions

5.1. **Parametrised Complexity of Functionals on $C([0,1])$.** Let $C([0,1])$ denote the space of continuous functions on the compact interval $[0,1]$. Using the framework introduced in Section 3 we can now state the computational problems mentioned in the introduction more formally. Subject of this section is to study the computational complexity of

- The evaluation functional on $C([0,1])$ with parameters in $[0,1]$ represented by the co-restriction of ρ_c to $[0,1]$ and with image space $\mathbb{R}$ represented by ρ.

$$\mathbf{EVAL}\colon\ C([0,1]) \times [0,1] \to \mathbb{R}, (f,x) \mapsto f(x)$$

- The maximisation functional on $C([0,1])$ with parameters in $[0,1]^2$ represented by the co-restriction of $\rho_c \times \rho_c$ to $[0,1] \times [0,1]$ and with image space $\mathbb{R}$ represented by ρ.

$$\mathbf{MAX}\colon\ C([0,1]) \times [0,1]^2 \to \mathbb{R}, (f,a,b) \mapsto \max\{f(x) \mid \min\{a,b\} \leq x \leq \max\{a,b\}\}$$

- The integration functional on $C([0,1])$ with parameters in $[0,1]^2$, represented by the co-restriction of $\rho_c \times \rho_c$ to $[0,1] \times [0,1]$ and with image space $\mathbb{R}$ represented by ρ.

$$\mathbf{INT}\colon ; C([0,1]) \times [0,1]^2 \to \mathbb{R},\ (f,a,b) \mapsto \int_a^b f(x)dx.$$

We will simply write ρ_c and $\rho_c \times \rho_c$ for $\rho_c|^{[0,1]}$ and $(\rho_c \times \rho_c)|^{[0,1]}$ respectively, as the co-domain will always be clear from the context.

Remark 5.1. (i) For any $f \in C([0,1])$, the modulus of continuity of $\mathbf{EVAL}(f,\cdot)$ and $\mathbf{MAX}(f,\cdot)$ in Theorem 3.18 is just the modulus of continuity of f.
(ii) Note that the space $C([0,1])/\sim_{\mathfrak{F}}$ introduced in Section 3 is isomorphic to $C([0,1])$ for each of the above functionals. In particular, Theorem 3.18 yields representations of $C([0,1])$ uniformly characterising the functionals.
(iii) We choose to represent the parameters by ρ_c rather than ρ to avoid additional second-order parameters in the time-bounds we are going to obtain. On the other hand, we choose for convenience not to 'normalise' the output and simply encode it by ρ. It is clear from Proposition 3.8 that one could perform an additional normalisation-step fairly efficiently.

By the Weierstraß Approximation Theorem, $\mathbb{PP}$ and $\mathbb{PQ}$, as defined in Section 4, are dense in $C([0,1])$; so $\nu_{\mathfrak{pp}}$ and $\nu_{\mathfrak{pq}}$ induce Cauchy representations of $C([0,1])$.

Definition 5.2. $\delta_{C([0,1]),\mathfrak{pp}}$ and $\delta_{C([0,1]),\mathfrak{pq}}$ are the Cauchy representations of $C([0,1])$ induced by $\nu_{\mathfrak{pp}}$ and $\nu_{\mathfrak{pq}}$ as defined in 4.35 respectively.

As in Section 4, the size of a $\delta_{C([0,1]),\mathfrak{pp}}$- or $\delta_{C([0,1]),\mathfrak{pq}}$-name of a function f can be expressed in terms of the size of the discrete functions approximating f. Note that unlike in the discrete case, the 'parameters' will be *functions* rather than constants.

Definition 5.3. Let $(f_n)_n$ be the image of a $\delta_{C([0,1]),\mathfrak{pp}}$- or $\delta_{C([0,1]),\mathfrak{pq}}$-name of f. We write $\Phi_f(n)$ for Φ_{f_n}, where $\Phi \in \{d, c, B, b, p\}$ (cf. Definition 4.36). We will also use the abbreviations $\dot{\Phi}_f := \Phi_f(n+1)$, $\ddot{\Phi}_f := \Phi_f(n+2)$ etc.[†]

[†]Caution: An expression like $\dot{\Phi}_f(a+b)$ is to be read as '$\Phi_f(n) \cdot (a+b)$' and not as $\Phi_f(a+b+1)$!

Remark 5.4. (i) Note that the above definition again is an abuse of notation, since it seems to suggest that the parameters only depend on f.

(ii) There exist no uniform bounds on the size of names of general continuous functions, and the size may grow arbitrarily fast, so we cannot avoid using functions (rather than constants) as parameters and will thus obtain 'genuine' second-order time-bounds. Note that this is a feature of *any* representation rendering evaluation second-order-polynomial-time computable, since the characterising representation of **EVAL** encodes a modulus of continuity, which can be arbitrarily big for arbitrary continuous functions. One can however show that every polynomial-time-computable function (in the sense that $\mathbf{EVAL}(f, \cdot)$ is computable in polynomial time) has an exponential-time computable $\delta_{C([0,1]),\mathfrak{pp}}$-name (cf. [Ko91], Theorem 8.8).

Proposition 5.5. *(i) Some modulus of continuity,*

$$C([0,1]) \ni f \rightrightarrows \mu_f \in \mathbb{N}^{\mathbb{N}},$$

$$|x-y| \leq 2^{-\mu_f(n)} \Rightarrow |f(x) - f(y)| \leq 2^{-n}$$

is $(\delta_{C([0,1]),\mathfrak{pp}}, \delta_{\mathbb{N}\to\mathbb{N}})$*- and* $(\delta_{C([0,1]),\mathfrak{pq}}, \delta_{\mathbb{N}\to\mathbb{N}})$*-computable in time*

$$\mathcal{O}(n + B_f(n+2)d_f(n+2)c_f(n+2))$$

with output-sizes $n + 2\log(d_f(n+2)+1) + c_f(n+2)$ *and* $n + 3\log(d_f(n+2)+1) + 2c_f(n+2)$ *respectively.*

(ii) Some L^1*-modulus*

$$C([0,1]) \ni f \rightrightarrows \mu_f \in \mathbb{N}^{\mathbb{N}},$$

$$|a-b| \leq 2^{-\mu_f(n)} \Rightarrow \int_a^b |f(t)|dt \leq 2^{-n}$$

is $(\delta_{C([0,1]),\mathfrak{pp}}, \delta_{\mathbb{N}\to\mathbb{N}})$*-computable in time* $\mathcal{O}(n + B_f(n+1)d_f(n+1)c_f(n+1))$ *with output-size* $n + c_f(n+1) + \log d_f(n+1) + 1$.

Proof. (i) Let $(f_n)_n$ be (the image of) a $\delta_{C([0,1]),\mathfrak{pp}}$ resp. $\delta_{C([0,1]),\mathfrak{pq}}$ name of f.Let L_n be a Lipschitz-constant of f_n. Then we have for all $x, y \in [0,1]$

$$|f(x) - f(y)| \leq L_{n+2}|x-y| + 2^{-n-1}.$$

So if $|x-y| \leq 2^{-n-1-\log L_{n+2}}$ it follows that $|f(x) - f(y)| \leq 2^{-n}$. A modulus of continuity of f is hence given by $\mu(n) := n + 1 + \log L_{n+2}$ and can be computed according to 4.43 in time $\mathcal{O}(n + B_f(n+2)d_f(n+2)c_f(n+2))$. The output-sizes are also given in 4.43.

(ii) Note that if $\| f - f_n \|_1 \leq 2^{-n}$ and $M_n \geq \| f_n \|_\infty$, then for all $a, b \in [0,1]$ we have

$$\int_a^b |f(x)|dx \leq M_n|a-b| + 2^{-n}.$$

So an L^1-modulus is given by $n + 1 + \log M_{n+1}$, which can be computed according to 4.43 in time $\mathcal{O}(n + B_f(n+1)d_f(n+1)c_f(n+1))$.

$\square$

Proposition 5.6. *(i) Evaluation*

$$C([0,1]) \times \mathbb{D} \ni (f,a) \mapsto f(a) \in \mathbb{R}$$

is $(\delta_{C([0,1]),\mathfrak{pq}} \times \nu_{\mathbb{D}}, \rho)$-computable in time

$$\mathcal{O}(B_f(\dot{n})(m+p_f(\dot{n})) + d_f(\dot{n})(b_f(\dot{n}) + md_f(\dot{n}))m + n(b_f(\dot{n}) + md_f(\dot{n}))),$$

and $(\delta_{C([0,1]),\mathfrak{pp}} \times \nu_{\mathbb{D}}, \rho)$-computable in time

$$\mathcal{O}(B_f(\dot{n})(m+p_f(\dot{n})) + d_f(\dot{n})(b_f(\dot{n}) + md_f(\dot{n}))m),$$

where $m = \ell(a)$.

(ii) Parametrised maximisation

$$C([0,1]) \times \mathbb{D}^2 \ni (f,(a,b)) \mapsto \max_{x\in[a,b]}\{f(x)\} \in \mathbb{R}$$

is $(\delta_{C([0,1]),\mathfrak{pq}} \times \nu_{\mathbb{D}} \times \nu_{\mathbb{D}}, \rho)$-computable in time

$$\mathcal{O}(\dot{B}_f\dot{d}_f^5(\dot{b}_f + \dot{d}_f)(\dot{b}_f + \dot{d}_f + n) + \dot{B}_f\dot{d}_f^4(\dot{b}_f + \dot{d}_f + n)^2 + \dot{B}_f\dot{d}_f^2(m+\dot{p}_f)^2).$$

where m is a bound on $\ell(a)$ and $\ell(b)$.

(iii) Parametrised integration

$$C([0,1]) \times \mathbb{D}^2 \ni (f,(a,b)) \mapsto \int_a^b \{f(x)\}dx \in \mathbb{R}$$

is $(\delta_{C([0,1]),\mathfrak{pp}} \times \nu_{\mathbb{D}} \times \nu_{\mathbb{D}}, \rho)$-computable in time

$$\mathcal{O}(\dot{B}_f\dot{d}_f(\dot{b}_f + n + \dot{d}_f(m+\dot{p}_f))(m+\dot{p}_f) + \dot{B}_f\dot{d}_f \log\dot{d}_f(\dot{b}_f + \log\dot{d}_f + n)).$$

where m is a bound on $\ell(a)$ and $\ell(b)$.

Proof. (i) Given a $\delta_{C([0,1]),\mathfrak{pq}}$-name of f, query the oracle for f_{n+1} and compute $f_{n+1}(a)$ up to error 2^{-n-1} according to Proposition 4.41, which can be done in time

$$\mathcal{O}(\dot{B}_f(m+\dot{p}_f) + \dot{d}_f(\dot{b}_f + m\dot{d}_f)m + n(\dot{b}_f + m\dot{d}_f)).$$

The bound given for $\delta_{C([0,1]),\mathfrak{pp}}$ follows analogously.

(ii) Given $a, b \in \mathbb{D}$ with $\ell(a), \ell(b) \leq m$, we first query the oracle for an approximation $\hat{f}$ to f up to error 2^{-n-1} and then compute $\max_{x\in[a,b]} f(x)$ up to error 2^{-n-1} according to 4.42, which can be done in time

$$\mathcal{O}(\dot{B}_f\dot{d}_f^5(\dot{b}_f + \dot{d}_f)(\dot{b}_f + \dot{d}_f + n) + \dot{B}_f\dot{d}_f^4(\dot{b}_f + \dot{d}_f + n)^2 + \dot{B}_f\dot{d}_f^2(m+\dot{p}_f)^2).$$

(iii) Given a $\nu_{\mathbb{D}} \times \nu_{\mathbb{D}}$-name of $a, b \in \mathbb{D}$, $\ell(a), \ell(b) \leq m$, we can compute the integral $\int_a^b f(x)dx$ up to error 2^{-n} by querying the oracle for an approximation to f up to error 2^{-n-1} and then computing the integral with precision 2^{-n-1}, using Proposition 4.44, in time

$$\mathcal{O}(\dot{B}_f\dot{d}_f(\dot{b}_f + n + \dot{d}_f(m+\dot{p}_f))(m+\dot{p}_f) + \dot{B}_f\dot{d}_f \log\dot{d}_f(\dot{b}_f + \log\dot{d}_f + n)).$$

□

As a corollary to the preceding propositions we obtain

Theorem 5.7. *(i) Evaluation*
$$C([0,1]) \times [0,1] \ni (f,x) \mapsto f(x)$$
is $(\delta_{C([0,1]),\mathfrak{pp}} \times \rho_c, \rho)$- and $(\delta_{C([0,1]),\mathfrak{pq}} \times \rho_c, \rho)$-computable in time
$$\mathcal{O}(\dddot{B}_f \dddot{d}_f \ddddot{c}_f + \ddot{B}_f(n+\log \dddot{d}_f + \ddddot{c}_f + \ddot{p}_f) + \ddot{d}_f \ddot{b}_f(n+\log \dddot{d}_f + \ddddot{c}_f) + \ddot{d}_f(n+\log \dddot{d}_f + \ddddot{c}_f)^2).$$
(ii) Parametrised maximisation
$$C([0,1]) \times [0,1] \times [0,1] \ni (f,a,b) \mapsto \max_{x\in[a,b]} f(x) \in \mathbb{R}$$
is $(\delta_{C([0,1]),\mathfrak{pp}} \times \rho_c \times \rho_c, \rho)$- and $(\delta_{C([0,1]),\mathfrak{pq}} \times \rho_c \times \rho_c, \rho)$-computable in time
$$\mathcal{O}(\dddot{B}_f \dddot{d}_f \ddddot{c}_f + \ddot{B}_f \ddot{d}_f^5(\ddot{b}_f+\ddot{d}_f)(\ddot{b}_f+\ddot{d}_f+n) + \ddot{B}_f \ddot{d}_f^4(\ddot{b}_f+\ddot{d}_f+n)^2 + \ddot{B}_f \ddot{d}_f^2(n+\log(\dddot{d}_f+1)+\ddddot{c}_f+\ddot{p}_f)^2).$$
(iii) Parametrised integration
$$C([0,1]) \times [0,1] \times [0,1] \ni (f,a,b) \mapsto \int_a^b f(x)dx \in \mathbb{R}$$
is $(\delta_{C([0,1]),\mathfrak{pp}} \times \rho_c \times \rho_c, \rho)$-computable in time
$$\mathcal{O}(\ddot{B}_f \ddot{d}_f(\ddot{b}_f+n+\ddot{d}_f(n+\ddot{c}_f+\log \ddot{d}_f+\ddot{p}_f))(n+\ddot{c}_f+\log \ddot{d}_f+\ddot{p}_f) + \ddot{B}_f \ddot{d}_f \log \ddot{d}_f(\ddot{b}_f+\log \ddot{d}_f+n)).$$

Proof. We only give the proof for $\delta_{C([0,1]),\mathfrak{pp}}$.

(i) We combine the algorithm given by Theorem 3.18 with Proposition 5.6. We first compute $\mu(n+1)$, where μ is a modulus of continuity for f, then query the oracle for a dyadic rational approximation ξ of x up to error $2^{-\mu(n+1)}$ and then compute $f(\xi)$ up to error 2^{-n-1}. The computation of $\mu(n+1)$ takes time
$$\mathcal{O}(n + \dddot{B}_f \dddot{d}_f \ddddot{c}_f)$$
and has output size $n + 2\log(\dddot{d}_f + 1) + \ddddot{c}_f$, so we perform an evaluation up to error 2^{-n-1} on a dyadic number of size $\mathcal{O}(n + \log \dddot{d}_f + \ddddot{c}_f)$, which according to Proposition 5.6 takes time
$$\mathcal{O}(\ddot{B}_f(n+\log \dddot{d}_f + \ddddot{c}_f + \ddot{p}_f) + \ddot{d}_f(\ddot{b}_f + (n+\log \dddot{d}_f + \ddddot{c}_f)\ddot{d}_f)(n+\log \dddot{d}_f + \ddddot{c}_f)).$$
Note that in the case of $\delta_{C([0,1]),\mathfrak{pq}}$ the additional complexity of the division step is already dominated by this.

(ii) Compute $\mu(n+1)$, where μ is a modulus of continuity of f, then query the oracle for approximations $\alpha, \beta \in \mathbb{D}$ to a, b up to error $2^{-\mu(n+1}$ and compute $\max_{x\in[\alpha,\beta]} f(x)$ up to error 2^{-n-1} using Proposition 5.6. The computation of the modulus of continuity takes time $\mathcal{O}(n + \dddot{B}_f \dddot{d}_f \ddddot{c}_f)$ and has output size $n + 2\log(\dddot{d}_f + 1) + \ddddot{c}_f$, so the parametrised maximisation takes time
$$\mathcal{O}(\ddot{B}_f \ddot{d}_f^5(\ddot{b}_f+\ddot{d}_f)(\ddot{b}_f+\ddot{d}_f+n) + \ddot{B}_f \ddot{d}_f^4(\ddot{b}_f+\ddot{d}_f+n)^2 + \ddot{B}_f \ddot{d}_f^2(n+\log(\dddot{d}_f+1)+\ddddot{c}_f+\ddot{p}_f)^2).$$

(iii) Compute $\mu(n+1)$, where μ is an L^1-modulus of f, then query the oracle for approximations $\alpha, \beta \in \mathbb{D}$ to a, b up to error $2^{-\mu(n+1}$ and compute $\int_\alpha^\beta f(x)dx$ up to error 2^{-n-1} using Proposition 5.6. The computation of the L^1-modulus takes time $\mathcal{O}(n + \ddot{B}_f \ddot{d}_f \ddot{c}_f)$ and has output size $n + \ddot{c}_f + \log \ddot{d}_f + 1$ according to Proposition 5.5, so the parametrised integration takes time
$$\mathcal{O}(\ddot{B}_f \ddot{d}_f(\ddot{b}_f+n+\ddot{d}_f(n+\ddot{c}_f+\log \ddot{d}_f+\ddot{p}_f))(n+\ddot{c}_f+\log \ddot{d}_f+\ddot{p}_f) + \ddot{B}_f \ddot{d}_f \log \ddot{d}_f(\ddot{b}_f+\log \ddot{d}_f+n)).$$

□

5.2. **Parametrised Complexity of Operators on $C([0,1])$.** We now proceed to show that the representations $\delta_{C([0,1]),\mathsf{pp}}$ and $\delta_{C([0,1]),\mathsf{pq}}$ render various elementary operations on the space $C([0,1])$ second-order polynomial time computable, thus allowing to construct names of more complicated functions from names of more elementary ones.

Theorem 5.8. *(i) Composition*

$$C([0,1]) \times \{h \in C([0,1]) \mid h([0,1]) \subseteq [0,1]\} \ni (f,g) \mapsto f \circ g \in C([0,1])$$

is $(\delta_{C([0,1]),\mathsf{pp}} \times \delta_{C([0,1]),\mathsf{pp}}, \delta_{C([0,1]),\mathsf{pp}})$*-computable in time*

$$\mathcal{O}\left(B_{\hat{g}}B_{\hat{f}}d_{\hat{g}}^5(b_{\hat{g}} + p_{\hat{f}} + \log d_{\hat{g}})^2 + B_{\hat{g}}B_{\hat{f}}Nd_{\hat{g}}^4(b_{\hat{g}} + p_{\hat{f}} + \log d_{\hat{g}} + N)\right)$$
$$+\mathcal{O}\left(B_{\hat{g}}B_{\hat{f}}Nd_{\hat{g}}^5(b_{\hat{g}} + p_{\hat{f}} + \log d_{\hat{g}}) + B_{\hat{g}}B_{\hat{f}}d_{\hat{f}}^3d_{\hat{g}}^3(d_{\hat{f}}b_{\hat{g}} + d_{\hat{f}}d_{\hat{g}} + b_{\hat{f}})^2\right),$$

where

$$N := n + c_{\hat{g}} + c_{\hat{f}} + \log d_{\hat{f}} + \log d_{\hat{g}}$$

and

$$\Phi_{\hat{g}} := \Phi_g(n + 2\log(d_f(n+4)+1) + c_f(n+4)),$$
$$\Phi_{\hat{f}} := \Phi_f(n+2)$$

and $\Phi \in \{d, c, B, b, p\}$.

(ii) Composition

$$C([0,1]) \times \{h \in C([0,1]) \mid h([0,1]) \subseteq [0,1]\} \ni (f,g) \mapsto f \circ g \in C([0,1])$$

is $(\delta_{C([0,1]),\mathsf{pq}} \times \delta_{C([0,1]),\mathsf{pq}}, \delta_{C([0,1]),\mathsf{pq}})$*-computable in time*

$$\mathcal{O}\left(B_{\hat{g}}B_{\hat{f}}d_{\hat{g}}^5(b_{\hat{g}} + p_{\hat{f}} + \log d_{\hat{g}})^2 + B_{\hat{g}}B_{\hat{f}}Nd_{\hat{g}}^4(b_{\hat{g}} + p_{\hat{f}} + \log d_{\hat{g}} + N)\right)$$
$$+\mathcal{O}\left(B_{\hat{g}}B_{\hat{f}}Nd_{\hat{g}}^5(b_{\hat{g}} + p_{\hat{f}} + \log d_{\hat{g}}) + B_{\hat{g}}B_{\hat{f}}d_{\hat{f}}^3d_{\hat{g}}^3(d_{\hat{f}}b_{\hat{g}} + d_{\hat{f}}d_{\hat{g}} + b_{\hat{f}})^2\right),$$

where

$$N := n + c_{\hat{g}} + c_{\hat{f}} + \log d_{\hat{f}} + \log d_{\hat{g}}$$

and

$$\Phi_{\hat{g}} := \Phi_g(n + 3\log(d_f(n+4)+1) + 2c_f(n+4)),$$
$$\Phi_{\hat{f}} := \Phi_f(n+2)$$

and $\Phi \in \{d, c, B, b, p\}$.

Proof. (i) Compute $\mu_f(n+2)$ for some modulus of continuity of f using 5.5. Query the oracle for an approximation $\hat{g}$ of g up to error $2^{-\mu_f(n+2)}$ and for an approximation $\hat{f}$ of f up to error 2^{-n-2}. Then compute an approximation h to $\hat{f} \circ \hat{g}$ using 4.38 up to error 2^{-n-1}. We then have for all $x \in [0,1]$:

$$|h(x) - f(g(x))| \leq |h(x) - \hat{f}(\hat{g}(x))| + |\hat{f}(\hat{g}(x)) - f(\hat{g}(x))| + |f(\hat{g}(x)) - f(g(x))| \leq 2^{-n}.$$

Since the modulus of continuity has size $n + 2\log(d_f(n+4)+1) + c_f(n+4)$, the complexity bound follows from 4.38.

(ii) idem.

□

Remark 5.9. The composition mapping defined in Theorem 5.8 actually allows for computing the general composition of two continuous functions $g\colon [0,1] \to \mathbb{R}$ and $f\colon g([0,1]) \to \mathbb{R}$ by first computing an estimate M on $\max_{x\in[0,1]} g(x)$ according to 4.43 and then putting $\hat{f}(t) := f_0(-M + 2Mt)$,where f_0 is some trivial continuation of f from $g([0,1])$ to $[-M, M]$. The function $\hat{f}$ is then in $C[0,1]$ and we may now, using Proposition 5.8, compute $\hat{f}((g(x) + M)/2M) = f(g(x))$. Note that approximating f by piecewise rational functions/polynomials on $[-M, M]$ is equivalent to approximating $\hat{f}$ by piecewise rational functions/polynomials on $[0, 1]$ - we will state this formally in 5.21.

Theorem 5.10. *Let B, d, b, p be bounds on $B_f, B_g, d_f, d_g, b_f, b_g, p_f, p_g$ respectively.*

(i) Addition,

$$C([0,1]) \times C([0,1]) \ni (f,g) \mapsto f + g \in C([0,1])$$

is $(\delta_{C([0,1]),\mathtt{pp}} \times \delta_{C([0,1]),\mathtt{pp}}, \delta_{C([0,1]),\mathtt{pp}})$-computable in time

$$\mathcal{O}(n + B(N)(d(N)b(N) + p(N))),$$

where $N := n + 1$.

(ii) Multiplication,

$$C([0,1]) \times C([0,1]) \ni (f,g) \mapsto f \cdot g \in C([0,1])$$

is $(\delta_{C([0,1]),\mathtt{pp}} \times \delta_{C([0,1]),\mathtt{pp}}, \delta_{C([0,1]),\mathtt{pp}})$-computable in time

$$\mathcal{O}(n + B(N)(d(N)^2 b(N)^2 + p(N))),$$

where $N := n + c(0) + \log d(0)$.

Proof. (i) Query the oracle for an approximation h to f and an approximation i to g, both up to error 2^{-n-1}. Then compute $h + i$ using 4.37. This takes time

$$\mathcal{O}((B_f(n{+}1){+}B_g(n{+}1))((d_f(n{+}1){+}d_g(n{+}1))(b_f(n{+}1){+}b_g(n{+}1)){+}(p_f(n{+}1){+}p_g(n{+}1)))).$$

(ii) Query the oracle for an approximation to f up to error 1 and use 4.43 to compute a bound M_f on $\log \parallel f \parallel$. Do the same for g to obtain a bound M_g. This can be done in time $\mathcal{O}(B_f(0)d_f(0)c_f(0) + B_g(0)d_g(0)c_g(0))$ and the constants have output-size $c_f(0) + \log d_f(0)$ and $c_g(0) + \log d_g(0)$ respectively. Let h be an approximation to f up to error 2^{-n-1-M_g} and i be an approximation to g up to error 2^{-n-1-M_f}. Let $\varepsilon_f(x) := h(x) - f(x)$ and $\varepsilon_g(x) := i(x) - g(x)$. Then

$$\parallel h \cdot i - f \cdot g \parallel = \parallel \varepsilon_f \cdot g + \varepsilon_g \cdot f \parallel \leq \parallel \varepsilon_f \cdot g \parallel + \parallel \varepsilon_g \cdot f \parallel \leq 2^{-n}.$$

Now, $h \cdot i$ can be computed using Theorem 4.37 in time

$$\mathcal{O}((B_f(N_f) + B_g(N_g))((d_f(N_f) + d_g(N_g))^2(b_f(N_f) + b_g(N_g))^2 + p_f(N_f) + p_g(N_g)))).$$

Where $N_f := n + M_g + 1$, $N_g := n + M_f + 1$.

□

Theorem 5.11. *Let B, d, b, p be bounds on $B_f, B_g, d_f, d_g, b_f, b_g, p_f, p_g$ respectively.*

(i) *Addition,*

$$C([0,1]) \times C([0,1]) \ni (f,g) \mapsto f+g \in C([0,1])$$

is $(\delta_{C([0,1]),\mathsf{pq}} \times \delta_{C([0,1]),\mathsf{pq}}, \delta_{C([0,1]),\mathsf{pq}})$*-computable in time*

$$\mathcal{O}(B(n+1)(d(n+1)^2 b(n+1)^2 + p(n+1))).$$

(ii) *Multiplication,*

$$C([0,1]) \times C([0,1]) \ni (f,g) \mapsto f \cdot g \in C([0,1])$$

is $(\delta_{C([0,1]),\mathsf{pq}} \times \delta_{C([0,1]),\mathsf{pq}}, \delta_{C([0,1]),\mathsf{pq}})$*-computable in time*

$$\mathcal{O}(n + B(N)(d(N)^2 b(N)^2 + p(N))),$$

where $N := n + c(0) + \log d(0)$*.*

Proof. We proceed as in 5.10, replacing the operations on polynomials by their counterparts for rational functions. The bounds given then follow immediately from 4.40 and 4.43. □

Theorem 5.12. *Division,*

$$\{g \in C([0,1]) \| g(x)| \geq 1 \text{ for all } x \in [0,1]\} \ni f \mapsto 1/f \in C([0,1])$$

is $(\delta_{C([0,1]),\mathsf{pq}}, \delta_{C([0,1]),\mathsf{pq}})$*-computable in time* $\mathcal{O}(n + B_f(n+2)d_f(n+2)b_f(n+2) + B_f(n+2)p_f(n+2))$*.*

Proof. Let $(f_n)_n$ be a $\delta_{C([0,1]),\mathsf{pq}}$-name of $f \in \{g \in C([0,1]) \| g(x)| \geq 1 \text{ for all } x \in [0,1]\}$ with breakpoints $(\vec{a}_n)_n$ and rational approximants $(p_n^k/q_n^k)_n$. By the triangle inequality we have

$$|p_n^k(x)/q_n^k(x)| \geq |f(x)| - 2^{-n} \geq 1 - 2^{-n}$$

and hence

$$|p_n^k(x)| \geq (1-2^{-n})|q_n^k(x)| \geq 1 - 2^{-n}.$$

It follows, that $p_n^k + 2^{-n}$ is minorised by 1 on its domain and

$$\begin{aligned}\left|\frac{q_{n+2}^k(x)}{p_{n+2}^k(x) + 2^{-n-2}} - \frac{1}{f(x)}\right| &\leq |q_{n+2}^k(x)| \cdot \frac{\left|f - \frac{p_{n+2}^k}{q_{n+2}^k}\right|}{|p_{n+2}^k|} + 2^{-n-2} \\ &\leq |q_{n+2}^k(x)| \frac{2^{-n-2}}{(1-2^{-n-2})|q_{n+2}^k(x)|} + 2^{-n-2} \\ &\leq 2^{-n}.\end{aligned}$$

So the sequence of piecewise rational functions with breakpoints $\vec{a}_{n+2}$ and rational approximants $p_{n+2}^k + 2^{-n-2}$ is a $\delta_{C([0,1]),\mathsf{pq}}$-name of $1/f$. It can be computed in time $\mathcal{O}(n + B_f(n+2)d_f(n+2)b_f(n+2) + B_f(n+2)p_f(n+2))$. □

We now give two additional interesting closure properties of $\mathbf{FP}_{\delta_{C([0,1]),\mathsf{pq}}}(C([0,1]))$ - case distinctions at **FP**-points of ρ and taking square roots. While the first follows from fairly standard arguments and thus will not be carried out in detail, the latter is somewhat more involved.

Theorem 5.13. *Case distinction,*

$$C([0,1]) \times C([0,1]) \times [0,1] \ni (f,g,a) \mapsto h \in C([0,1])$$

where $f(a) = g(a)$ *and*

$$h(x) = \begin{cases} f(x) & x \in [0,a] \\ g(x) & x \in [a,1] \end{cases}$$

is $(\delta_{C([0,1]),\mathtt{pp}} \times \delta_{C([0,1]),\mathtt{pp}} \times \rho_c, \delta_{C([0,1]),\mathtt{pp}})$*- and* $(\delta_{C([0,1]),\mathtt{pq}} \times \delta_{C([0,1]),\mathtt{pq}} \times \rho_c, \delta_{C([0,1]),\mathtt{pq}})$-$\mathbf{FP}^2$.

Theorem 5.14. *The function* $x \mapsto \sqrt{x}$ *has a* **POLYTIME**-*computable* $\delta_{C^0,\mathtt{pq}}$*-name.*

This result can also be found in [LHE01], although it is formulated in a somewhat different framework. The proof is essentially an analysis of the computational complexity of *Newman's Theorem* a classic result in approximation theory.

Theorem 5.15 (Newman, [New64])**.** *Let* $n \geq 4$, $\xi_n = e^{-\frac{1}{\sqrt{n}}}$ *and*

$$p_n(x) := \prod_{k=0}^{n-1} (x + \xi^k),$$

$$r_n(x) := x \frac{p_n(x) - p_n(-x)}{p_n(x) + p_n(-x)}.$$

Then for all $x \in [-1,1]$ *we have*

$$||x| - r(x)| \leq 3e^{-\sqrt{n}}$$

So essentially, all we need to verify in order to establish 5.14 is that in the above theorem, the denominators are "well-behaved" and that we can approximate the numbers ξ_n efficiently.

The following proposition can be easily verified using the power series of the exponential function.

Proposition 5.16. *Given a* $\nu_{\mathbb{N}} \times \nu_1$*-name of* $(n,m) \in \mathbb{N} \times \mathbb{N}$ *we can compute* $e^{-\frac{1}{n}}$ *up to precision* 2^{-m} *in time polynomial in* $\log n$ *and* m*.*

Proof of Theorem 5.14. Picking up the notation of 5.15, we define the even polynomials $P_n(x^2) := x(p_n(x) - p_n(-x))$ and $Q_n(x^2) := p_n(x) + p_n(-x)$. Note that all the coefficients of $Q_n(x)$ are positive, so for all $x \in [0,1]$ we have

$$Q_{n^2}(x) \geq Q_{n^2}(0) = 2 \prod_{k=0}^{n^2-1} e^{-\frac{k}{n}} = 2e^{-(n^3-n)/2} \geq 2 \cdot 3^{-(n^3-n)/2}$$

Define $\hat{P}_{n^2} := 3^{(n^3-n)/2} P_{n^2}$, $\hat{Q}_{n^2} := 3^{(n^3-n)/2} Q_{n^2}$ and consider the sequence $(\hat{P}_{n^2}/\hat{Q}_{n^2})_{n \geq 6}$. By Newman's Theorem, we have for all $x \in [0,1]$:

$$|\frac{\hat{P}_{n^2}(x)}{\hat{Q}_{n^2}(x)} - \sqrt{x}| \leq 3e^{-n} \leq 2^{-n}$$

and $\hat{Q}_{n^2}$ is minorised by 1 on $[0,1]$. It remains to verify that, given $n \in \mathbb{N}$ we can compute some $\nu_{\mathfrak{pq}}$-name of $\frac{\hat{P}_{n^2}}{\hat{Q}_{n^2}}$ in time polynomial in n. Using Proposition 5.16, we can compute $c \leq e^{-1/n}$ with

$$|c - e^{-1/n}| \leq 2^{-n^2-n-4\log n-1}$$

in time polynomial in n. For $a, b \in [-1,1]$, $k \in \mathbb{N}$ we have by the Mean Value Theorem

$$|a^k - b^k| \leq k|a-b|$$

So for $k \leq n^2$

$$|c^k - e^{-k/n}| \leq 2^{-n^2-n-2\log n-1}.$$

Let $\varepsilon_k := c^k - e^{-k/n}$. Then

$$\begin{aligned}
|\prod_{k=0}^{n^2-1}(x+\xi_n^k) - \prod_{k=0}^{n^2-1}(x+c^k)| &= |\prod_{k=0}^{n^2-1}(x-\xi_n^k) - \prod_{k=0}^{n^2-1}(x+\xi^k+\varepsilon_k)| \\
&= |\sum_{k=0}^{n^2-1}\varepsilon_k \prod_{j<k}(x+\xi^j)\prod_{j>k}(x+\xi^j+\varepsilon_j)| \\
&\leq 2^{-n^2-n-2\log n-1}(n^2-1)2^{n^2-1} \\
&\leq 2^{-n-1}.
\end{aligned}$$

And $\tilde{p}_n(x) = \prod_{k=0}^{n^2-1}(x+c^k)$ can be computed in time polynomial in n using 4.9. Put

$$\hat{p}_n := 3^{(n^3-n)/2}\tilde{p}_n.$$

Then $\hat{p}_n$ can also be computed in time polynomial in n and we have for all $x \in [0,1]$

$$|x(\hat{p}_n(x) - \hat{p}_n(-x)) - \hat{P}_n^2(x^2)| \leq 2^{-n}$$

and

$$|\hat{p}_n(x) + \hat{p}_n(-x) - \hat{Q}_n^2(x^2)| \leq 2^{-n}.$$

So $\sqrt{x}(\hat{p}_n(\sqrt{x}) - \hat{p}_n(-\sqrt{x}))$ and $\hat{p}_n(\sqrt{x}) + \hat{p}_n(-\sqrt{x})$ define a $\nu_{\mathbb{D}_m(x)}$-name of $\frac{\hat{P}_{n^2}}{\hat{Q}_{n^2}}$, which is uniformly **POLYTIME** computable in n. □

Corollary 5.17. *Taking square roots,*

$$\{g \in C([0,1]) \mid g(x) \geq 0 \textit{ for all } x \in [0,1]\} \ni f \mapsto \sqrt{f} \in C([0,1]),$$

is $(\delta_{C([0,1]),\mathfrak{pq}}, \delta_{C([0,1]),\mathfrak{pq}})$-$\mathbf{FP}^2$.

Proof. According to Proposition 4.43 we can compute $M \in \mathbb{N}$ such that $f \leq 2^M$ throughout $[0,1]$ in second order polynomial time. Then $f/2^{2M}$ maps $[0,1]$ into $[0,1]$, so by Theorem 5.8 and Theorem 5.14 we can compute a $\delta_{C([0,1]),\mathfrak{pq}}$-name of $g := \sqrt{f/2^{2M}}$ in second order polynomial time. It follows, that $\sqrt{f} = 2^M g$ is $(\delta_{C([0,1]),\mathfrak{pq}}, \delta_{C([0,1]),\mathfrak{pq}})$-computable in second-order polynomial time. □

5.3. **Robustness of Definitions and Comparison of the Representations Introduced.** We are now going to study the relationship between the various representations introduced with respect to the reduction-relation defined in Section 2 (Definition 2.10). We will also justify some of the choices we made when defining the representations.

It seems natural to ask whether one really has to use *piecewise* polynomial- and rational functions as discrete approximants to obtain a sensible Cauchy representation and whether it would not suffice to represent a function by a globally fast converging sequence of polynomials or rational functions. Let $\delta_{C([0,1]),\mathfrak{p}}$ denote the Cauchy representation induced by $\nu_{\mathbb{D}[x]}$ and $\delta_{C([0,1]),\mathfrak{q}}$ denote the Cauchy representation induced by $\nu_{\mathbb{D}_m(x)}$. It is well known that $\delta_{C([0,1]),\mathfrak{p}}$ already fails to have a polynomial-time computable (or even **PSIZE**) name for very simple functions and thus is a too weak representation - in fact, by the Jackson-Bernstein Theorem, only infinitely many times differentiable functions can have a **PSIZE**-$\delta_{C([0,1]),\mathfrak{p}}$ name (cf. Corollary 8.11 in [Ko91]; see also [LHE01] for a more detailed discussion). We will give a simple example based on *Markov's inequality*, a fundamental inequality in approximation theory we are also going to need in Section 6. For a proof see e.g. [Che66].

Theorem 5.18 (Markov's Inequality)**.** *Let P be a polynomial of degree d. Then*

$$\max_{x\in[-1,1]} |P'(x)| \leq 2d^2 \max_{x\in[-1,1]} |P(x)|.$$

Proposition 5.19. *Let P be a polynomial of degree d. Then for all $a, b \in \mathbb{R}$ we have*

$$\max_{x\in[a,b]} |P^{(k)}(x)| \leq (b-a)^{-k}(2d)^{2k} \max_{x\in[a,b]} |P(x)|.$$

Proposition 5.20. *The function $x \mapsto |x - 1/2|$ does not have a* **PSIZE** *$\delta_{C([0,1]),\mathfrak{p}}$-name.*

Proof. Let $(P_n)_n$ be a sequence of polynomials in $\mathbb{D}[x]$ with degrees $(d_n)_n$ satisfying

$$|P_n(x) - |x-1/2|| \leq 2^{-n} \text{ for all } x \in [0,1].$$

Then

$$\begin{cases} |P_n(x) + x - 1/2| \leq 2^{-n} & \text{for all } x \in [0,1/2] \\ |P_n(x) - x + 1/2| \leq 2^{-n} & \text{for all } x \in [1/2,1] \end{cases}.$$

So by 5.19

$$\begin{cases} |P_n'(x) + 1| \leq \frac{8d_n^2}{2^n} & \text{for all } x \in [0,1/2] \\ |P_n'(x) - 1| \leq \frac{8d_n^2}{2^n} & \text{for all } x \in [1/2,1] \end{cases}.$$

in particular

$$\begin{cases} |P_n'(1/2) + 1| \leq \frac{8d_n^2}{2^n} \\ |P_n'(1/2) - 1| \leq \frac{8d_n^2}{2^n} \end{cases}.$$

Suppose that $(d_n)_n$ is uniformly bounded by some polynomial $Q \in \mathbb{N}[x]$. Then

$$\frac{8d_n^2}{2^n} \leq \frac{8Q(n)^2}{2^n} \to 0 \text{ as } n \to \infty$$

so $P_n'(1/2) \to 1$ and $P_n'(1/2) \to -1$. Contradiction. □

On the other hand, global approximation by rational functions is equivalent to approximation by piecewise rational functions. This is a consequence of Newman's Theorem (Theorem 5.15), which entails that the function-family

$$h_n(x) = \begin{cases} 0 & -1 \leq x \leq -1/2 - 2^{-n} \\ 2^n(x + \frac{1}{2} + 2^{-n}) & -1/2 - 2^{-n} \leq x \leq -1/2 \\ 1 & -1/2 \leq x \leq 1/2 \\ 2^n(\frac{1}{2} + 2^{-n} - x) & 1/2 \leq x \leq 1/2 + 2^{-n} \\ 0 & 1/2 + 2^{-n} \leq x \leq 1 \end{cases}$$

can be approximated with exponentially small error by a single rational function on $[-1, 1]$. One can then use this to 'simulate' the characteristic function of an interval and construct piecewise rational functions by superposition. For details see [LHE01], Théorème 5.3.2 (cf. also [Ko91], Lemma 8.12) (the framework used there is different from ours, but the proof given carries over easily).

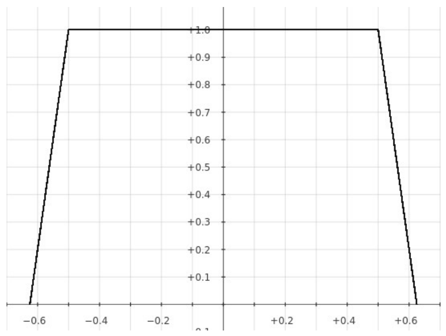

FIGURE 1. The function h_3, approaching the characteristic function of $[-\frac{1}{2}, \frac{1}{2}]$.

Finally, we remark that the choice of the interval $[0, 1]$ was essentially arbitrary.

Proposition 5.21. *Let $a, b \in \mathbb{R}$ and let $\delta_{C[a,b],\mathfrak{pq}}$ denote the Cauchy representation of $C[a, b]$ induced by the notation of the set of piecewise rational functions with dyadic rational coefficients and dyadic rational breakpoints(the endpoints of the interval being given as ρ-names), defined analogously to 4.35. Then the mapping*

$$C[a, b] \ni f(x) \mapsto f(a + t(b - a)) \in C[0, 1]$$

is $(\delta_{C[a,b],\mathfrak{pq}}, \delta_{C([0,1]),\mathfrak{pq}})$-computable in second-order polynomial time and an analogous result holds for the analogue of $\delta_{C([0,1]),\mathfrak{pp}}$.

Remark 5.22. In the above Proposition it is crucial that the endpoints of the interval are also encoded in the representation, since otherwise the result would depend on the computability of a and b.

We now compare the representations introduced so far. Let $\delta_{C([0,1]),e}$, $\delta_{C([0,1]),m}$, $\delta_{C([0,1]),i}$ denote the representations uniformly characterising evaluation, maximisation and integration as defined in Section 5.1 respectively. It follows immediately from Theorem 5.7 and Theorem 3.18 that $\delta_{C([0,1]),\mathfrak{pq}} \preceq \delta_{C([0,1]),m}$ and that $\delta_{C([0,1]),\mathfrak{pp}} \preceq \delta_{C([0,1]),i}$. It is equally obvious that $\delta_{C([0,1]),m} \preceq \delta_{C([0,1]),e}$.

Proposition 5.23. *The reduction $\delta_{C([0,1]),m} \preceq \delta_{C([0,1]),e}$ is strict.*

Proof. We will show that the functional **MAX** is not $(\delta_{C([0,1]),e} \times \rho, \rho)$-$\mathbf{FP}^2$. Suppose there exists some Oracle-Turing-machine M computing a $(\delta_{C([0,1]),e} \times \rho \times \rho, \rho)$-$\mathbf{FP}^2$-realiser of **MAX**, whose running time is bounded by some second-order polynomial $P(x, l)$. Let n be such that $2^n \geq P(n+3, id_{\mathbb{N}} + 3)$. Consider the family of functions

$$h_{n,k}(x) = \begin{cases} 0 & x \in [0, \frac{k}{2^n}] \cup [\frac{k+1}{2^n}, 1] \\ (x - \frac{k}{2^n}) & x \in [\frac{k}{2^n}, \frac{2k+1}{2^{n+1}}] \\ (\frac{k+1}{2^n} - x) & x \in [\frac{2k+1}{2^{n+1}}, \frac{k+1}{2^n}] \end{cases}$$

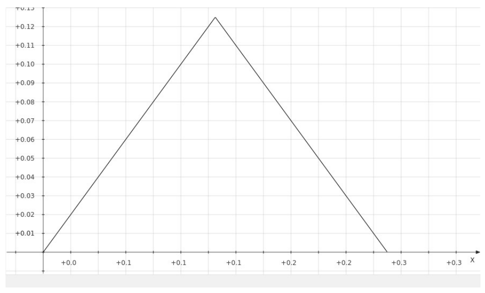

FIGURE 2. The function $h_{3,0}$. It is thin, but not steep.

Note that each $h_{n,k}$ has a $\delta_{C([0,1]),e}$-name of size $id_{\mathbb{N}} + 3$, since $id_{\mathbb{N}}$ is a modulus of continuity for $h_{n,k}$ and the evaluations on discrete sample-points are encoded by ρ-names which have size at most $id_{\mathbb{N}}+3$. Now, given such a $\delta_{C([0,1]),e}$-name of $h_{n,k}$ and 1^{n+3}, $n \in \mathbb{N}$, the machine M will only query the oracle $P(n+3, id_{\mathbb{N}})$ times, so there is at least one function h_{n,k_0} such that the result of all the queries issued by the machine for evaluations on dyadic numbers is zero. So the machine will have to produce the same output on input $(h_{n,k_0}, 0, 1, 1^{n+3})$ as on input $(\lambda(x).0, 0, 1, 1^{n+3})$, where the zero-function $\lambda(x).0$ is given with modulus of continuity $id_{\mathbb{N}} + 3$. However, $\mathbf{MAX}(h_{n,k_0}, 0, 1) - 2^{-n-3} \geq 2^{-n-3}$. Contradiction. ☐

Remark 5.24. (i) A somewhat stronger version of the above Proposition has already been proved in [KMRZ12].

(ii) A similar argument can be used to show that there exists no Cauchy representation of $C([0,1])$ uniformly characterising **EVAL** (i.e. that $\delta_{C([0,1]),e}$ is not a Cauchy representation). Using Theorem 2.22 in [Ko91] one can easily construct Cauchy representations of $C([0,1])$ non-uniformly characterising **EVAL** (for examples, see [LHE01]). This shows that in general Theorem 3.20 does not admit a converse.

Proposition 5.25. *The reduction $\delta_{C([0,1]),\mathsf{pq}} \preceq \delta_{C([0,1]),m}$ is strict.*

Proof. Consider the function-family

$$f_m(x) := \frac{sin(2^m \pi x)}{2^m \pi}.$$

Note that there exists a family of $\delta_{C([0,1]),m}$-names of $(f_m)_m$ with size uniformly bounded by $n+3$, since each of the functions has modulus of continuity $id_{\mathbb{N}}$. Consequently, if $id_{C([0,1])}$ were $(\delta_{C([0,1]),m}, \delta_{C([0,1]),\mathfrak{pq}})$-$\mathbf{FP}^2$, then the family would have a family of $\delta_{C([0,1]),\mathfrak{pq}}$-names with size uniformly bounded by some polynomial P. However, any continuous function uniformly approximating f_m up to error 2^{-m-1} has at least $2^m - 1$ zeroes in $[0,1]$ and a piecewise rational function with M zeroes B_f breakpoints and degree d_f satisfies $B_f d_f \geq M$. This implies $P(n+1) \geq 2^n - 1$ for all n. Contradiction. □

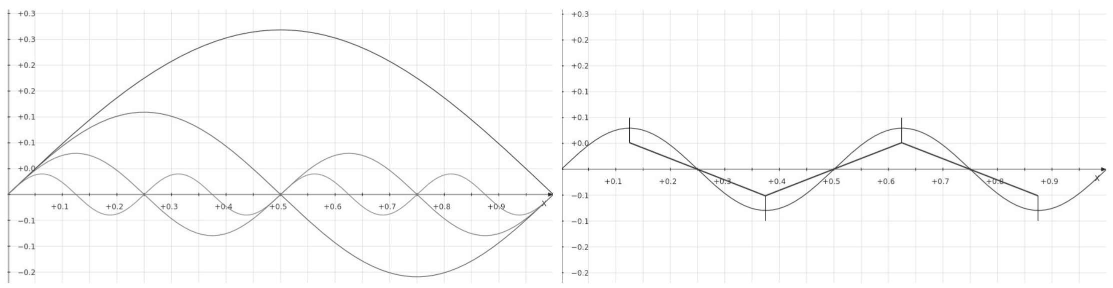

FIGURE 3. Right: The functions f_1 to f_4 employed in the proof of Proposition 5.25. Left: Any continuous approximation to f_m up to error 2^{-m-1} has at least $2^m - 1$ zeroes.

Proposition 5.26. *Evaluation*

$$C([0,1]) \times \mathbb{R} \ni (f,x) \mapsto f(x)$$

is not $(\delta_{C([0,1]),i} \times \rho_c, \rho)$-computable.

Proof. Suppose there exists a machine M computing a $(\delta_{C([0,1]),i} \times \rho_c, \rho)$-realiser of **EVAL**. Let $\psi \in \mathfrak{Reg}$ be a $\delta_{C([0,1]),i}$-name of the zero function $\lambda x.0$, consisting of a $\delta_{\mathbb{N}\to\mathbb{N}}$-name of $id_{\mathbb{N}}$, encoding an L^1-modulus of $\lambda x.0$, and the ρ-name of 0 given by the sequence $(2^{-n})_n \in \mathbb{D}^\omega$ for each $a, b \in \mathbb{D}$. Given ψ and a $\nu_{\mathbb{D}}^{\omega}$-name of the constant sequence $(\frac{1}{2})_n$ as an oracle and 1 as input, the machine M outputs an approximation to 0 up to error $\frac{1}{2}$ after a finite number of steps $T(n)$. In particular, the machine only queries the oracle for approximations to integrals with dyadic endpoints $\int_a^b f(x)dx$ up to error $2^{-T(n)}$. Now, let $a := \frac{1}{2} - 2^{-T(n)-1}$, $b := \frac{1}{2} + 2^{-T(n)-1}$ and consider the function

$$h(x) := \begin{cases} 0 & x \in [0,a] \\ \frac{x-a}{\frac{1}{2}-a} & x \in [a, \frac{1}{2}] \\ \frac{b-x}{b-\frac{1}{2}} & x \in [\frac{1}{2}, b] \\ 0 & x \in [b,1] \end{cases}$$

Note that $\int_0^1 h(x)dx = 2^{-T(n)-1}$, so there exists a name φ of h which coincides with ψ on all inputs of length smaller than $T(n)$. The machine M will consequently yield the same output as for the above input if one replaces ψ by φ. However, $h(\frac{1}{2}) = 1 > \frac{1}{2}$. Contradiction. □

The above proposition illustrates that the representation $\delta_{C([0,1]),i}$ only encodes 'L^1-information' of a function and would be more naturally defined as a mapping with co-domain L^1.

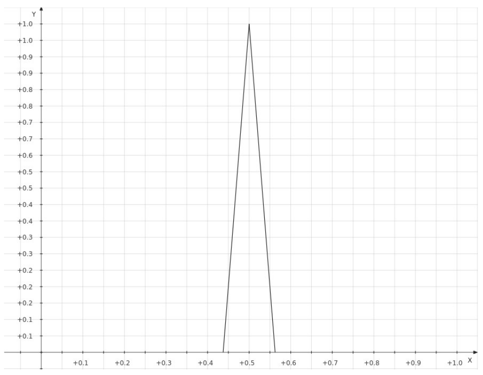

FIGURE 4. The function h employed in the proof of Proposition 5.26. It is very thin and very steep.

Corollary 5.27. *The reduction $\delta_{C([0,1]),\mathsf{pp}} \preceq \delta_{C([0,1]),i}$ is strict.*

The situation can be summarised in the following diagram, where an arrow represents reducibility and an equality-sign represents equivalence

$$\begin{tikzcd}
 & \delta_{C([0,1]),i} & & & & \\
\delta_{C[a,b],\mathsf{p}} \arrow[r] \arrow[d, equal] & \delta_{C([0,1]),\mathsf{pp}} \arrow[r] \arrow[u] \arrow[d, equal] & \delta_{C([0,1]),\mathsf{pq}} \arrow[r, equal] \arrow[d, equal] & \delta_{C([0,1]),\mathsf{q}} \arrow[r] \arrow[d, equal] & \delta_{C([0,1]),m} \arrow[r] & \delta_{C([0,1]),e} \\
\delta_{C[a,b],\mathsf{p}} \arrow[r] & \delta_{C[a,b],\mathsf{pp}} \arrow[r] & \delta_{C[a,b],\mathsf{pq}} \arrow[r, equal] & \delta_{C[a,b],\mathsf{q}} & &
\end{tikzcd}$$

none of the arrows reverses unless indicated, except perhaps the one joining $\delta_{C([0,1]),\mathsf{pp}}$ with $\delta_{C([0,1]),\mathsf{pq}}$ and the one joining $\delta_{C[a,b],\mathsf{pp}}$ with $\delta_{C[a,b],\mathsf{pq}}$. Note that also the relationship between $\delta_{C([0,1]),i}$ and $\delta_{C([0,1]),\mathsf{pq}}$ is unknown.

We conclude this section with the remark, that the question of whether the representations $\delta_{C([0,1]),\mathsf{pp}}$ and $\delta_{C([0,1]),\mathsf{pq}}$ are equivalent is directly tied to the $(\delta_{C([0,1]),\mathsf{pp}}, \delta_{C([0,1]),\mathsf{pp}})$-complexity of division.

Proposition 5.28. *The following are equivalent*

(i) $\delta_{C([0,1]),\mathsf{pp}} \equiv \delta_{C([0,1]),\mathsf{pq}}$
(ii) Division, as in Theorem 5.11 is $(\delta_{C([0,1]),\mathsf{pp}}, \delta_{C([0,1]),\mathsf{pp}})$-computable in second order polynomial time.

Proof. (i) $\Rightarrow$ (ii) follows immediately from Theorem 5.11.

Suppose now that (ii) holds. Let $P(x,l)$ be a second-order polynomial bounding the running time of some $(\delta_{C([0,1]),\mathsf{pp}}, \delta_{C([0,1]),\mathsf{pp}})$-realiser of division. Let $f \in C([0,1])$ and let $(f_n)_n$ be (the image of) some $\delta_{C([0,1]),\mathsf{pq}}$-name of f. Let $(g_n)_n$ and $(h_n)_n$ be the sequences of piecewise polynomials defined by the equation $f_n = g_n/h_n$, given as ν_{pp}-names respectively. Let $M(g_n)$ and $M(h_n)$ be constants majorising the logarithm of g_n and h_n respectively. They can be computed in second-order polynomial time according to Proposition 4.43. Given the constant sequence $(h_n)_k$ we can compute some piecewise polynomial function $\hat{h}_n$ satisfying

$$\| \hat{h}_n - 1/h_n \| \leq 2^{-n-M(g_n)-M(h_n)}$$

in time $P(n + M(g_n) + M(h_n), |h_n|)$. Using 4.37 we can then compute $g_n\hat{h}_n$ in time uniformly bounded in $\ell(h_n)$, $\ell(g_n)$ and n. Let $\varepsilon_n(x) = \frac{1}{h_n(x)} - \hat{h}_n(x)$. Then for all $x \in [0,1]$ we have

$$\begin{aligned}
|\frac{g_n(x)}{h_n(x)} - g_n(x)\hat{h}_n(x)| &= \left| g_n(x) \left(\frac{1 - \hat{h}_n(x)h_n(x)}{h_n(x)} \right) \right| \\
&\leq |g_n(x)(1 - \hat{h}_n(x)h_n(x))| \\
&= |g_n(x)(1 - (\frac{1}{h_n(x)} - \varepsilon_n(x))h_n(x))| \\
&= |g_n(x)h_n(x)\varepsilon_n(x)| \\
&\leq 2^{-n}.
\end{aligned}$$

So the sequence $(g_n\hat{h}_n)_n$ is a $\delta_{C([0,1]),\mathfrak{pp}}$-name of f, which can be computed in time uniformly bounded in $\ell(h_n)$, $\ell(g_n)$ and n. □

Proposition 5.29. *The following are equivalent*

(i) $\delta_{C([0,1]),\mathfrak{pp}} \equiv \delta_{C([0,1]),\mathfrak{pq}}$

(ii) The sequence $(f_m)_m$ *with*

$$f_m(x) = \begin{cases} 2^m & \textit{if } x \leq 2^{-m} \\ \frac{1}{x} & \textit{otherwise} \end{cases}$$

is an **FP***-point of* $\delta^{\omega}_{C([0,1]),\mathfrak{pp}}$.

(iii) Some sequence $(g_m)_m \in \mathbb{PP}^{\omega}$ *satisfying* $\| g_m - f_m \|_\infty \leq C$ *for all* m *and some* $C < 1$ *(where* $(f_m)_m$ *is defined as in (ii)) is an* **FP***-point of* $\nu^{\omega}_{\mathfrak{pp}}$.

Proof. It is clear that (i) implies (ii) since the sequence $(f_m)_m$ is clearly an **FP**-point of $\delta^{\omega}_{C([0,1]),\mathfrak{pq}}$. Suppose now, that (ii) holds. We show that under this assumption division is $(\delta_{C([0,1]),\mathfrak{pp}}, \delta_{C([0,1]),\mathfrak{pp}})$-$\mathbf{FP}^2$. Let $(g_n)_n$ be (the image of) some $\delta_{C([0,1]),\mathfrak{pp}}$-name of $g \in C([0,1])$ with $g(x) \geq 1$ on $[0,1]$. We query the oracle for g_0 and use 4.43 to compute a bound M on $\log \| g_0 + 1 \|$. Now, using 5.8 and 5.10, we can compute the sequence $(2^M(f_M \circ (2^{-M}g_n)))_n$, which is a $\delta_{C([0,1]),\mathfrak{pp}}$-name of $1/g$ in second order polynomial time. It is also clear, that (ii) implies (iii). Suppose that (iii) holds. We use the approximations to $(f_m)_m$ given by the sequence $(g_m)_m$ as a sequence of initial values for the Newton-Raphson division algorithm. Note that $f_m = 1/h_m$, where

$$h_m(x) := \begin{cases} 2^{-m} & x \leq 2^{-m} \\ x & \text{otherwise} \end{cases}.$$

Let $T_m \colon C([0,1]) \to C([0,1]), T_m f := 2f - h_m f^2$. Note that T maps $\mathbb{PP}$ into $\mathbb{PP}$. For all $f \in C([0,1])$ we have

$$\| Tf - f_m \|_\infty = \| 2f - h_m f^2 - 1/h_m \|_\infty \leq \| h_m \|_\infty \| f - 1/h_m \|^2_\infty = \| f - 1/h_m \|^2_\infty .$$

So the fix-point iteration $x_0 := g_m$, $x_{i+1} := Tx_i$ will yield an approximation to f_m up to error 2^{-n} in logarithmically many steps. Consequently, $(f_m)_m$ is an **FP**-point of $\delta^{\omega}_{C([0,1]),\mathfrak{pp}}$. $\square$

6. Representations of the Space of Real Analytic Functions

We are now going to study the functionals introduced in Section 5 on the space $C^\omega([0,1])$ of real analytic functions over $[0,1]$, that is, real-valued functions on $[0,1]$ admitting a holomorphic extension to some open neighbourhood of $[0,1]$ in the complex plane. In [KMRZ12] it is shown that, unlike in the case of continuous functions, there exist representations of $C^\omega([0,1])$ rendering **MAX**, **INT** and **EVAL** second-order polynomial time computable such that every function has a linear-size name. Consequently, the size of such a name can be expressed using *constant* parameters (i.e. parameters that only depend on the represented function and not on the input-length) and bounds on the time-complexity of operators on $C^\omega([0,1])$ can be expressed as usual (parametrised) first-order bounds. We give a representation of $C^\omega([0,1])$ similar to those studied in [KMRZ12] but closer to the representation $\delta_{C([0,1]),\mathtt{pp}}$ introduced in Section 5 and give quantitative bounds on the complexity of integration, evaluation, maximisation and some elementary operations on $C^\omega([0,1])$.

Definition 6.1. (i) A $\delta_{C^\omega([0,1])}$-name of $f \in C^\omega([0,1])$ is a $\rho_c^* \times (\rho_c^\omega)^* \times \nu_{\mathbb{N}}^* \times \nu_1^*$-name of

$$(\vec{x}, \vec{a}, \vec{A}, \vec{L}) \in \mathbb{R}^* \times (\mathbb{R}^\omega)^* \times \mathbb{N} \times \mathbb{N},$$

with $|\vec{x}| = |\vec{a}| = |\vec{L}| = |\vec{A}| =: B$ such that

$$[0,1] \subseteq \bigcup_{m=1}^{B} [x_m - \tfrac{1}{4L_m}, x_m + \tfrac{1}{4L_m}]$$

and $f^{(j)}(x_m) = a_{m,j} \cdot j!$ and $|a_{m,j}| \leq A_m \cdot L_m^j$. We furthermore require $L_m \geq 2$ and $x_1 \leq x_2 \leq \cdots \leq x_B$. Also, $[x_i, x_i + \frac{1}{2L_i}] \cap [x_{i+1} - \frac{1}{2L_{i+1}}, x_{i+1}]$ must have diameter at least $\frac{1}{4L_i} + \frac{1}{4L_{i+1}}$.

(ii) A $\delta_{C^\omega,\mathtt{pp}}$ of $f \in C^\omega([0,1])$ is a $\nu_{\mathbb{D}}^* \times (\nu_{\mathbb{D}[x]}^*)^\omega \times \nu_{\mathbb{N}} \times \nu_1$-name of $(\vec{a}, \vec{P}_n, A, L) \in \mathbb{D}^* \times (\mathbb{D}[x]^*)^\omega \times \mathbb{N} \times \mathbb{N}$ so that

$$0 = a_1 < a_2 < \cdots < a_{B_f} = 1,\ a_{k+1} - a_k \geq \tfrac{1}{L} \text{ for all } k$$

$$|P_{i,n}(x) - f(x)| \leq 2^{-n} \text{ for all } x \in [a_i, a_{i+1}],$$

and

$$f^{(k)}(x) \leq A \cdot L^k \cdot k! \text{ for all } x \in [0,1]. \tag{1}$$

Furthermore, we require that for all i, n, we have $\deg P_{i,n} \leq d_f \cdot (n + \log A + 1)$ for some constant $d_f \in \mathbb{N}$ and $b_{P_{i,n}} = b_f \cdot (n + \log A) \log L$ for some constant $b_f \in \mathbb{N}$.

Remark 6.2. $\delta_{C^\omega([0,1])}$ is a slightly adapted version of the representation α given in [KMRZ12]. We have made some inessential additional requirements to avoid certain trivial case-distinctions in our further discussion. The representation $\delta_{C^\omega,\mathtt{pp}}$ is essentially the representation $\delta_{C([0,1]),\mathtt{pp}}$ from Section 5, enriched with information on the differentiability of the represented function f. Note that unlike in the case of $\delta_{C([0,1]),\mathtt{pp}}$, we do not require the piecewise-polynomial functions approximating f to be continuous. In the case of $\delta_{C([0,1]),\mathtt{pp}}$ this requirement came essentially with no loss of generality, since, given a discontinuous approximation by piecewise polynomial functions, one could always introduce 'joints' on sufficiently small intervals around the breakpoints to make the approximation continuous (so long as the number of breakpoints would grow sub-exponentially in n). However, in the case of $\delta_{C^\omega,\mathtt{pp}}$ we need the polynomials to approximate f on intervals of a size controlled by the parameters, so that their derivatives approximate the derivatives of f in a rate controlled by the parameters. Apart from these differences, the algorithms on $\delta_{C([0,1]),\mathtt{pp}}$-names found in Section 5 can be easily adapted to apply to $\delta_{C^\omega,\mathtt{pp}}$-names. Another reason for choosing continuous approximants in the continuous case is that the approximating functions are then Lipschitz continuous, which allows for an efficient computation of the modulus of continuity of a represented function. Here, the logarithm of a Lipschitz-constant for f represented by a $\delta_{C^\omega,\mathtt{pp}}$-name is given by $\log A_f + \log L_f$ and $\max|f|$ on $[0,1]$ is bounded by A_f - these estimates are somewhat better than the more general ones found in Proposition 4.43. Note that approximations by discontinuous polynomials are also used in the discontinuous Galerkin method in numerical analysis.

Again, the size of $\delta_{C^\omega,\mathtt{pp}}$- and $\delta_{C^\omega([0,1])}$-names is controlled by certain parameters. In the case of $\delta_{C^\omega([0,1])}$ these are $\log A_f$, L_f and B_f, in the case of $\delta_{C^\omega,\mathtt{pp}}$ these are $\log A_f$, L_f, d_f, $\log b_f$, B_f and p_f, where p_f is a bound on the size of $a_1,\dots,a_{B_f}$. So far it is not obvious at all that every function in $C^\omega([0,1])$ can be represented by a $\delta_{C^\omega,\mathtt{pp}}$-name. This will be the main result of this section.

Theorem 6.3. *$\delta_{C^\omega,\mathtt{pp}}$ is a representation and the identity $C^\omega([0,1]) \to C^\omega([0,1])$ is $(\delta_{C^\omega([0,1])},\delta_{C^\omega,\mathtt{pp}})$-computable in time*

$$\mathcal{O}(B(n+\log A)^3(n+\log A+L)^2),$$

where

$$A = \max_{k=0,\dots,B} A_k,\ L = \max_{k=0,\dots,B} L_k.$$

The piecewise polynomial function computed has $B+2$ breakpoints of size $\mathcal{O}(\log L)$, the polynomials have degrees bounded by $n+\log A+1$ and coefficient-size $\mathcal{O}((n+\log A)\log L)$.

Proof. Let $f \in C^\omega([0,1])$, $n \in \mathbb{N}$ and let a $\delta_{C^\omega([0,1])}$-name of f be given by $x_1,\dots,x_B$, $A_1,\dots,A_B$, $L_1,\dots,L_B$ and $(a_{i,j})_{i=1,\dots,B,j\in\mathbb{N}}$. We construct an approximation to f by piecewise polynomials up to error 2^{-n}. Let $i < B$. By assumption, the interval

$$[x_i, x_i + \frac{1}{2L_i}] \cap [x_{i+1} - \frac{1}{2L_{i+1}}, x_{i+1}]$$

has diameter at least $\frac{1}{4L_i} + \frac{1}{4L_{i+1}}$ and contains $x_i + \frac{1}{4L_i}$. Let $c_i \in \mathbb{D}$ be an approximation to $x_i + \frac{1}{4L_i}$ up to error $\min\{\frac{1}{8L_i}, \frac{1}{8L_{i+1}}\}$. Let $c_0 := 0$ and $c_B := 1$. Note that for all $i \in \{0,\dots,B-1\}$

$$[c_i, c_{i+1}] \subseteq [x_{i+1} - \frac{1}{2L_{i+1}}, x_{i+1} + \frac{1}{2L_{i+1}}].$$

Put $N_i := n + \log A_{i+1} + 2$ and

$$P_i := \sum_{j=0}^{N_i} a_{i+1,j}(x - x_{i+1})^j.$$

Then for all $x \in [c_i, c_{i+1}]$

$$|f(x) - P_i(x)| \le 2^{-n-1}.$$

Now, let $C_i := \log A_{i+1} + \log L_{i+1} + 3$. Note that 2^{C_i} is a Lipschitz-constant for P_i on $[x_{i+1} - \frac{1}{2L_{i+1}}, x_{i+1} + \frac{1}{2L_{i+1}}]$, since

$$|P_i'(x)| \leq \sum_{k=1}^{N_i} kA_{i+1}L_{i+1}^k \left(\frac{1}{2L_{i+1}}\right)^{k-1} \leq 2A_{i+1}L_{i+1} \sum_{k=1}^{\infty} \frac{k}{2^k} = 4A_{i+1}L_{i+1}.$$

Let $\tilde{a}_{i,j}$ be a dyadic rational approximation of $a_{i,j}$ up to error $2^{-n-\log N_i - 2}$ and $\tilde{x}_i$ a dyadic rational approximation of x_i up to error $\min\{2^{-n-C_i-2}, \frac{1}{8L_{i+1}}\}$. The polynomial

$$\tilde{P}_i := \sum_{j=0}^{N_i} \tilde{a}_{i+1,j}(x - \tilde{x}_{i+1})^j$$

then approximates f on $[c_i, c_{i+1}]$ up to error 2^{-n-1}, since

$$|P_i(x) - \tilde{P}_i(x)| \leq |P_i(x + x_{i+1} - \tilde{x}_{i+1}) - \tilde{P}_i(x)| + |P_i(x) - P_i(x + x_{i+1} - \tilde{x}_{i+1})|$$

with

$$\begin{aligned}
|P_i(x + x_{i+1} - \tilde{x}_{i+1}) - \tilde{P}_i(x)| &\leq \sum_{k=0}^{N_i} |a_{i+1,k} - \tilde{a}_{i+1,k}||x - \tilde{x}_{i+1}|^k \\
&\leq \sum_{k=0}^{N_i} 2^{-n-3}(|x - x_{i+1}| + |x_{i+1} - \tilde{x}_{i+1}|)^k \\
&\leq 2^{-n-3} \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k \\
&\leq 2^{-n-2}
\end{aligned}$$

and

$$|P_i(x) - P_i(x + x_{i+1} - \tilde{x}_{i+1})| \leq 2^C |x_{i+1} - \tilde{x}_{i+1}| \leq 2^{-n-2}.$$

Note that the above estimate is correct, since for $x \in [c_i, c_{i+1}]$ we have

$$x + x_{i+1} - \tilde{x}_{i+1} \in [x_{i+1} - \frac{1}{2L_{i+1}}, x_{i+1} + \frac{1}{2L_{i+1}}].$$

The complexity of the computation of $\tilde{P}_i$ is majorised by the complexity of the polynomial translation algorithm (4.7), where the degree is given by N_i, the size of the coefficients is bounded by $n + \log N_i + 2$ and the size of $\tilde{x}_{i+1}$ is bounded by $n + C_i + 2$, so the complexity is bounded by

$$\mathcal{O}(N_i^2(n + \log N_i)(n + C_i) + N_i^3(n + C_i)^2)$$

which simplifies to

$$\mathcal{O}(N_i^3(n + C_i)^2).$$

Since there are B such polynomials to compute, the total complexity is given by

$$\mathcal{O}(\sum_{k=0}^{B} N_k^3(n + C_k)^2) = \mathcal{O}(B(\log A + n)^3(n + \log A + L)^2).$$

where $A = \max_{k=0,\dots,B} A_k$, $L = \max_{k=0,\dots,B} L_k$. By Proposition 4.7, the coefficients of the thus computed polynomials are bounded by

$$n + \log(n + \log A + 2) + 2 + (n + \log A + 1)(n + \log A + \log L + 3).$$

Using Lemma 3.7, we may now truncate the coefficients to obtain a smaller output. Note that according to the proof of 4.7 the logarithm of the coefficients of $\tilde{P}_i$ in the monomial basis is bounded by $\log A + N \log L + N$, so according to Lemma 3.7, we can compute for each i a polynomial

$$Q_i(x) = \sum_{j=0}^{N_i} b_{i+1,j} x^j$$

with coefficient-size $\log A + N \log L + N + n + \log N + 1$ satisfying $|b_{i+1,j} - \hat{a}_{i+1,j}| \leq 2^{-n-1-\log N}$, where $\hat{a}_{i+1}, j$ is the j^{th} coefficient of $\tilde{P}_i$ in the monomial basis, and thus

$$\| Q_i - \tilde{P}_i \| \leq 2^{-n-1}.$$

According to Lemma 3.7, this normalisation step takes time

$$\mathcal{O}((n + \log A)(n + \log A + \log L)).$$

□

Remark 6.4. (i) Actually, the representations $\delta_{C^\omega([0,1])}$ and $\delta_{C^\omega,\mathtt{pp}}$ are equivalent, since $\delta_{C^\omega,\mathtt{pp}}$ allows for efficient function evaluation, which in turn allows to retrieve the coefficients of the Taylor series (cf. [Mü87] and [KMRZ12]).

(ii) Since $\delta_{C^\omega,\mathtt{pp}}$ is a seemingly natural representation of $C^\omega([0,1])$ which renders maximization, evaluation and integration second-order polynomial time computable and since non-uniformly these problems are equivalent, one might hope that $\delta_{C^\omega,\mathtt{pp}}$ characterises some of them. In fact, as we will show in a moment, it does nonuniformly characterise evaluation but does not uniformly characterise any of them (cf. [KMRZ12] 'Optimality Questions').

Proposition 6.5. *$\delta_{C^\omega,\mathtt{pp}}$ nonuniformly characterises evaluation, that is, the functional*

$$\mathbf{EVAL}\colon\ C^\omega([0,1]) \times [0,1] \to \mathbb{R},\ (f,x) \mapsto f(x)$$

with parameters represented by $\rho_c|^{[0,1]}$ and image space represented by ρ.

Proof. As mentioned before, evaluation of a real analytic function in polynomial time allows for retrieving its Taylor-coefficients in polynomial time and the additional parameters are constants and thus trivially computable in polynomial time, which shows that $\delta_{C^\omega([0,1])}$ non-uniformly characterises **EVAL**. For details, see the proof of '$\alpha \preceq_p \tilde{\beta}$' in [KMRZ12]. The result then follows from Theorem 6.3 and Proposition 3.14. □

Remark 6.6. A more general version of the above proposition Proposition can be found in [KMRZ12], Remark 4.2.

Proposition 6.7. *The natural embedding $C^\omega([0,1]) \to C([0,1])$ is $(\delta_{C^\omega,\mathtt{pp}}, \delta_{C([0,1]),\mathtt{pp}})$-$\mathbf{FP}^2$.*

Proof. A $\delta_{C^\omega,\mathtt{pp}}$-name immediately yields an approximation to f by piecewise polynomial functions $(\vec{P}_n)_n$ with discontinuities at the breakpoints $(\vec{a}_n)_n$. To obtain a $\delta_{C([0,1]),\mathtt{pp}}$-name we need to introduce 'joints' to make the approximating functions continuous. For this, recall that $C := \log A_f + \log L_f$ is a Lipschitz-constant for f on $[0,1]$. For each k define $a^r_{n,k} := a_{n,k} + 2^{-C(n+2)-1}$, $a^l_{n,k} := c_k - 2^{-C(n+2)-1}$ and

$$l_k(x) := P_{n,k}(a^l_{n,k}) + \frac{x - a^l_{n,k}}{a^r_{n,k} - a^l_{n,k}}(P_{k+1}(a^r_{n,k}) - P_k(a^l_{n,k}))$$

on $[a^l_{n,k}, a^r_{n,k}]$. We then have for all $x \in [a^l_{n,k}, a^r_{n,k}]$

$$|f(x) - l_k(x)| \leq |f(a^l_{n,k}) - l_k(a^l_{n,k})| + |l_k(x) - l(a^l_{n,k})| + |f(x) - f(a^l_{n,k})|$$

with $|f(x) - f(a^l_{n,k})| \leq 2^{-n-2}$, $|f(a^l_{n,k}) - l_k(a^l_{n,k})| \leq 2^{-n-1}$ and $|l(x) - l(a^l_{n,k})| \leq 2^{-n-2}$. In order to compute a $\nu_{\mathbb{D}[x]}$-name of l_i, we simply need to evaluate the polynomials $P_{n,i}$ in $a^l_{n,k}$ and $a^l_{n,k}$, which can be done in second-order polynomial time according to 4.6, and then perform some elementary arithmetic operations. Observe that $a^r_{n,k} - a^l_{n,k}$ is a power of 2, so we do not have to perform actual divisions (i.e. we can compute l_i exactly). $\square$

Corollary 6.8. *(i) Every polynomial-time computable real analytic function on* $[0,1]$ *(in the sense of Definition 1.1) is an* $\mathbf{FP}$*-point of* $\delta_{C([0,1]),\mathtt{pp}}$.

(ii) More generally, every polynomial-time computable real analytic function on $[a,b]$*, where* $a,b \in \mathbb{D}$ *is an* $\mathbf{FP}$*-point of* $\delta_{C[a,b],\mathtt{pp}}$.

Proof. Item (i) immediately follows from Propositions 6.5, 6.7 and 3.14 and Example 3.16. If $f \in C^\omega([a,b])$ is computable in polynomial time, then so is $\hat{f}(x) := f(a+x(b-a))$, where $x \in [0,1]$. The result then follows from (i) and Proposition 5.21. $\square$

We can now prove Theorem 1.4, which we may also state more uniformly

Theorem 6.9. *Let f be an efficiently constructible function over the interval* $[0,1]$*. Then* $f \in \mathbf{FP}_{\delta_{C([0,1]),\mathtt{pq}}}(C([0,1]))$*. In particular, f and* $\mathbf{MAX}(f,\cdot,\cdot)$ *are polynomial-time computable functions. Moreover, a* $\delta_{C([0,1]),\mathtt{pq}}$*-name of f can be computed in polynomial time, given polynomial-time computable* $\delta_{C^\omega,\mathtt{pp}}$*-names (or* $\delta_{C^\omega([0,1])}$*-names) of the basic functions used to construct f. If the construction of f does not involve divisions or square-roots, then f is already an* $\mathbf{FP}$*-point of* $\delta_{C([0,1]),\mathtt{pp}}$*, so in particular,* $\mathbf{INT}(f,\cdot,\cdot)$ *is a polynomial-time computable function.*

Proof. Corollary 6.8 asserts the claim for all polynomial-time computable real analytic functions. According to Theorem 5.11, the set $\mathbf{FP}_{\delta_{C([0,1]),\mathtt{pq}}}(C([0,1]))$ is 'uniformly' closed under addition, multiplication and division. Theorem 5.13 and Corollary 5.17 show the closure under case distinctions at polynomial-time computable points and taking square roots. Because of Corollary 6.8 and Proposition 5.21, the same holds true for $\mathbf{FP}_{\delta_{C[a,b],\mathtt{pq}}}(C([a,b]))$, where $a,b \in \mathbb{D}$. Now, any efficiently constructible function $f \in C([0,1])$ can be written in the form

$$f := f_1 \circ f_2 \circ \cdots \circ f_s$$

where $f_1, \dots, f_s$ are efficiently constructible functions over compact intervals with dyadic rational endpoints, whose construction does not involve composition. It follows that the f_i's are **FP**-points of $\mathbf{FP}_{\delta_{C[a_i,b_i],\mathtt{pq}}}(C([a_i,b_i]))$ for appropriate $a_i, b_i \in \mathbb{D}$. Using Proposition 5.21 and the remark following Theorem 5.8 we can write

$$f := \tilde{f}_1 \circ \tilde{f}_2 \cdots \circ \tilde{f}_s$$

where $\tilde{f}_i \in \mathbf{FP}_{\delta_{C([0,1]),\mathtt{pq}}}(C([0,1]))$ for $i = 1, \dots, s$. So, by Theorem 5.8, f is an **FP**-point of $\delta_{C([0,1]),\mathtt{pq}}$. The case where the construction of f does not involve square-roots and divisions is proved analogously and the fact, that a name of f can be computed in polynomial time, given polynomial-time computable names of the basic functions involved in the construction follows from the constructive nature of the theorems involved in this proof. $\square$

Now, since as mentioned before, $\delta_{C^\omega,\mathtt{pp}}$ behaves essentially like $\delta_{C([0,1]),\mathtt{pp}}$, we are able to formulate an analogue to Theorem 5.7 for the space $C^\omega([0,1])$.

Theorem 6.10. *(i) Evaluation*

$$C^\omega([0,1]) \times [0,1] \ni (f,x) \mapsto f(x) \in \mathbb{R}$$

is $(\delta_{C^\omega,\mathtt{pp}} \times \rho_c, \rho)$-computable in time

$$\mathcal{O}(B_f(n+\log A_f+\log L_f+p_f)+d_f^2(n+\log A_f)^2(n+\log A_f+\log L_f)(n+\log A_f+b_f\log L_f))$$

and $(\delta_{C^\omega([0,1])} \times \rho_c, \rho)$-computable in time

$$\mathcal{O}(B_f(n+\log A_f)^3(n+\log A_f+L_f)^2).$$

(ii) Parametrised maximisation

$$C^\omega([0,1]) \times [0,1] \times [0,1] \ni (f,a,b) \mapsto \max_{x\in[a,b]} f(x) \in \mathbb{R}$$

is $(\delta_{C^\omega,\mathtt{pp}} \times \rho_c \times \rho_c, \rho)$-computable in time

$$\mathcal{O}(B_f d_f^5(n+\log A)^7(b_f\log L+d_f)^2 + B_f d_f^2(n+\log A)^2 p_f^2).$$

and $(\delta_{C^\omega([0,1])} \times \rho_c \times \rho_c, \rho)$-computable in time

$$\mathcal{O}(B_f(n+\log A_f)^3(n+\log A_f+L_f)^2 + B_f(n+\log A_f)^7(\log L_f)^2).$$

(iii) Parametrised integration

$$C^\omega([0,1]) \times [0,1] \times [0,1] \ni (f,a,b) \mapsto \int_a^b f(x) \in \mathbb{R}$$

is $(\delta_{C^\omega,\mathtt{pp}} \times \rho_c \times \rho_c, \rho)$-computable in time

$$\mathcal{O}(B_f d_f(n+\log A_f)^2(b_f(n+\log A_f+\log d_f+p_f)\log L + d_f(n+\log A_f+p_f)^2)).$$

and $(\delta_{C^\omega([0,1])} \times \rho_c \times \rho_c, \rho)$-computable in time

$$\mathcal{O}(B_f(n+\log A_f)^3(n+\log A_f+L_f)^2).$$

Proof. The proof is analogous to that of Theorem 5.7. The logarithm of a Lipschitz-constant of f is given by $\log A + \log L$ and the absolute value of f on $[0,1]$ is bounded by A, which yields the slightly improved bounds stated. The complexity-bounds for $\delta_{C^\omega([0,1])}$ then follow from Theorem 6.3. □

Remark 6.11. The complexity bounds obtained here for evaluation and maximisation, given a $\delta_{C^\omega([0,1])}$-name, are fairly crude - better bounds would be obtained by working with $\delta_{C^\omega([0,1])}$ directly without taking the 'detour' over $\delta_{C^\omega,\mathsf{pp}}$ and invoking the Translation Algorithm 4.7, which in both cases dominates the overall complexity.

Finally, we observe that $\delta_{C^\omega,\mathsf{pp}}$ renders some natural operations on $C^\omega([0,1])$ second-order polynomial-time computable. The case of differentiation requires some technical preparation.

Lemma 6.12. *(i) Let $r > 1$. Then we have*

$$k \leq \frac{r^k}{e \ln r}$$

for all $k \in \mathbb{N}$.

(ii) More generally,

$$k^d \leq \left(\frac{d}{e \ln r}\right)^d r^k.$$

Theorem 6.13. *Let A,B,L p, be a bounds on A_f,A_g,B_f,B_g,L_f,L_g,p_f,p_g respectively.*

(i) Addition

$$C^\omega([0,1]) \times C^\omega([0,1]) \ni (f,g) \mapsto f + g \in C^\omega([0,1]),$$

is $(\delta_{C^\omega,\mathsf{pp}} \times \delta_{C^\omega,\mathsf{pp}}, \delta_{C^\omega,\mathsf{pp}})$-computable in time $\mathcal{O}(Bd_f b_f(n + \log A)^2 \log L + Bp + L)$

(ii) Multiplication

$$C^\omega([0,1]) \times C^\omega([0,1]) \ni (f,g) \mapsto f \cdot g \in C^\omega([0,1]),$$

is $(\delta_{C^\omega,\mathsf{pp}} \times \delta_{C^\omega,\mathsf{pp}}, \delta_{C^\omega,\mathsf{pp}})$-computable in time

$$\mathcal{O}(Bd^2 b^2 (n + \log A)^4 (\log L)^2 + Bp + L),$$

(iii) Iterated Anti-Differentiation

$$C^\omega([0,1]) \times \mathbb{N} \ni (f,m) \mapsto f^{(-m)} \in C^\omega([0,1]),$$

where $f^{(-m)}$ is the unique function satisfying $(f^{(-m)})^{(k)}(0) = 0$ for $k < m$ and $(f^{(-m)})^{(m)} = f$, is $(\delta_{C^\omega,\mathsf{pp}} \times \nu_1, \delta_{C^\omega,\mathsf{pp}})$-computable in time

$$\mathcal{O}\left(B_f(n + \log A_f) d_f \left[m^2 (\log(d_f(n + \log A_f)))^2 + m b_f (n + \log A_f) \log L_f \log(d_f(n + \log A_f))\right]\right).$$

The output parameters are $A_{f^{(-m)}} = A_f$, $L_{f^{(-m)}} = L_f$, $d_{f^{(-m)}} := d_f + m$ and $b_{f^{(-m)}} := b_f + \log d_f + 2$.

(iv) Iterated Differentiation

$$C^\omega([0,1]) \times \mathbb{N} \ni (f,m) \mapsto f^{(m)} \in C^\omega([0,1])$$

is $(\delta_{C^\omega,\mathrm{pp}} \times \nu_1, \delta_{C^\omega,\mathrm{pp}})$*-computable in time*

$$\mathcal{O}(m^2 B_f N(\log N)^2 + m B_f b_f N \log N),$$

where $N = 2d_f(n+\log A+m\log L+2m\log m+2m\log d_f+m\log 20+3)$*. The output parameters are* $d_{f^{(m)}} = 6d_f+2d_f\log d_f$*,*$b_{f^{(m)}} := b_f+m\log\log d_f$*,* $A_{f^{(m)}} := 4^m m^m A L^m$ *and* $L_{f^{(m)}} := 2L_f$*.*

Proof. The results (i) and (ii) follow almost immediately from Theorem 5.10, we only have to tread the additional parameters.

(i) If for all $x \in [0,1]$ we have $f^{(k)}(x) \le A_f \cdot L_f^k \cdot k!$ and $g^{(k)}(x) \le A_g \cdot L_g^k \cdot k!$, then

$$f^{(k)}(x) + g^{(k)}(x) \le A_f \cdot L_f^k \cdot k! + A_g \cdot L_g^k \cdot k! \le (A_g + A_f)(\max\{L_f, L_g\})^k k!.$$

So in addition to a $\delta_{C([0,1]),\mathrm{pp}}$-name of $f+g$, obtained via 5.10, we output $A_f + A_g$ and $\max\{L_f, L_g\}$.

(ii) Picking up the notation in (i), we have

$$(fg)^{(k)}(x) = \sum_{i=0}^{k} \binom{k}{i} f^i(x) g^{k-i}(x) \le \sum_{i=0}^{k} \binom{k}{i} A_f L_f^i i! A_g L_g^{k-i}(k-i)! = A_f A_g (L_f + L_g)^k k!.$$

So in addition to a $\delta_{C([0,1]),\mathrm{pp}}$-name of $f+g$, obtained via 5.10, we output $A_f \cdot A_g$ and $L_f + L_g$. Note that a bound on $|\log f(x)|$ on $[0,1]$ is given by $\log A_f$, so we may replace the bound $c_f(0) + \log d_f(0)$ found in 5.10 by this.

(iii) Let f_{n+1} be the piecewise polynomial function given by the $\delta_{C^\omega,\mathrm{pp}}$-name of f approximating f up to error 2^{-n-1}. For each polynomial of f_{n+1} compute the m-fold anti-derivative up to error 2^{-n-1} using Proposition 4.14 (i.e. approximate the coefficients up to error $2^{-n-\log d}$ where d is the degree of f_{n+1}). The piecewise-polynomial function thus obtained approximates f up to error 2^{-n}. By the Mean Value Theorem, A and L, such that equation (1) is satisfied for $f^{(-m)}$ are given by A_f and L_f, the degree of the output-polynomial is $d_f(n+\log A+2)+m$ and its coefficient-size is at most

$$b_f(n+\log A)\log L + n + \log(d_f(n+\log A+2)),$$

so we may put $d_{f^{(-m)}} := d_f + m$ and $b_{f^{(-m)}} := b_f + \log d_f + 2$. According to Proposition 4.14, the total complexity is given by

$$\mathcal{O}\left(B_f(n+\log A)d_f\left[m^2(\log(d_f(n+\log A)))^2 + m b_f(n+\log A)\log L\log(d_f(n+\log A))\right]\right).$$

(iv) Let $d \ge m$. For $x_0 \in [0,1]$ consider the Taylor-polynomial

$$T_{x_0}(x) = \sum_{j=0}^{d} \frac{f^{(j)}(x_0)}{j!}(x-x_0)^j.$$

Note that f is analytic on $B(x_0, \frac{1}{2L})$ and that $|T_{x_0}(x) - f(x)| \le A2^{-d}$ for all $x \in B(x_0, \frac{1}{2L})$. Now, for every $x \in [x_0 - \frac{1}{2L}, x_0 + \frac{1}{2L}]$ there exists a $\xi \in [x_0 - \frac{1}{2L}, x_0 + \frac{1}{2L}]$ such that

$$T_{x_0}^{(m)}(x) - f^{(m)}(x) = \frac{f^{(d+1)}(\xi)}{(d-m+1)!}(x-x_0)^{d-m+1}.$$

Using the estimate (1), we obtain

$$|T_{x_0}^{(m)}(x) - f^{(m)}(x)| \le AL^m(d+1)^m 2^{m-d-1}.$$

Now Lemma 6.12 yields

$$(d+1)^m \le \left(\frac{m}{e\ln\sqrt{2}}\right)^m (\sqrt{2})^{d+1}$$

so

$$|T_{x_0}^{(m)}(x) - f^{(m)}(x)| \le AL^m \left(\frac{m}{e\ln\sqrt{2}}\right)^m 2^{m-d/2-1/2} \le AL^m 2^m m^m 2^{m-d/2-1/2}. \qquad (2)$$

Let $x \in [a_k, a_{k+1}]$. Let P be the polynomial given by the $\delta_{C^\omega,\mathtt{pp}}$-name of f approximating f up to error $2^{-d+\log A+1}$. Note that P has degree $d_f \cdot d$. By Markov's inequality (more precisely, Proposition 5.19) we have

$$\begin{aligned}|P^{(m)}(x) - T_{x_0}^{(m)}(x)| &\le L^m 4^m d_f^{2m} d^{2m} \max\{|P(x) - T_{x_0}(x)| \mid x \in [x_0 - \tfrac{1}{2L}, x_0 + \tfrac{1}{2L}]\}\\ &\le L^m 4^m d_f^{2m} d^{2m}(2^{\log A+1-d} + A2^{-d}).\end{aligned}$$

Again, Lemma 6.12 yields

$$d^{2m} \le \left(\frac{2m}{e\ln\sqrt{2}}\right)^{2m} (\sqrt{2})^d \le 5^m m^{2m} (\sqrt{2})^d.$$

So we obtain

$$|P^{(m)}(x) - T_{x_0}^{(m)}(x)| \le AL^m 20^m d_f^{2m} m^{2m}(2^{1-d/2} + 2^{-d/2}). \qquad (3)$$

Consequently, if

$$d \ge 2(n + \log A + m\log L + 2m\log m + 2m\log d_f + m\log 20 + 3)$$

then equations (1) and (2) yield

$$|P^{(m)}(x) - f^{(m)}(x)| \le |P^{(m)}(x) - T_{x_0}^{(m)}(x)| + |T_{x_0}^{(m)}(x) - f^{(m)}(x)| \le 2^{-n}.$$

The polynomial P has degree

$$d' := 2d_f(n + \log A + m\log L + 2m\log m + 2m\log d_f + m\log 20 + 3)$$

and $P^{(m)}$ can be obtained using Proposition 4.13 in time $\mathcal{O}(d'm^2(\log d')^2 + md'b_f\log d')$, where d' is the degree of P. Since there are B_f differentiations to perform the complexity of this step is $\mathcal{O}(B_f d'm^2(\log d')^2 + B_f m d' b_f \log d')$. It remains to compute the new parameters A'_f, L'_f, d'_f, b'_f such that equation (1) is satisfied for $f^{(m)}$ with A'_f, L'_f,such that the degree of $P^{(m)}$ is bounded by $d'_f(n + \log A + 1)$ and such that the coefficient-size of $P^{(m)}$ is bounded by $b'_f(n + \log A')\log L'$. Equation (1) yields

$$f^{(m+k)}(x) \le AL^{m+k}(m+k)! \le AL^{m+k}(m+k)^m k!$$

and by Lemma 6.12

$$(m+k)^m \le \left(\frac{m}{e\ln 2}\right)^m 2^{m+k} \le m^m 4^m 2^k.$$

So we may put $A'_f := 4^m m^m A L^m$ and $L'_f := 2L_f$. In order to obtain a bound on the degree of P we may then put

$$d'_f := 2d_f \left(1 + \frac{m\log m + 2m\log d_f + m\log 5 + 2}{n + \log A + m\log L + 2m + m\log m + 1}\right)$$

which can be very roughly estimated by

$$d'_f \leq 6d_f + 2d_f \log d_f.$$

The coefficient-size of the new polynomials is bounded by

$$b_f(n + \log A)\log L + m\log d',$$

so we may put $b'_f := b_f + m\log\log d_f$.

$\square$

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[BPR08] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry - second edition*. Springer, 2008. available online at http://perso.univ-rennes1.fr/marie-francoise.roy/bpr-ed2-posted2.pdf - retrieved May 1st 2012.

[Che66] E. W. Cheney. *Introduction to Approximation Theory*. AMS Chelsea, 1966.

[Fü09] Martin Fürer. Faster integer multiplication. *SIAM Journal on Computing*, 39-3:979–1005, 2009.

[Kaw11] Akitoshi Kawamura. On function spaces and polynomial-time computability, 2011. ongoing work. Preprint available online at www-imai.is.s.u-tokyo.ac.jp/ kawamura/dagstuhl.pdf - retrieved August 21st 2012.

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

[KMRZ12] Akitoshi Kawamura, Norbert Th. Müller, Carsten Rösnick, and Martin Ziegler. Parameterized uniform complexity in numerics: from smooth to analytic, from $\mathcal{NP}$-hard to polytime. unpublished preprint, 2012.

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

[LHE01] S. Labhalla, H.Lombardi, and E.Moutai. Espaces métriques rationnellement présentés et complexité, le cas de l'espace des fonctions réelles uniformément continues sur un intervalle compact. *Theoretical Computer Science*, 250:265–332, 2001.

[Mü87] Norbert Th. Müller. Uniform computational complexity of taylor series. In *Automata, Languages and Programming*, volume 267 of *Lecture Notes in Computer Science*, pages 435–444. Springer, 1987.

[New64] D. J. Newman. Rational approximation to $|x|$. *Michigan Math. Journal*, 11:11–14, 1964.

[Rud87] Walter Rudin. *Real and Complex Analysis*. McGraw-Hill, 1987.

[SS71] A. Schönhage and V. Strassen. Schnelle multiplikation großer zahlen. *Computing*, 7:281–292, 1971.

[Usp48] J.V. Uspensky. *Theory of Equations*. MacGraw Hill, 1948.

[Vin36] A.J.H. Vincent. Sur la résolution des équations numériques. *Journal de Mathématiques Pures et Appliquées*, pages 341–372, 1836.

[Wei00] Klaus Weihrauch. *Computable Analysis*. Springer, 2000.