On representations of irrational numbers and the computational complexity of converting between such representations

Amir Ben-Amram*   Lars Kristiansen†
Jakob Grue Simonsen‡

June 25, 2025

Abstract

We study the computational complexity of converting between different representations of irrational numbers. Typical examples of representations are Cauchy sequences, base-10 expansions, Dedekind cuts and continued fractions.

1 Introduction

1.1 Motivations

Real numbers can be represented by Cauchy sequences, base-10 expansions, Dedekind cuts, continued fractions and a number of other representations (we will consider quite a few of them in this paper). Our goal is to analyze the computational complexity of converting one representation into another. Let us say that we have access to the Dedekind cut of the real number α. How hard will it be to compute a Cauchy sequence for α? How hard will it be to compute the continued fraction of α? Or let us say that we have access to the continued fraction of α, how hard will it then be to compute the base-10 expansion of α? Will there be an efficient algorithm? Can it be done in polynomial time? Exponential time?

These are very natural questions to ask, but they are also naive, and the way the questions are posed above, does hardly make any sense at all. We will aim at posing such questions in a mathematically satisfactory manner, and then derive

---
*Qiryat Ono, Israel
†Department of Mathematics, University of Oslo, Norway and Department of Informatics, University of Oslo, Norway
‡IT University of Copenhagen, Denmark

1

reasonably tight upper bounds on the computational complexity of a number of conversions.

At the outset we are mainly interested in classic and historically prominent representations like Cauchy sequences and Dedekind cuts, which are important in the foundation of analysis, or base-b expansions, continued fractions and representations related to the Stern-Brocot tree which have numerous applications in both pure and applied mathematics. In general, we cannot uniformly convert one of these representations into another. This is well known. E.g., we cannot uniformly convert a Cauchy sequence into a base-b expansion: No algorithm with oracle access to a Cauchy sequence for a real number α can compute the base-b expansion of α because it is not possible to determine the digits in such an expansion by examining a finite part of the Cauchy sequence. The conversion is not continuous (see e.g. [40]), that is, we cannot determine a finite part of the output by examining a finite part of the input. For the same reason, we cannot uniformly convert Dedekind cuts to continued fractions or Cauchy sequences to Dedekind cuts.

The computational complexity of an operation is determined by imposing resource bounds on an algorithm (Turing machine) carrying out the operation. The considerations above show that, in general, it may make no sense to talk about the computational complexity of converting one representation of (the set of all) real numbers into another representation. If there does not exists a uniform conversion from one representation to another, then no algorithm can carry out the conversion, and then there is nothing to impose resource bounds on.

The absence of uniform conversions is due to the presence of the rational numbers. If we exclude the rationals and restrict our attention to the irrational numbers, then it is indeed possible to uniformly convert back and forth between any of the classic representations discussed above. We will use this insight to give a formal definition of what a *representation of an irrational number* is, see Definition 1.2 below. Roughly speaking, the definition states that a class of functions is a representation if it is uniformly computably equivalent to the class of Dedekind cuts of the irrational numbers. In addition to capture the classic representations discussed above, this definition is generous enough to include some representations popular in the practice of numerical computations with exact real arithmetic (Boehm et al. [10]) and modern computable analysis (Weihrauch [41], Ko [29]). We will study all sorts of representations captured by our definition, including some representations due to the authors.

We note that there exists a very general and well-developed theory of Type-2 computability, continuity, and representation of sets of continuum cardinality (some references are Weihrauch [42], Kreitz and Weihrauch[30], Brattka et al. [12] and Weihrauch [41]). This theory may serve as a foundation for modern computable analysis and allows for a wide range of representations of real and irrational numbers. In the present paper, our definition of the notion of *representation* (of the irrational numbers) is restrictive in the sense that it excludes some of the representations allowed in Type-2 computability. We do not regard this as a problem as our definition is instrumental rather than foundational,

2

and its purpose is to provide a neat and transparent development of the degree structure depicted in Figure 1.

1.2 What is a representation?

Formally, a *representation* of the *irrational* numbers will be a class of functions with countable domain and codomain (typically $\mathbb{Q}$ or $\mathbb{N}$)[1]. Every function in the class will represent a particular irrational number, and each irrational number will be represented by some function in the class. The class of Dedekind cuts of irrational numbers will be a canonical representation to us.

Definition 1.1. The *Dedekind cut* of an irrational number α is the function $\alpha : \mathbb{Q} \longrightarrow \{0, 1\}$ where

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

$\square$

Each irrational number has a unique representation in this class, and we can identify an irrational number α with its Dedekind cut $\alpha : \mathbb{Q} \longrightarrow \{0, 1\}$.

We will take advantage of the uniqueness of the Dedekind cuts to define what a representation in general is. We refer to the functions in a representation R as R-representations. When f is an R-representation of α, we will require that it is possible to compute the Dedekind cut of α in f, that is, we will require that there exists an oracle Turing machine M such that

$$\alpha(q) = \Phi_M^f(q)$$

where Φ_M^f is the function computed by M with oracle f. We will also require that at least one R-representation f of α can be computed in the Dedekind cut of α, that is, we will require that there exists an oracle Turing machine N and an R-representation f such that

$$f(x) = \Phi_N^\alpha(x)$$

where Φ_N^α is the function computed by N when the oracle is the Dedekind cut of α. We are now ready to give our formal definition.

Definition 1.2. A class of functions R is a *representation (of the irrational numbers)* if

1. There exists a Turing machine M with the following property: For every $f \in R$ there exists an irrational α such that $\alpha = \Phi_M^f$. When $\alpha = \Phi_M^f$, we say that f *represents* α and that f is an *R-representation* of α.

---

[1]In fact, as we use Turing machines, we assume the domain and codomain to be equipped with an encoding as words over a finite alphabet. For $\mathbb{N}$ and $\mathbb{Q}$ we use standard encodings, described later.

3

2. There exists a Turing machine N with the following property: For every irrational α there exists an R-representation f of α such that $f = \Phi_N^\alpha$.

We say that an oracle Turing machine M *converts an R_1-representation into an R_2-representation* if for any $f \in R_1$ representing α there exists $g \in R_2$ representing α such that $g = \Phi_M^f$. $\qquad\qquad\square$

Let us study a few examples in order to see how this definition works. We define a Cauchy sequence for α as a function $C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ with the property

$$|C(n) - \alpha| < n^{-1} .$$

Let $\mathcal{C}$ be the class of all Cauchy sequences for all irrational numbers. We will now argue that $\mathcal{C}$ is a representation according to the definition above.

First we observe that we can compute the Dedekind cut of an irrational α in any Cauchy sequence for α. In order to compute $\alpha(q)$, we search for the least $n > 0$ such that $|C(n) - q| > n^{-1}$. This search terminates as q is rational and α is irrational. If $q < C(n)$, it will be the case that $\alpha(q) = 0$ (we have $q < \alpha$), otherwise, we have $q > C(n)$, and then it will be case that $\alpha(q) = 1$ (we have $q > \alpha$). Thus there will be an oracle Turing machine M such that $\alpha = \Phi_M^f$ whenever f is a Cauchy sequence for α. Now, M has the following property: For every $f \in \mathcal{C}$ there exists irrational α such that $\alpha = \Phi_M^f$. This shows that clause (1) of Definition 1.2 is satisfied.

Next we observe that we can compute a Cauchy sequence C for α if we have access to the Dedekind cut of α. We can use the Dedekind cut to find an integer a such that $a < \alpha < a + 1$. Thereafter, we can use the Dedekind cut and the equations

$$C(1) = a + \frac{1}{2} \quad \text{and} \quad C(i+1) = \begin{cases} C(i) - 2^{-i-1} & \text{if } C(i) > \alpha \\ C(i) + 2^{-i-1} & \text{if } C(i) < \alpha \end{cases}$$

to compute $C(n)$ for arbitrary n. This is one possible way to compute a Cauchy sequence in a Dedekind cut. There are for sure other ways. Other algorithms may yield Cauchy sequences that converge faster, or slower, than the ones computed by the algorithm suggested above. Anyway, there will be an oracle Turing machine N such that we have $C = \Phi_N^\alpha$ where C is some Cauchy sequence for α.

Now, in order to see that clause (2) of Definition 1.2 is satisfied, pick an arbitrary irrational α. Then we have $f = \Phi_N^\alpha$ for some Cauchy sequence f, moreover, we have $\alpha = \Phi_M^f$, and hence f is a $\mathcal{C}$-representation of α. Hence, N has the following property: For every irrational α there exists a $\mathcal{C}$-representation f of α such that $f = \Phi_N^\alpha$. This shows that also clause (2) is satisfied, and we conclude that $\mathcal{C}$, that is, the class of all Cauchy sequences for all irrational numbers, is a representation.

Let $a_0, a_1, a_2, \ldots$ be an infinite sequence of integers, where $a_1, a_2, a_3, \ldots$ are

4

positive. The *continued fraction* $[a_0; a_1, a_2, \ldots]$ is defined by

$$[\,a_0; a_1, a_2, a_3, \ldots\,] \;=\; a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ldots}}}$$

It is well known that any irrational number can be uniquely written as an infinite continued fraction, and moreover, each infinite continued fraction equals an irrational number (rationals have finite continued fractions). There is a one-to-one correspondence between the infinite continued fractions and the irrational numbers.

Let $\mathcal{F}$ be the class of all infinite continued fractions where $[\,a_0; a_1, a_2, a_3 \ldots\,]$ is identified with a function f where $f(n) = a_n$. Then $\mathcal{F}$ will be a representation according to Definition 1.2. If we have access to the continued fraction of α, we can compute the Dedekind cut of α, and if we have access to the Dedekind cut of α, we can compute the continued fraction of α. Thus there exist Turing machines M and N such that

$$\alpha = \Phi_M^f \text{ and } f = \Phi_N^\alpha$$

whenever f is the continued fraction of the irrational number α. Each irrational number will have one, and only one, $\mathcal{F}$-representation, and it is easy to see that both clause (1) and clause (2) of Definition 1.2 is fulfilled.

We will not give a formal definition of a representation of all real numbers, and it is essential that Definition 1.2 is restricted to the irrational numbers. If we involve the rationals, the definition will not serve its purpose as we cannot always uniformly convert one standard representation into another, even if we are dealing with representations of a computable nature. E.g., the algorithm above converting a Cauchy sequence C into a Dedekind cut, searches for a number n such that $|C(n) - q| > n^{-1}$. This search will not terminate when C is a Cauchy sequence for the rational number q. Thus, if α might be rational, the algorithm does not yield a Turing machine M such that Φ_M^C is the Dedekind cut of α whenever C is a Cauchy sequence for α, moreover, it can be proven that no such Turing machine M exists (see Mostowski [36]). We *cannot* uniformly convert Cauchy sequences for *real* numbers into Dedekind cuts, but we *can* uniformly convert Cauchy sequences for *irrational* numbers into Dedekind cuts.

The purpose of Definition 1.2 is to capture what we intuitively consider as computable representations of the real numbers, and maybe somewhat paradoxically, we enable the study of conversions among such representations by restricting the definition to the irrational numbers.

A sequence of rationals $q_0, q_1, q_2, \ldots$ containing all rational numbers less than α, and nothing but the rationals less than α, will not yield a representation according to our definitions. Such a sequence defines a unique real number α, but we cannot use the sequence to compute the Dedekind cut of α (even if α is irrational). By the same token, Cauchy sequences without a modulus of convergence will not yield a representation.

5

1.3 An ordering relation on representations

We have seen that there is an algorithm for computing the Dedekind cut of an irrational α in an arbitrary Cauchy sequence for α. The algorithm searches for the least natural number n that fulfills certain criteria. It is easy to see that such an unbounded search is necessary. We cannot convert a Cauchy sequence into a Dedekind cut if we are not allowed to carry out unbounded search. Neither can we convert a base-10 expansion into a base-3 expansion if we are not allowed to carry out unbounded search. Suppose an oracle tells us that the base-10 expansion of an irrational starts with 0.66666. That will not be enough for us to decide if the base-3 expansion starts with 0.1 or 0.2. Thus, in order to determine which of these two options we should pick, we have to ask the oracle for the next digit of the decimal expansion, but of course, the next digit might also be 6, and so might the next after the next. The oracle may continue for an arbitrarily long time to tell us that the next digit is 6. Since the number is irrational the oracle will eventually yield a digit that allows to determine if the base-3 expansion starts with 0.1 or 0.2, but we need to carry out an unbounded search to get that digit.

Algorithms (conversions, computations, etc.) that do not perform unbounded search will be referred to as *subrecursive* algorithms (conversions, computations, etc.), in general, the word *subrecursive* signifies *absence of unbounded search*. This terminology might not be standard, but it will be very convenient.

We will now define an ordering relation $\preceq_S$ over the representations. Intuitively, the relation $R_1 \preceq_S R_2$ will indicate that the representation R_2 is more informative than the representation R_1. If $R_1 \preceq_S R_2$ holds, a Turing machine with oracle access to an R_2-representation of α can subrecursively (yes, that means without carrying out unbounded search) compute an R_1-representation of α. Thus, if the relation $R_1 \preceq_S R_2$ does not hold, it will not make much sense to talk about the computational complexity of converting an R_2-representation into an R_1-representation as such a conversion requires unbounded search, and thus, there will be no upper bound on the running time of a Turing machine undertaking the conversion. On the other hand, if the relation holds, it should make sense to analyze the computational complexity of the conversion. So far we have just indicated our intention with the relation; below we shall give the formal definition. First, however, we need an auxiliary definition. We are going to formulate our definition in terms of time-bounded computation, and there is need to specify what functions we admit as time bounds. We admit *time-constructible functions*, as defined next. This is a standard choice in complexity theory.

Definition 1.3. A function $t : \mathbb{N} \longrightarrow \mathbb{N}$ is a *time bound* if (i) $n \leq t(n)$, (ii) t is nondecreasing and (iii) t is time-constructible: there is a single-tape Turing machine that, on input 1^n, computes the binary expansion of $t(n)$ in $\Theta(t(n))$ steps. $\qquad\square$

Clause (iii) in the definition is needed because there are functions whose computational complexity is disproportionate to the size of their values. We

exclude such functions as time bounds, avoiding certain pitfalls in proofs. The class of functions we admit as time bounds includes all the functions familiar from analysis of algorithms such as polynomials (with positive coefficients), exponentials, the tower-functions, and so on.

Definition 1.4. Let t be a time-bound and let R be a representation. Then, $O(t)_R$ denotes the class of all irrational α in the interval $(0,1)$ such that at least one R-representation of α is computable by a Turing machine running in time $O(t(n))$ (where n is the length of the input).

Let R_1 and R_2 be representations. The relation $R_1 \preceq_S R_2$ holds if for any time-bound t there exists a time-bound s such that

$$O(t)_{R_2} \subseteq O(s)_{R_1} .$$

If the relation $R_1 \preceq_S R_2$ holds, we will say that the representation R_1 is *subrecursive* in the representation R_2. $\qquad \Box$

We will now study a few examples and discuss how the definition above works. Recall that $\mathcal{C}$ denotes the representation by Cauchy sequences (see page 4). Let $\mathcal{D}$ denote the representation by Dedekind cuts. It turns out that we have $\mathcal{C} \preceq_S \mathcal{D}$ and $\mathcal{D} \not\preceq_S \mathcal{C}$. Let us see why.

Intuitively we have $\mathcal{C} \preceq_S \mathcal{D}$ because a Cauchy sequence for an irrational α in the interval $(0,1)$ can be subrecursively computed in the Dedekind cut of α. No unbounded search is required. We simply sets $C(1)$ equal to $1/2$, and then we use the Dedekind cut and, e.g., the equations

$$C(i+1) = \begin{cases} C(i) - 2^{-i-1} & \text{if } C(i) > \alpha \\ C(i) + 2^{-i-1} & \text{if } C(i) < \alpha \end{cases}$$

to compute $C(n)$. Then $C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ will be a Cauchy sequence for α. Formally we have $\mathcal{C} \preceq_S \mathcal{D}$ because for every time-bound t there exists a time-bound s such that $O(t)_{\mathcal{D}} \subseteq O(s)_{\mathcal{C}}$. A Turing machine that uses the equations above to compute $C(n)$ needs to compute the Dedekind cut n times, hence at most $2^{\|n\|}$ times where $\|n\|$ is the length of the input. Furthermore, assuming numbers are represented in binary form, the Turing machine only needs to compute the Dedekind cut for inputs of size bounded by $2^{\|n\|}$. Hence, if the Dedekind cut of α is computable in time $O(t(m))$ where m is the length of the input and t is a time-bound, then a Turing machine can compute a Cauchy sequence for α in time $O(nt(2^{\|n\|}))$. Thus, let s be the time-bound $s(\|n\|) = nt(2^{\|n\|})$, and we have $O(t)_{\mathcal{D}} \subseteq O(s)_{\mathcal{C}}$.

Note that the set $O(t)_{\mathcal{D}}$ only contains irrationals from the interval $(0,1)$. This is important. When we compute a Cauchy sequence C for an irrational in this interval, we can simply let $C(1) = 1/2$. We cannot in general compute a Cauchy sequence for an irrational β subrecursively in the Dedekind cut of β. In order to set the value of $C(1)$ we will need a rational a such that $a < \beta < a+1$, and we cannot get hold of such an a without resorting to unbounded search.

7

Thus, if we do not restrict $O(t)_{\mathcal{D}}$ to irrationals in the interval $(0,1)$, the relation $\mathcal{C} \preceq_S \mathcal{D}$ will not hold. It is therefore important that we are interested in representations of the fractional part of irrational numbers.

Now, let us discuss why we have $\mathcal{D} \not\preceq_S \mathcal{C}$. As explained above, we have $\mathcal{D} \not\preceq_S \mathcal{C}$ because we cannot avoid unbounded search when we compute a Dedekind cut in a Cauchy sequence, but what does a formal proof look like? In general it is much harder to prove that the relation $\preceq_S$ does not hold than it is to prove that it holds. In order to prove $R_1 \preceq_S R_2$, we just have to come up with a subrecursive algorithm for converting an R_2-representation into an R_1-representation. In order to prove $R_1 \not\preceq_S R_2$, we have, according to our definitions, to prove that there is a time-bound t such that we have $O(t)_{R_2} \not\subseteq O(s)_{R_1}$ for any time-bound s. This might not be all that easy. This might call for involved diagonalization arguments. In some cases we might do with a growth argument, that is, we might be able to prove, for some time-bound t, that for any time-bound s there exists $\alpha_s \in O(t)_{R_2}$ such that any Turing machine computing an R_1-representation of α_s will have to give very large outputs, that is, outputs whose length is not of order $O(s)$ (and thus the Turing machine cannot run in time $O(s)$). Growth arguments tend to be easier, or at least less tedious, than diagonalization arguments, and we will present a rather detailed proof based on a growth argument in Section 8. But we cannot prove $\mathcal{D} \not\preceq_S \mathcal{C}$ by a growth argument as a Turing machine computing a Dedekind cut gives outputs of length 1. Let us see how we can prove $\mathcal{D} \not\preceq_S \mathcal{C}$ by a diagonalization argument.

Let s be an arbitrary time-bound. We will, by standard diagonalization techniques, construct a Cauchy sequence for an irrational α in the interval $(0,1)$ such that α becomes different from each $\beta \in O(s)_{\mathcal{D}}$, and hence we will have $\alpha \notin O(s)_{\mathcal{D}}$. Our construction can be carried out by a Turing machine, that is, the Cauchy sequence for α can be computed by a Turing machine. That Turing machine will run in time $O(t)$ for some time-bound t, and thus we have $\alpha \in O(t)_{\mathcal{C}}$. Moreover, it will turn out that t does not depend on s. Hence, we have a time-bound t and for every time-bound s there exists α such that $\alpha \in O(t)_{\mathcal{C}}$ and $\alpha \notin O(s)_{\mathcal{D}}$. Hence, we have t such that $O(t)_{\mathcal{C}} \not\subseteq O(s)_{\mathcal{D}}$ for every s, and thus, by Definition 1.4, we can conclude that $\mathcal{D} \not\preceq_S \mathcal{C}$.

We will now give an algorithm for computing a Cauchy sequence C such that $\lim_n C(n) \notin O(s)_{\mathcal{D}}$ where s is an arbitrary time-bound. We will need a standard enumeration $\{e\}_{e \in \mathbb{N}}$ of the Turing machines, and we use $\{e\}(x)$ to denote the execution of the e'th Turing machine on input x. Furthermore, we need a strictly increasing time-bound function S that eventually dominates any time-bound of order $O(s)$, that is, for any s_0 of order $O(s)$ we have $s_0(m) < S(m)$ for all sufficiently large m. Such an S will always exist, and, moreover, it can be chosen so that $S(n) > 2n$ for all n. Note that by the definition of time-bound functions, we have a Turing machine that given n, computes $S(n)$ in at most $a_S \cdot S(n)$ steps for some constant a_S. We define the sequence $d_0, d_1, d_2, \ldots$ by $d_0 = 2$ and $d_{i+1} = S(d_i)$. Finally, we will need a standard computable bijection $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$. For any $i \in \mathbb{N}$, our algorithm needs to compute the unique $j, e \in \mathbb{N}$ such that $\langle j, e \rangle = i$.

The algorithm sets $C(1) := 1/2$. If $n > 1$, the algorithm checks if there

8

exists i such that $n = d_{3i+2}$. If such an i does not exist, the algorithm simply set $C(n) := C(n-1)$; if such i exists the algorithm finds the unique j, e such that $\langle j, e \rangle = i$ and sets

- $C(n) := C(n-1)$ if $\{e\}(C(n-1))$ does not terminate within n steps

- $C(n) := C(n-1) - 2^{-n}$ if $\{e\}(C(n-1))$ terminates within n steps and outputs 0

- $C(n) := C(n-1) + 2^{-n}$ if $\{e\}(C(n-1))$ terminates within n steps and outputs something other than 0.

It is clear that the algorithm indeed computes a Cauchy sequence for a real number in the interval $(0, 1)$, and it is also pretty easy to see that the length of the output $C(n)$ will be bounded by a function of order $O(n)$ if we represent numbers in binary form and code rationals in a reasonable way. We can w.l.o.g. assume that S eventually dominates any function of order $O(n)$ (we can just pick an S that increases fast enough). Thus, for all sufficiently large n, we have

$$\|C(n)\| < S(n) . \tag{1.1}$$

Now, let $\alpha = \lim_n C(n)$ and let $\beta \in O(s)_{\mathcal{D}}$. We will prove that $\alpha \neq \beta$.

The Dedekind cut of β can be computed by a Turing machine $\{e\}$ running in time $O(s)$. Thus, $S(\|x\|)$ will be an upper bound on the number of steps in the computation $\{e\}(x)$ when x is large. Pick a sufficiently large j and let $i = \langle j, e \rangle$ (we can make d_{3i} as big as we want by picking a big j). Our algorithm is designed so that we have

$$C(d_{3i}) = C(d_{3i+2} - 1) . \tag{1.2}$$

Hence, by (1.2) and (1.1), the number of steps in the computation $\{e\}(C(d_{3i+2} - 1))$ will be bounded by

$$S(\|C(d_{3i+2} - 1)\|) = S(\|C(d_{3i})\|) < S(S(d_{3i})) = d_{3i+2} .$$

Assume that the output of the computation $\{e\}(C(d_{3i+2} - 1))$ is 0. Then we have

$$C(d_{3i+2}) = C(d_{3i+2} - 1) - 2^{-d_{3i+2}}$$

but as $\{e\}$ computes the Dedekind cut of β, we have $\beta > C(d_{3i+2} - 1)$. It follows that

$$\alpha = \lim_n C(n) \leq C(d_{3i+2} - 1) - 2^{-d_{3i+2}} + \sum_{n > 3i+2} 2^{-d_n} < C(d_{3i+2} - 1) < \beta .$$

Assume that the output of the computation $\{e\}(C(d_{3i+2} - 1))$ is different from 0. Then we have

$$C(d_{3i+2}) = C(d_{3i+2} - 1) + 2^{-d_{3i+2}}$$

9

but as $\{e\}$ computes the Dedekind cut of β, we have $\beta < C(d_{3i+2}-1)$. It follows that

$$\alpha = \lim_n C(n) \geq C(d_{3i+2}-1) + 2^{-d_{3i+2}} - \sum_{n>3i+2} 2^{d_n} > C(d_{3i+2}-1) > \beta \ .$$

This proves that $\alpha \neq \beta$.

The same argument can be used to prove that the limit α is irrational: Among the Turing machines enumerated there is a machine e that computes the Dedekind cut of any given rational q; this computation can be done in linear time under a reasonable encoding of rationals, so we may assume that e has running time in $O(s)$; and we conclude that $\alpha \neq q$.

We have proved that our algorithm computes a Cauchy sequence for an irrational number which cannot be in the class $O(s)_{\mathcal{D}}$. Let us undertake a complexity analysis of a Turing machine M executing the algorithm:

- The input to M is a natural number n (we will estimate an upper bound for M's running time as a function of n).

- First M will recursively compute $C(n-1)$.

- Then M will check if there exists i such that $n = d_{3i+2}$. Recall that $d_0 = 2$ and $d_{i+1} = S(d_i)$ where S is a time-bound function. It is possible to check if such an i exists in time $O(n)$. Briefly, M computes $d_0, d_1, d_2 \ldots$ until it either hits j such that $n = d_j$, or j such that $n < d_j$, or j such that the computation of d_j exceeds $a_S n$ time (the upper bound that we have on this computation time is $a_S S(d_{j-1})$). In all cases, the computation of the last d_j is either completed or stopped after $O(n)$ steps. The computation of each of the previous elements of the sequence, needed for computing C(n), also takes $O(n)$ steps, and there are at most $\log n$ such elements, because the sequence d_i grows at least exponentially. We conclude that this computation runs in $O(n \log n)$ time.

- If M finds i such that $n = d_{3i+2}$, then M will compute j, e such that $\langle j, e \rangle = i$, check if the computation $\{e\}(C(n-1))$ terminates within n steps, and finally, compute the output. All this can be done in time $O(n^2)$ on a multi-tape Turing machine.

Since the computation of $C(n)$ takes time $O(n^2)$, provided we have $C(n-1)$, we conclude that M runs in time $O(\sum_{i=1}^{n} i^2) = O(n^3)$, where $n \in \mathbb{N}$ is the input, and thus in time $O(2^{3\|n\|})$ where $\|n\|$ is the length of the input (time complexity is always stated as a function of the input bit-length). This allows us to conclude that M computes a Cauchy sequence for an irrational in the class $O(2^{3\|n\|})_{\mathcal{C}}$. We also know that this irrational is not in the class $O(s)_{\mathcal{D}}$, and recall that s was an arbitrary chosen time-bound. Hence, we have $O(2^{3\|n\|})_{\mathcal{C}} \nsubseteq O(s)_{\mathcal{D}}$ for any time-bound s (note that the witness for the non-containement depends on s, and further its computation time depends on the constant a_S, but it is always in $O(2^{3\|n\|})$). This proves that $\mathcal{D} \nleq_S \mathcal{C}$.
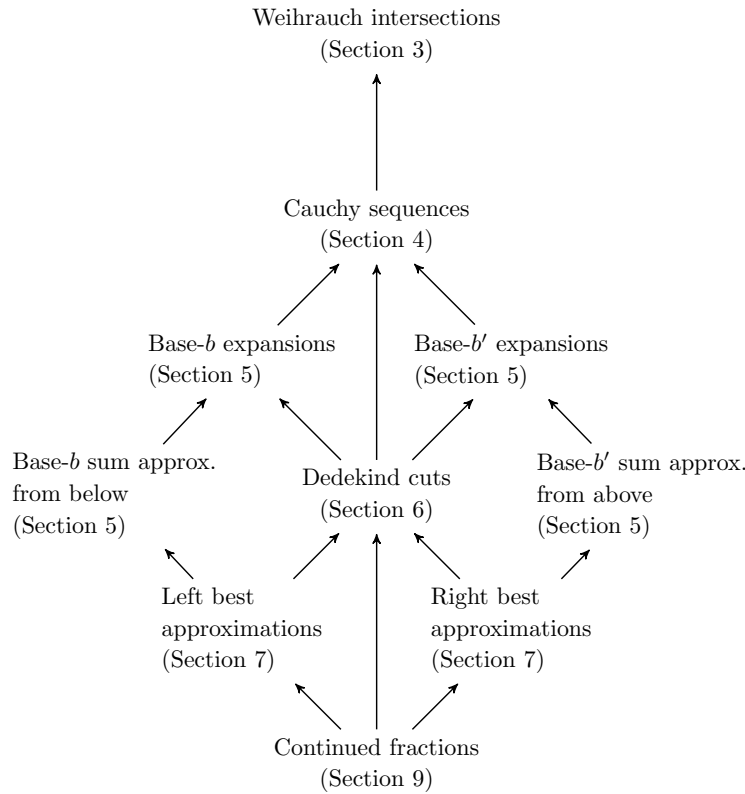
10

Figure 1: Subrecursive degrees (equivalence classes) of representations.

Our proof that $\mathcal{D} \not\preceq_S \mathcal{C}$ is meant to illustrate how our definitions work. In the current paper we will in general not formally prove results of the form $R_1 \not\preceq_S R_2$, but for the benefit of the reader we will to a certain extent provide informal explanations and intuitive arguments of why it is impossible to subrecursively convert an R_2-representation into an R_1-representation (full proofs may be long and involved, and they are available elsewhere, see Section 1.4).

Definition 1.5. Let R_1 and R_2 be representations. The relation $R_1 \equiv_S R_2$ holds when $R_1 \preceq_S R_2$ and $R_2 \preceq_S R_1$. If the relation $R_1 \equiv_S R_2$ holds, we will say that the representation R_1 is *subrecursively equivalent* to the representation R_2.

The relation $R_1 \prec_S R_2$ holds when $R_1 \preceq_S R_2$ and $R_2 \not\preceq_S R_1$. $\qquad\qquad \square$

The equivalence relation $\equiv_S$ induces a degree structure on the representations. The directed graph in Figure 1 gives an overview of the relationship between some natural degrees (equivalence classes). The nodes depict degrees

11

of representations, and each degree is labeled with one of the most well-known representations in the degree. For two representations R_1 and R_2, there is a directed path from a node labeled R_1 to a node labeled R_2 if and only if $R_2 \prec_S R_1$. Thus, if there is a directed path from R_1 to R_2, we can subrecursively convert an R_1-representation into an R_2-representation, and if there is no directed path from R_1 to R_2, we cannot subrecursively convert an R_1-representation into an R_2-representation. Unfortunately we are not able to accurately depict the complex relationship between the degrees of the base-b expansions and the degrees of the base-b sum approximations from below and above (for $b = 2, 3, 4, \ldots$), but our graph gives a rough idea of what this world looks like. See Section 5 for more on how these degrees relate to each other.

1.4 Our goals and some references

We present a (degree) theory of representations (of irrational numbers) which is based on Turing machines and standard complexity theory. This theory should be considered as a recast and an improvement of the theory developed in Kristiansen [31] [32] and Georgiev et al. [20] which is based on honest functions and subrecursive classes. The two approaches studying representations of reals and conversions between them are essentially the same, even if, e.g., the reducibility relation $\preceq_S$ is never formally defined in any other paper. It follows more or less straightforwardly from results proved in [31] [32] [20] that the picture drawn in Figure 1 is correct.

Recently Georgiev [18] has identified some interesting degrees which are not depicted in Figure 1, moreover, Ben-Amram & Kristiansen [7] have proved that the structure is a distributive lattice with a minimal and a maximal degree. Two other recent papers which shed light on the degree structure, are Georgiev [19] and Hiroshima & Kawamura [22].

In papers like [31] [32] [20], and furthermore Georgiev [18] [19] and Kristiansen [33], the authors are just concerned with the existence or inexistence of subrecursive conversions. In cases where the relation $R_1 \preceq_S R_2$ holds, they never analyze the computational complexity of converting an R_2-representation into an R_1-representation, and they do not make any effort to find efficient conversions. In this paper we will care about such matters. Indeed, from now on such matters will be our primary concern: We will prove that the relation $R_1 \preceq_S R_2$ holds by imposing tight upper bounds on the running time of an oracle Turing machine which converts an R_2-representation into an R_1-representation. We will also give upper bounds on the number of oracle calls required and the size of those calls.

We will impose resource bounds on oracle Turing machines that convert irrational numbers in the interval $(0, 1)$. The reader should note that many of our results can be easily generalized to oracle Turing machines converting arbitrary real numbers (including rationals). Moreover, many of these generalizations are trivial. We leave the details to the interested reader.

12

1.5 Overview of the paper

In Section 2 we provide the necessary preliminaries on complexity theory, oracle Turing machines and the Stern-Brocot tree. In Section 3, we introduce the representation by Weihrauch intersections. The equivalence class of this representation will be the minimal degree in the degree structure depicted in Figure 1 (a detailed proof can be found in [7]).

In Section 4 we study conversions between representations subrecursively equivalent to the representation by Cauchy sequences. In Section 5 we study conversions between representations subrecursively equivalent to representations by base-b expansions and base-b sum approximations. In Section 6 we treat representations subrecursively equivalent to the representation by Dedekind cuts, and thereafter, in Section 7, representations subrecursively equivalent to the representation by left/right best approximations.

The degree structure will be discussed further in Section 8. Finally, in Section 9, we study conversions between representations subrecursively equivalent to the representation by continued fractions. It is proved in [7] that equivalence class of the continued fractions is the maximal degree in our degree structure.

1.6 Acknowledgements

The authors are deeply grateful to two anonymous referees for their tremendous jobs. The authors also want to thank Dag Normann for helpful comments and enlightening discussions which gave birth to some of the representations studied in this paper.

2 Preliminaries

2.1 Oracle Turing machines and complexity theory

We assume basic familiarity with computability and computational complexity (standard textbooks are Sipser [38], Du & Ko [16] and Arora & Barak [1]).

We will work with Turing machines with oracle access to the representation being converted from. Unless otherwise stated, elements of $\mathbb{N}$ are assumed to be written on input, query, and output tapes in their binary representation, least-significant bit first. Pairs (p, q) of integers are assumed to be written using interleaved notation (i.e., the first bit of the binary representation of p followed by the first bit of the binary representation of q, and so forth). Observe that the length of the representation of a pair (p, q) is then $O(\log \max\{p, q\})$. Elements $p/q \in \mathbb{Q}$ are assumed to be represented by the representation of (p, q). We denote the length of the binary representation of x by $\|x\|$.

Function-oracle machines are in standard use in complexity theory of functions on the set of real numbers (see, e.g., Ko [27]), and the next definition is a standard one.

13

Definition 2.1. A (parameterized) *function-oracle Turing machine* is a (multi-tape) Turing machine $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ with initial state $q_0 \in Q$, final states $F \subseteq Q$, input and tape alphabets Σ and Γ (with $\Sigma \subseteq \Gamma$ and $\{\_\} \subseteq \Gamma \setminus \Sigma$), and partial transition function δ such that M has a special *query tape* and two distinct states $q_q, q_a \in Q$ (the *query* and *answer* states).

To be executed, M is provided with a total function $f : (\Gamma \setminus \{\_\})^* \longrightarrow (\Gamma \setminus \{\_\})^*$ (the oracle) prior to execution on any input. We write M^f for M when f has been fixed. We use Φ_M^f to denote the function computed by M^f.

The transition relation of M^f is defined as usual for Turing machines, except for the query state q_q: If M enters state q_q with the word x on its query tape, then (i) the contents of the query tape are instantaneously changed to $f(x)$, (ii) the query-tape head is reset to the origin, while other heads do not move, and (iii) M moves to state q_a. The *time and space complexity* of a function-oracle machine is counted as for usual Turing machines, with the transition between q_q and q_a taking $\|f(x)\|$ time steps. The *input size* of a query is the number of non-blank symbols on the query tape when M enters state q_q. $\qquad\square$

In other work on real number computation, there is a well-developed notion of reducibility between representations that, roughly, requires the representation to be written as an infinite string on one of the input tapes of a type-2 Turing machine [30, 43, 40]. In that setting, e.g., a function $f : \mathbb{Q} \cap [0,1] \longrightarrow \{0, \ldots, b-1\}$ is most naturally expressed by imposing a computable ordering on its domain (e.g., rationals appear in non-decreasing order of their denominator), and the function values $f(q)$ appear encoded as bit strings in this order. We strongly conjecture that our results carry over to the type-2 setting *mutatis mutandis*.

2.2 Some notation

We write $f(n) = \text{poly}(n)$ if $f : \mathbb{N} \longrightarrow \mathbb{N}$ is bounded above by a polynomial in n with positive integer coefficients, and $f(n) = \text{polylog}(n)$ if f is bounded above by a polynomial in $\log n$ with positive integer coefficients.

We use the notation $f^{(n)}$ for the nth iterate of the function $f : \mathbb{N} \longrightarrow \mathbb{N}$, that is, $f^{(0)}(x) = x$ and $f^{(n+1)} = f \circ f^{(n)}$. Note the parentheses in the superscript position, that distinguish this notation from ordinary exponentiation. This notation is often used in conjunction with λ-notation, e.g.

$$(\lambda x.g(x))^{(4)}(0) = (\lambda x.g(x))(\lambda x.g(x))^{(3)}(0)$$
$$= g((\lambda x.g(x))^{(3)}(0)) = \cdots = g(g(g(g(0)))) \, .$$

2.3 Farey sequences and the Stern-Brocot tree

A *Farey sequence* is a strictly increasing sequence of fractions between 0 and 1. The Farey sequence of *order k*, denoted F_k, contains all fractions which when

14

written in their lowest terms, have denominators less than or equal to k. Thus, e.g., F_5 is the sequence

$$0/1 \,,\, 1/5 \,,\, 1/4 \,,\, 1/3 \,,\, 2/5 \,,\, 1/2 \,,\, 3/5 \,,\, 2/3 \,,\, 3/4 \,,\, 4/5 \,,\, 1/1 \,.$$

The ordered pair of two consecutive fractions in a Farey sequence is called a *Farey pair*. Let $(a/b, c/d)$ be a Farey pair. The fraction $(a+c)/(b+d)$ is called the *mediant* of a/b and c/d. The next theorem was originally proved by Cauchy [13] in 1826.

Theorem 2.2. *Let $(a/b, c/d)$ be a Farey pair. (i) We have $cb - ad = 1$ (or, equivalently $c/d - a/b = 1/(bd)$); (ii) The mediant $(a+c)/(b+d)$ is in its lowest terms and lies strictly between a/b and c/d, moreover, every other fraction lying strictly between a/b and c/d has denominator strictly greater than $b + d$.*

E.g., $(1/3, 2/5)$ is a Farey pair as $1/3$ and $2/5$ are neighbors in the sequence F_5 (see above). The mediant of $1/3$ and $2/5$ is $3/8$. Thus, $3/8$ lies in the open interval $(1/3, 2/5)$, and any fraction in this open interval, with the exception of $3/8$, has denominator strictly greater than 8. For more on Farey pairs and Farey sequences, see Hardy & Wright [21].

We arrange the fractions strictly between 0 and 1 in a binary search tree $\mathcal{T}_{\mathrm{F}}$.

Definition 2.3. The *Farey pair tree* $\mathcal{T}_{\mathrm{F}}$ is the complete infinite binary tree where each node has an associated Farey pair $(a/b, c/d)$ defined by recursion on the position $\sigma \in \{0, 1\}^*$ of a node in $\mathcal{T}_{\mathrm{F}}$ as follows: $\mathcal{T}_{\mathrm{F}}(\epsilon) = (0/1, 1/1)$, and if $\mathcal{T}_{\mathrm{F}}(\sigma) = (a/b, c/d)$, then $\mathcal{T}_{\mathrm{F}}(\sigma 0) = (a/b, (a+c)/(b+d))$ and $\mathcal{T}_{\mathrm{F}}(\sigma 1) = ((a+c)/(b+d), c/d)$. The *depth* of a node in $\mathcal{T}_{\mathrm{F}}$ is the length of its position (with the depth of the root node being 0).

Abusing notation slightly, we do not distinguish between the pair $\mathcal{T}_{\mathrm{F}}(\sigma) = (a/b, c/d)$ and the open interval $(a/b, c/d)$. $\qquad \square$

Thus, we have, for example

$$\mathcal{T}_{\mathrm{F}}(0) = \left(\frac{0}{1}, \frac{1}{2}\right), \, \mathcal{T}_{\mathrm{F}}(1) = \left(\frac{1}{2}, \frac{1}{1}\right), \, \mathcal{T}_{\mathrm{F}}(10) = \left(\frac{1}{2}, \frac{2}{3}\right), \, \mathcal{T}_{\mathrm{F}}(0000) = \left(\frac{0}{1}, \frac{1}{5}\right) \,.$$

The infinite binary tree obtained from the Farey pair tree by replacing each Farey pair $(a/b, c/d)$ by its mediant $(a + c)/(b + d)$, is known as the (left) *Stern-Brocot tree*[2] (we will not use the Stern-Brocot tree directly). Efficient computation of the elements of the Stern-Brocot tree (and hence also the Farey pair tree) is possible, see Bates et al. [4]; for our purposes, we simply need the next proposition.

Proposition 2.4. *There is a Turing machine M such that for any $\sigma \in \{0, 1\}^*$, $\Phi_M(\sigma) = \mathcal{T}_{\mathrm{F}}(\sigma)$ and M runs in time $\mathrm{poly}(1 + |\sigma|)$.*

---

[2] "Left" because the Stern-Brocot tree originally concerns the interval $(0, \infty)$ and we are interested only in $(0, 1)$ which corresponds to the left child of the Stern-Brocot tree.

15

We round off this section by stating and proving a few properties of the Farey pair tree.

Proposition 2.5. *If $(a/b, c/d)$ is a Farey pair at depth h in $\mathcal{T}_\mathrm{F}$, then $a + b + c + d \geq h + 3$.*

Proof. For $h = 0$, we have $0 + 1 + 1 + 1 = 3 = h + 3$.

Let $h > 0$. Assume the proposition for $h-1$ and let $(a/b, c/d)$ be an arbitrary pair at depth $h-1$. Then a pair at level h is of the form (i) $(a/b, (a+c)/(b+d))$ or of the form (ii) $((a+c)/(b+d), c/d)$. In case (i), we have

$$
\begin{aligned}
a + (a+c) + b + (b+d) \;=\; (a+b+c+d) + (a+b) \;&\geq\; (a+b+c+d) + 1 \\
&\geq\; (h+2) + 1 = h+3 \,.
\end{aligned}
$$

A symmetric argument will show that the proposition also holds in case (ii). □

Proposition 2.6. *Let $p/q \in \mathbb{Q} \cap (0,1]$ be a fraction in its lowest terms. Then, p/q is a fraction in a Farey pair at depth at most $p + q - 2$ in $\mathcal{T}_\mathrm{F}$.*

Proof. By construction, for any depth $n \geq 0$, the set of intervals $[a/b, c/d]$ occurring in $\mathcal{T}_\mathrm{F}$ at depth n covers the unit interval, and each pair of intervals has at most one point in common (which must be an end point). Hence, p/q occurs in some interval $[a/b, c/d]$ at any depth n, and by Theorem 2.2 we have $cb - ad = 1$. Assume for the sake of a contradiction that $n \geq p + q - 2$ and $p/q \notin \{a/b, c/d\}$. Since $p/q \notin \{a/b, c/d\}$, we have $a/b < p/q < c/d$, and thus also $1 \leq pb - qa$ and $1 \leq qc - pd$. Hence

$$
\begin{aligned}
a + b + c + d &\leq (a+b)(qc - pd) + (c+d)(pb - qa) \\
&= p(cb - ad) + q(cb - ad) = p + q \,.
\end{aligned}
$$

This contradicts Proposition 2.5 which implies that $a + b + c + d \geq p + q + 1$. Hence, p/q must occur as an endpoint, and the first level at which p/q appears as an endpoint must be at most $p + q - 2$. □

Lemma 2.7. *Let $I = (a_n/b_n, c_n/d_n)$ be a Farey interval at depth n in $\mathcal{T}_\mathrm{F}$, and for $i = 0, \ldots, n$, let $(a_i/b_i, c_i/d_i)$, denote the Farey pairs along the path from the root to I. Then $a_i, b_i, c_i, d_i \leq (n+3)(a_n + b_n)$.*

Proof. We claim that for all $i \leq n$, $a_i + b_i + c_i + d_i \leq (i+3)(a_n + b_n)$. For $i = 0$, we have $a_0 + b_0 + c_0 + d_0 = 3(a_0 + b_0) \leq 3(a_n + b_n)$. Assume this holds for arbitrary $i < n$. If the next node is a left child, then

$$
\begin{aligned}
a_{i+1} + b_{i+1} + c_{i+1} + d_{i+1} &= a_i + b_i + (a_i + c_i) + (b_i + d_i) \\
&\leq a_i + b_i + (i+3)(a_n + b_n) \leq (i+4)(a_n + b_n) \,.
\end{aligned}
$$

16

If the next node is a right child, then

$$
\begin{aligned}
a_{i+1} + b_{i+1} + c_{i+1} + d_{i+1} &= (a_i + c_i) + (b_i + d_i) + c_i + d_i \\
&\le (i+3)(a_n + b_n) + c_i + d_i \le (i+3)(a_n + b_n) + a_{i+1} + b_{i+1} \\
&\le (i+4)(a_n + b_n) \, .
\end{aligned}
$$

$\square$

3   Weihrauch intersections

Definition 3.1. A function $I : \mathbb{N} \longrightarrow \mathbb{Q} \times \mathbb{Q}$ is a *Weihrauch intersection* for the real number α if the left component of the pair $I(i)$ is strictly less than the right component of the pair $I(i)$ (for all $i \in \mathbb{N}$) and

$$
\{\, \alpha \,\} \; = \; \bigcap_{i=0}^{\infty} I_i^O
$$

where I_i^O denotes the open interval given by the pair $I(i)$.

Theorem 3.2. *Any computable real number can be represented by a polynomial-time computable Weihrauch intersection.*

Proof. A computable real number α has a computable Cauchy sequence $C : \mathbb{N} \longrightarrow \mathbb{Q}$ with the property $|C(n) - \alpha| < 2^{-n}$. Let M be a Turing machine computing C. We can w.l.o.g. assume that $\alpha \in (0,1)$.

Compute $I(k)$ by the following algorithm: Find the greatest n such that $n \le \|k\|$ and $C(n)$ can be computed by M in at most $\|k\|$ steps. Let $I(k) = (C(n) - 2^{-n}, C(n) + 2^{-n})$. Let $I(k) = (0,1)$ if no such n exists (it is possible to arrange this so that we have $I_{k+1}^O \subseteq I_k^O$). This gives a Weihrauch intersection for α because $I(k)$ assumes values $(C(n) - 2^{-n}, C(n) + 2^{-n})$ for infinitely many n. $\square$

The representation by Weihrauch intersections is one of the main representations in Weihrauch's seminal book [41], and it is special among the representations we consider in this paper: There exists a time-bound t such that every computable real has a Weihrauch intersection computable by a Turing machine running in time $O(t)$ (by Theorem 3.2, this will for sure be true for any t that dominates all polynomials). For every other representation R considered in this paper, there will for any time-bound t exist a time bound s such that $O(t)_R$ is strictly included in $O(s)_R$. The degree of the representation by Weihrauch intersections will be the zero degree of the degree structure described in Section 1.

Representation of reals by Weihrauch intersections is also known as representation by *nested intervals*. In order to simplify our definition, we have not

17

required the intervals to be nested, but any Weihrauch intersection can be easily converted to a nested one. A number of subrecursively equivalent representations can be found in [41], but they are all pretty similar from our point of view, and we will not discuss any of them.

4 Representations subrecursively equivalent to Cauchy sequences

4.1 Cauchy sequences.

Definition 4.1. Let $\alpha \in (0,1)$ be an irrational number. Then $C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ is a *Cauchy sequence* for α if for all $n > 0$, $|\alpha - C(n)| < n^{-1}$. $\qquad\square$

Lemma 4.2. *There is a parameterized function-oracle Turing machine M with the following properties. Let $C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ be a Cauchy sequence for an irrational number $\alpha \in (0,1)$. Then*

- $\Phi_M^C : \mathbb{N} \longrightarrow \mathbb{Q} \times \mathbb{Q}$ *is a Weihrauch intersection for α*

- M^C *on input n runs in time* $\mathrm{polylog}(\max\{C(n), n\})$ *and uses exactly one oracle call of input size* $O(\log n)$.

Proof. Let $I(n) = (C(n) - n^{-1}, C(n) + n^{-1})$. Then, I is a Weihrauch intersection for α if C is a Cauchy sequence for α. Hence, only one oracle call to C is needed, and $I(n)$ can be obtained by basic arithmetic operations on (the binary representations of) $C(n)$ and n. $\qquad\square$

Lemma 4.2 shows that a Cauchy sequence can be subrecursively converted into a Weihrauch intersection. *Will it be possible to subrecursively convert a Weihrauch intersection into a Cauchy sequence?* In order to give a negative answer to that question, we need a presumably rather well-known theorem.

Theorem 4.3. *For any time-bound t there exists a computable irrational number α such that no Cauchy sequence for α can be computed by a Turing machine running in time $O(t)$.*

It is not hard to see that the theorem holds. Let t be an arbitrary time bound, and let $t'(m) = 2^{t(2m+3)}$. Let A be any set of natural numbers such that membership in A can be decided by a Turing machine, but not by an $O(t')$-time Turing machine (the existence of such a set can be shown by a standard diagonalization argument). Consider the irrational number α given by the base-2 expansion $0.a_1 a_2 a_3 \ldots$ where the two digits $a_{2i-1} a_{2i}$ are 10 if $i \in A$; and 01 otherwise. Now, α will obviously have a computable Cauchy sequence, but no $O(t)$-time Turing machine can compute such a Cauchy sequence. If a Cauchy sequence for α can be computed in time $O(t)$, then a Turing machine M can decide if m is in the set A in time $O(t')$: First M computes $C(2^{2m+2})$. By

18

assumption this can be done in time $O(t(\|2^{2m+2}\|)) = O(t(2m+3))$. Thereafter, M determines the digits $0.\mathtt{D}_1\mathtt{D}_2\ldots$ of the base-2 expansion of $C(2^{2m+2})$. Observe that $C(2^{2m+2})$ lies sufficiently close to α to ensure that the digit $\mathtt{D}_{2m-1}$ coincides with digit a_{2m-1}, and thus, $m \in A$ iff $\mathtt{D}_{2m-1} = 1$. Hence, M can decide if m is a member of A by computing $\mathtt{D}_{2m-1}$, and this can obviously be done in time polynomial in $t(2m + 3)$, hence in time $O(t')$. Since no Turing machine can decide membership in A in time $O(t')$, we can conclude that the theorem holds.

Theorem 3.2 states that any computable real can be represented by a polynomial-time computable Weihrauch intersection. Thus, any computable real can be represented by a Weihrauch intersection computable in, let us say, time $O(2^n)$. Now, 2^n is a fixed time-bound, and if it were possible to subrecursively convert a Weihrauch intersection into a Cauchy sequence, the any computable irrational would be represented by a Cauchy sequence computable in time $O(t)$ for some fixed time-bound t. By Theorem 4.3, such a t does not exist, and we can conclude that Weihrauch intersections cannot be subrecursively converted into Cauchy sequences.

4.2 Definitions

The next definition gives some representations which are subrecursively equivalent to the representation by Cauchy sequences.

Definition 4.4. Let $\alpha \in (0, 1)$ be an irrational number.

1. $C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ is a *strictly increasing Cauchy sequence* for α if (i) C is a Cauchy sequence for α and (ii) $C(n) < C(n + 1)$.

2. Let $b \geq 2$ be a natural number. Then, $A : \mathbb{N}^+ \longrightarrow \mathbb{Z}$ is a *converging base-b sequence* for α if $A(n)b^{-n}$ is a Cauchy sequence for α.

3. $D : \mathbb{Z} \times \mathbb{N}^+ \longrightarrow \{0, 1\}$ is a *fuzzy (Dedekind) cut* for α if

$$D(p,q) = 0 \ \Rightarrow \ \alpha < \frac{p+1}{q} \qquad \text{and} \qquad D(p,q) = 1 \ \Rightarrow \ \frac{p-1}{q} < \alpha \ .$$

4. $S : \mathbb{N}^+ \longrightarrow \{-1, 0, 1\}$ is a *signed digit expansion* for α if

$$\alpha = \sum_{i=1}^{\infty} S(i)2^{-i} \ .$$

$\square$

Converging base-2 sequences are used in Friedman and Ko [28] and also in the monograph Ko [27]. Signed digit expansions also seem to be well known. The representation was introduced for the first time by Avizienis [3] and appears in several rather recent papers, eg. Berger et al. [8] and Bauer et al. [5]. It is also discussed in Weihrauch's book [41]. The representations by strictly increasing Cauchy sequences and fuzzy Dedekind cuts are discussed for the first time in this paper.

19

4.3 Cauchy sequences to fuzzy cuts

Let C be a Cauchy sequence for α. We define the map $D : \mathbb{Z} \times \mathbb{N}^+ \longrightarrow \{0, 1\}$ by

$$D(p, q) \;=\; \begin{cases} 0 & \text{if } C(q) \le pq^{-1} \\ 1 & \text{if } C(q) > pq^{-1}. \end{cases}$$

Lemma 4.5. *D is a fuzzy cut for α.*

Proof. First we prove

$$D(p, q) = 0 \;\Rightarrow\; \alpha < \frac{p+1}{q} \;. \tag{4.1}$$

Assume $D(p, q) = 0$. By the definition of a Cauchy sequence, we have $\alpha < C(q) + q^{-1}$. By the definition of D, we have

$$\alpha \;<\; C(q) + \frac{1}{q} \;\le\; \frac{p}{q} + \frac{1}{q} \;=\; \frac{p+1}{q} \;.$$

This proves (4.1). We also need to prove

$$D(p, q) = 1 \;\Rightarrow\; \frac{p-1}{q} < \alpha \;. \tag{4.2}$$

The proof of (4.2) is symmetric to the proof of (4.1). The lemma follows from (4.1) and (4.2). $\square$

Lemma 4.6. *There is a parameterized function-oracle Turing machine M with the following properties. Let $C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ be a Cauchy sequence for an irrational number $\alpha \in (0, 1)$. Then*

- *$\Phi_M^C : \mathbb{Z} \times \mathbb{N}^+ \longrightarrow \{0, 1\}$ is a fuzzy Dedekind cut for α*

- *M^C on input (p, q) runs in time $\mathrm{poly}(\max(\|p\|, \|q\|, \|C(q)\|))$ and uses a single oracle call of input size at most $O(\log q)$.*

Proof. A single call to C on input (the binary representation of) q yields (the binary representation of) $C(q)$, and by Lemma 4.5, a single comparison of $C(q)$ to pq^{-1} yields $D(p, q)$. A binary representation of the rational number pq^{-1} can be computed in polynomial time in the size of the representations of p and q (that is, in time $\mathrm{polylog}(\max\{p, q\})$), and the final comparison of $C(q)$ and pq^{-1} can be performed in time polynomial in $\max\{\|C(q)\|, \|pq^{-1}\|\}$. $\square$

4.4 Fuzzy cuts to signed digit expansions

Let $\alpha \in (0, 1)$, and let D be a fuzzy cut for α. We will use D to define a signed digit expansion S.

20

For any map $S : \mathbb{N}^+ \longrightarrow \{-1, 0, 1\}$, let $s_n = \sum_{i=1}^n S(i)2^{-i}$. Furthermore, let a_n be the unique integer such that $a_n 2^{-n} = s_n$, and let $M(r_1, r_2)$ denote the midpoint between the two rationals r_1 and r_2, that is, $M(r_1, r_2) = (r_1 + r_2)/2$. Observe that

$$M\left(s_n - \frac{1}{2^{n+1}}, s_n\right) = \frac{4a_n - 1}{2^{n+2}} \quad \text{and} \quad M\left(s_n, s_n + \frac{1}{2^{n+1}}\right) = \frac{4a_n + 1}{2^{n+2}}$$

Moreover, observe that $s_n = 2a_n/2^{n+1}$.

We define the signed digit expansion S by $S(1) = 1$ and

$$S(n+1) \;\; = \;\; \begin{cases} -1 & \text{if } D(2a_n, 2^{n+1}) = 0 \text{ and } D(4a_n - 1, 2^{n+2}) = 0 \text{ (Case 1)} \\ 0 & \text{if } D(2a_n, 2^{n+1}) = 0 \text{ and } D(4a_n - 1, 2^{n+2}) = 1 \text{ (Case 2)} \\ 1 & \text{if } D(2a_n, 2^{n+1}) = 1 \text{ and } D(4a_n + 1, 2^{n+2}) = 1 \text{ (Case 3)} \\ 0 & \text{if } D(2a_n, 2^{n+1}) = 1 \text{ and } D(4a_n + 1, 2^{n+2}) = 0 \text{ (Case 4)} \end{cases}$$

Lemma 4.7.

$$\alpha \;=\; \sum_{i=1}^\infty S(i)2^{-i} \; .$$

Proof. We will prove

$$s_n - 2^{-n} < \alpha < s_n + 2^{-n} \tag{4.3}$$

by induction on n. The lemma follows straightforwardly from (4.3). It is obvious that (4.3) holds when $n = 1$ (as we have assumed $\alpha \in (0, 1)$).

Assume by induction hypothesis that (4.3) holds. We need to prove that

$$\alpha \in \left(s_{n+1} - 2^{-(n+1)}, \, s_{n+1} + 2^{-(n+1)}\right) \tag{4.4}$$

(Case 1.) In this case we have $s_{n+1} = s_n - 2^{-(n+1)}$. Thus, in order to prove (4.4), we need to prove $\alpha \in (s_n - 2^{-n}, s_n)$. We have $s_n - 2^{-n} < \alpha$ by the induction hypothesis (4.3). Moreover, since $D(4a_n - 1, 2^{n+2}) = 0$, we have

$$\alpha \;<\; \frac{4a_n - 1 + 1}{2^{n+2}} \;=\; \frac{a_n}{2^n} \;=\; s_n \; .$$

This proves that (4.4) holds in (Case 1).

(Case 2.) In this case we have $s_{n+1} = s_n$. Thus, in order to prove (4.4), we need to prove $\alpha \in (s_n - 2^{-(n+1)}, s_n + 2^{-(n+1)})$. Since $D(2a_n, 2^{n+1}) = 0$, we have

$$\alpha \;<\; \frac{2a_n + 1}{2^{n+1}} \;=\; \frac{a_n}{2^n} + \frac{1}{2^{n+1}} \;=\; s_n + 2^{-(n+1)} \; .$$

Since $D(4a_n - 1, 2^{n+2}) = 1$, we have

$$\alpha \;>\; \frac{4a_n - 1 - 1}{2^{n+2}} \;=\; \frac{a_n}{2^n} - \frac{1}{2^{n+1}} \;=\; s_n - 2^{-(n+1)} \; .$$

This proves that (4.4) holds in (Case 2).

(Case 3) is symmetric to (Case 1), and (Case 4) is symmetric to (Case 2). $\quad\square$

21

Lemma 4.8. *There is a parameterized function-oracle Turing machine M with the following properties. Let $D : \mathbb{Z} \times \mathbb{N}^+ \longrightarrow \{0,1\}$ be a fuzzy cut for an irrational number $\alpha \in (0,1)$. Then*

- *$\Phi_M^D : \mathbb{N}^+ \longrightarrow \{-1,0,1\}$ is a signed digit expansion of α*

- *M^D on input n runs in time $\mathrm{poly}(n)$ and uses $2(n-1)$ oracle calls of input size at most $O(n)$.*

Proof. By Lemma 4.7, $S(1) = 1$, and computing $S(n)$ for $n > 1$ can be done by computing the integer a_{n-1} satisfying $a_{n-1}2^{-(n-1)} = s_{n-1} = \sum_{i=1}^{n-1} S(i)2^{-i}$, and subsequently performing oracle calls returning the values of

$$D(2a_{n-1}, 2^n) \ \text{ and either } \ D(4a_{n-1}-1, 2^{n+1}) \ \text{ or } \ D(4a_{n-1}+1, 2^{n+1}) \, . \quad (4.5)$$

Observe that the binary representations of the rational numbers $2^{-1}, \ldots, 2^{-(n-1)}$ have length $O(n)$, and that each representation can be computed in time $\mathrm{poly}(n)$. Hence, if $S(1), \ldots, S(n-1)$ are known, then s_{n-1} can be computed in time $\mathrm{poly}(n)$ using standard arithmetical operations. Furthermore, observe that the binary representation of s_{n-1} (and thus a_{n-1}) has length $O(n)$. This implies that (i) the size of the oracle calls in (4.5) is at most $O(n)$, and that (ii) a_{n-1}, $2a_{n-1}$ and $4a_{n-1}$ can be computed in time $\mathrm{poly}(n)$.

Using the obvious recursive algorithm for $S(n)$ requires computing $S(1), \ldots, S(n-1)$, hence time $O(n\mathrm{poly}(n)) = \mathrm{poly}(n)$, and a total of $2(n-1)$ oracle calls, each of size $O(n)$. □

4.5   Signed digit expansions to Cauchy sequences

Let S be a signed digit expansion of an irrational number $\alpha \in (0,1)$. Then we have $\left|\alpha - \sum_{i=1}^n S(i)2^{-i}\right| < 2^{-n}$. Let

$$C(n) \ = \ \sum_{i=1}^{\lceil \log_2 n \rceil} S(i)2^{-i}$$

then we have

$$|\alpha - C(n)| \ = \ \left|\alpha - \sum_{i=1}^{\lceil \log_2 n \rceil} S(i)2^{-i}\right| \ < \ 2^{-\lceil \log_2 n \rceil} \ \leq \ n^{-1} \, .$$

Hence, C is a Cauchy sequence for α.

Lemma 4.9. *There is a parameterized function-oracle Turing machine M with the following properties. Let $S : \mathbb{N}^+ \longrightarrow \{-1,0,1\}$ be a signed digit expansion for an irrational number $\alpha \in (0,1)$. Then*

- $\Phi_M^S : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ *is a Cauchy sequence for* α

- M^S *on input* n *runs in time* $\mathrm{polylog}(n)$ *and uses* $\lceil \log_2 n \rceil$ *oracle calls of input size at most* $1 + \log \log n$.

Proof. The result follows almost immediately from the text just prior to the lemma. Observe that the computation of $C(n) = \sum_{i=1}^{\lceil \log_2 n \rceil} S(i) 2^{-i}$ can be performed using $\mathrm{poly}(\log n) = \mathrm{polylog}(n)$ operations on rationals whose representation has length at most $O(\log n)$, hence in total time $\mathrm{polylog}(n)$. $\qquad\square$

4.6 From Cauchy sequences to strictly increasing Cauchy sequences

Let C be a Cauchy sequence for some real number α. Thus, for all n, we have

$$\alpha \in (\ C(2^n) - 2^{-n}, \ C(2^n) + 2^{-n}\ ) \tag{4.6}$$

We will use $C(n)_\ell$ to denote the left endpoint of the interval in (4.6), that is, $C(n)_\ell = C(2^n) - 2^{-n}$, and we define $\widehat{C}$ by

$$\widehat{C}(n) \ = \ C(n+2)_\ell - 2^{-(n+1)} \ = \ C(2^{n+2}) - 2^{-(n+2)} - 2^{-(n+1)}\ .$$

Lemma 4.10. *If* C *is a Cauchy sequence for* α, *then* $\widehat{C}$ *is a strictly increasing Cauchy sequence for* α.

Proof. Let C be a Cauchy sequence for α. We will prove

$$\left| \alpha - \widehat{C}(n) \right| \ < \ 2^{-n} \tag{4.7}$$

and

$$\widehat{C}(n) \ < \ \widehat{C}(n+1)\ . \tag{4.8}$$

and thus the lemma holds.

First we observe that $|\alpha - C(n+2)_\ell| < 2^{-(n+1)}$, and thus, we have

$$\left| \alpha - \widehat{C}(n) \right| \ = \ \left| \alpha - (\ C(n+2)_\ell - 2^{-(n+1)}\ ) \right| \ =$$
$$| \alpha - C(n+2)_\ell | + 2^{-(n+1)} \ < \ 2^{-(n+1)} + 2^{-(n+1)} \ = \ 2^{-n}\ .$$

The first equality holds by the definition of $\widehat{C}$, and the second equality holds since $C(n+2)_\ell$ lies below α. This proves that (4.7) holds.

Next we observe that

$$\left| C(2^n) - C(2^{n+1}) \right| \ < \ 2^{-n} + 2^{-(n+1)} \tag{4.9}$$

23

holds for all n. Now, assume for the sake of a contradiction that (4.8) does not hold, that is, assume there exists m such that $\widehat{C}(m) \geq \widehat{C}(m+1)$. Then we have

$$\widehat{C}(m) - \widehat{C}(m+1) \geq 0 \; .$$

By the definition of $\widehat{C}$, we have

$$C(m+2)_\ell - 2^{-(m+1)} - (\; C(m+3)_\ell - 2^{-(m+2)} \;) \geq 0 \; .$$

Hence

$$C(m+2)_\ell - C(m+3)_\ell \geq 2^{-(m+2)} \; .$$

By the definition of $C(\cdot)_\ell$, we have

$$C(2^{m+2}) - 2^{-(m+2)} - (\; C(2^{m+3}) - 2^{-(m+3)} \;) \geq 2^{-(m+2)} \; .$$

Hence, we have

$$C(2^{m+2}) - C(2^{m+3}) \geq 2^{-(m+2)} + 2^{-(m+2)} - 2^{-(m+3)} = 2^{-(m+2)} + 2^{-(m+3)} \; .$$

This contradicts (4.9), and we have proved that (4.8) holds. $\qquad\square$

Lemma 4.11. *There is a parameterized function-oracle Turing machine M with the following properties. Let $C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ be a Cauchy sequence for an irrational $\alpha \in (0, 1)$. Then*

- $\Phi_M^C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ *is a strictly increasing Cauchy sequence for α*

- M^C *on input n runs in time* $\text{poly}(\max\{\|C(2^{n+2})\|, n\})$ *and uses a single oracle call of input size $n + 3$.*

Proof. On input n, M constructs the number 2^{n+2} (representable in $n + 3$ bits) and performs the oracle call $C(2^{n+2})$, and then calculates and outputs the rational number $C(2^{n+2}) - 2^{-(n+2)} - 2^{-(n+1)} = \widehat{C}(n)$. This calculation involves basic arithmetic on numbers representable in $\max\{\|C(2^{n+2})\|, O(n)\}$ bits, hence in total time $\text{poly}(\max\{\|C(2^{n+2})\|, n\})$. By Lemma 4.10, $\widehat{C}$ is a strictly increasing Cauchy sequence for α. $\qquad\square$

4.7 From converging base-b sequences to Cauchy sequences

Let A be a converging base-b sequence for α, and let $C(n) = A(n)b^{-n}$. Now, by our definitions, C is a Cauchy sequence for α. Thus the proof of the next lemma is straightforward.

Lemma 4.12. *Let $b \geq 2$. There is a parameterized function-oracle Turing machine M with the following properties. Let $A : \mathbb{N}^+ \longrightarrow \mathbb{Z}$ be a converging base-b sequence for an irrational $\alpha \in (0, 1)$. Then*

- $\Phi_M^A : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ *is a Cauchy sequence for* α

- M^A *on input* n *runs in time* $\mathrm{poly}(\max\{\log A(n), n\})$ *and uses a single oracle call of input size* $O(\|n\|)$.

Proof. On input $n \in \mathbb{N}^+$, M queries the oracle for $A(n)$ and then computes the rational number $A(n)b^{-n}$. All the numbers involved occupy $O(\max\{\log A(n), n\})$ bits. Hence, M runs in time at most $\mathrm{poly}(\max\{\log A(n), n\})$. $\square$

4.8   From Cauchy sequences to converging base-b sequences

Let C be a Cauchy sequence for α. We show how to compute $A(n)$ where A is a converging base-b sequence for α.

First we define the sequences $X_0, X_1, X_2, \ldots$ and $Y_0, Y_1, Y_2, \ldots$. Let $p \in \mathbb{Z}$ and $q \in \mathbb{N}^+$ be arbitrary. Let $X_0 = p \operatorname{div} q$, let $Y_0 = p \bmod q$, and let

$$X_{n+1} = X_n \times b + ((Y_n \times b) \operatorname{div} q) \quad \text{and} \quad Y_{n+1} = (Y_n \times b) \bmod q$$

where the operators

- $x \operatorname{div} y$ (integer division)

- $x \bmod y$ (the remainder of integer division, in $\{0, \ldots, y-1\}$)

have the property

$$(x \operatorname{div} y) \times y + (x \bmod y) = x \ . \tag{4.10}$$

Lemma 4.13. *For any* n, *we have (i)* $Y_n < q$, *(ii)* $pq^{-1} = X_n b^{-n} + Y_n b^{-n} q^{-1}$ *and (iii)* $X_n b^{-n} \leq pq^{-1} < (X_n + 1)b^{-n}$.

Proof. It is obvious that (i) holds, and (iii) follows straightforwardly from (i) and (ii). We prove (ii) by induction on n. It is obvious that (ii) holds if $n = 0$. Furthermore, we have

$$\begin{aligned}
X_{n+1}b^{-(n+1)} &+ Y_{n+1}b^{-(n+1)}q^{-1} = \\
&(X_n b + ((Y_n b) \operatorname{div} q))b^{-(n+1)} + ((Y_n b) \bmod q)b^{-(n+1)}q^{-1} = \\
&[X_n bq + ((Y_n b) \operatorname{div} q)q + ((Y_n b) \bmod q)]b^{-(n+1)}q^{-1} = \\
&[X_n bq + Y_n b]b^{-(n+1)}q^{-1} = X_n b^{-n} + Y_n b^{-n}q^{-1} = pq^{-1}
\end{aligned}$$

where the first equality holds by the definition of X_{n+1} and Y_{n+1}, the third equality holds by (4.10) and the last equality holds by the induction hypothesis. $\square$

We may now compute $A(n)$ by the following procedure:

- let $p/q = C(2n)$ where $p \in \mathbb{Z}$ and $q \in \mathbb{N}^+$

25

- compute X_n

- output $A(n) := X_n$.

By Lemma 4.13 (iii), we have

$$A(n)b^{-n} \leq C(2n) < (A(n)+1)b^{-n}$$

and thus

$$\left| A(n)b^{-n} - C(2n) \right| < b^{-n} \leq (2n)^{-1} .$$

Moreover, as C is a Cauchy sequence for α, we have $|\alpha - C(2n)| < (2n)^{-1}$, and thus, we also have $|A(n)b^{-n} - \alpha| < n^{-1}$. This proves that A is a converging base-b sequence for α.

Lemma 4.14. *Let $b \geq 2$. There is a parameterized function-oracle Turing Machine M with the following properties. Let $C : \mathbb{N}^+ \longrightarrow \mathbb{Q}$ be a Cauchy sequence for an irrational $\alpha \in (0,1)$. Then*

- $\Phi_M^C : \mathbb{N}^+ \longrightarrow \mathbb{Z}$ *is a converging base-b sequence for α*

- M^C *on input n runs in time $\mathrm{poly}(n + \|C(2n)\|)$ and uses a single oracle call of input size $O(\log n)$.*

Proof. The algorithm given above uses $C(2n)$, and hence an oracle call of input size $O(\log n)$. Moreover the algorithm uses n iterations involving 5 arithmetical operations in each iteration; it is a straightforward induction to see that each of these operations is applied to integer arguments of size at most

$$\|C(2n)\| + n(\|b\| + 1) \ = \ O(n + \|C(2n)\|)$$

bits. As each arithmetical operation is computable in polynomial time in the size of the representation, the total time used is

$$O(n \, \mathrm{poly}(n + \|C(2n)\|)) \ = \ \mathrm{poly}(n + \|C(2n)\|) .$$

$\square$

4.9   Summary

Recall that $O(t)_R$ denotes the class of all irrational α in the interval $(0,1)$ such that at least one R-representation of α is computable by a Turing machine running in time $O(t(n))$ (n is the length of the input, see Definition 1.4 on page 7). When we combine the results on the complexity of conversions among representations in this section, we get the following theorem.

Theorem 4.15. *Consider the representations by (1) Cauchy sequences, (2) increasing Cauchy sequences, (3) fuzzy cuts, (4) signed digit expansions and (5) converging base-b sequences, and let R_1 and R_2 be any two of these five representations. For an arbitrary time-bound t, we have*

$$O(t)_{R_2} \ \subseteq \ O(\mathrm{poly}(2^{2^n} \cdot t(2^n + 3)))_{R_1} .$$
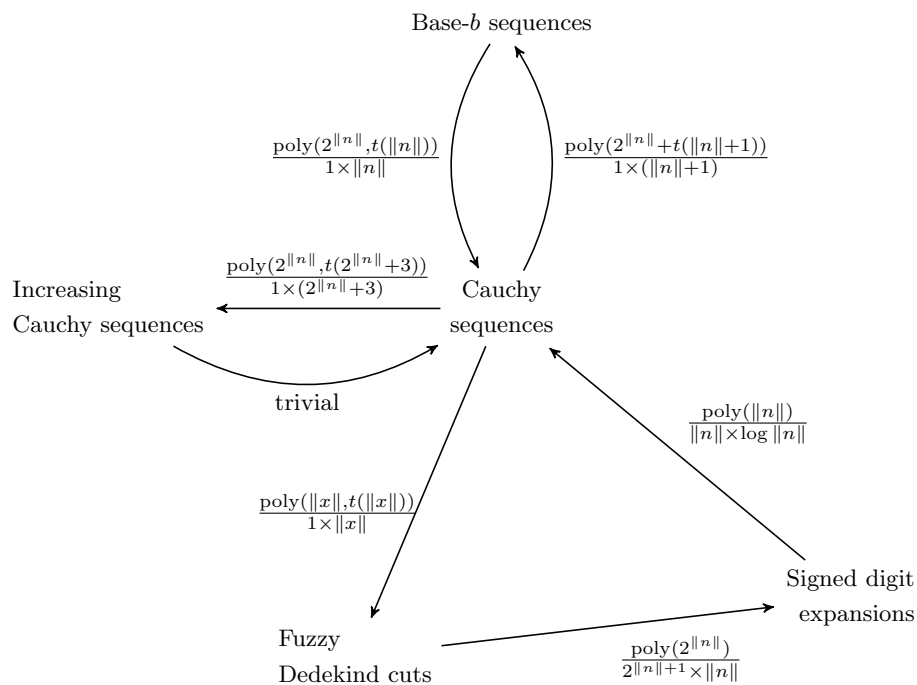
26

Figure 2: Reductions among representations in the cluster of Cauchy sequences. Next to each arrow representing a reduction is the cost of the reduction (above the line) and the number and size of oracle calls (below).

Proof. The result is obtained by combining the complexities of the reductions shown in this section; see Figure 2. The complexity figures in this diagram are drawn from the corresponding lemmata, where in case the formula in the lemma uses the size of the result of the oracle call, we substitute it with $t(x)$ where x is the size of the oracle argument. We have also rounded the figures a little (e.g., $2n$ instead of $2n - 2$). For example, to compute an increasing Cauchy sequence from an arbitrary Cauchy sequence, according to Lemma 4.11, we need time $\operatorname{poly}(\max\{\|C(2^{n+2})\|, n\})$. We bound $\|C(2^{n+2})\|$ by $t(\|2^{n+2}\|) = t(n+3) \leq t(2^{\|n\|} + 3)$. We bound n by $2^{\|n\|}$. The oracle call has argument 2^{n+2} whose bit-size is bounded by $t(2^{\|n\|} + 3)$.

Note that to get from certain representations to some others, it is necessary to compose reductions; the costliest path leads from base-b sequences to increasing Cauchy sequences via ordinary Cauchy sequences. The calculation proceeds as follows. Assume that the Cauchy sequence is computed in time t_{CS}. Then the computation of the increasing Cauchy sequence takes $\operatorname{poly}(2^n, t_{CS}(2^n + 3))$ (here n denotes the input size!). We also know that $t_{CS}(x) = \operatorname{poly}(2^x, t(x))$ where t is the time to compute the base-b sequence and x is the size of the input. In fact, we use this computation as an oracle and the size of the oracle call is $2^n + 3$; so it runs in $\operatorname{poly}(2^{2^n+3}, t(2^n + 3))$. This stage of the computation dominates, so we conclude that the time to compute the composed reduction is given by the latter formula.

A similar calculation for other paths gives smaller bounds, hence the statement of the theorem. $\qquad\square$

5 Base-b expansions and sum approximations

5.1 The base-b expansions

The representation of reals by base-b expansions, or perhaps we should say base-10 expansions, is very well known. We are talking about the standard daily-life representation of reals. We will restrict our attention to irrational numbers between 0 and 1.

Definition 5.1. A *base* is a natural number strictly greater than 1, and a *base-b digit* is a natural number in the set $\{0, 1, \ldots, b - 1\}$.

Let b be a base, and let $\mathtt{D}_1, \ldots, \mathtt{D}_n$ be base-b digits. We will use $(0.\mathtt{D}_1\mathtt{D}_2\ldots\mathtt{D}_n)_b$ to denote the rational number $0 + \sum_{i=1}^n \mathtt{D}_i b^{-i}$.

Let $\mathtt{D}_1, \mathtt{D}_2, \ldots$ be an infinite sequence of base-b digits. We say that $(0.\mathtt{D}_1\mathtt{D}_2\ldots)_b$ is *the base-b expansion of the irrational number* α if we have

$$(0.\mathtt{D}_1\mathtt{D}_2\ldots\mathtt{D}_n)_b \ \leq \ \alpha \ < \ (0.\mathtt{D}_1\mathtt{D}_2\ldots\mathtt{D}_n)_b + b^{-n}$$

for all $n \geq 1$. We identify the base-b expansion with the function $E_b^\alpha : \mathbb{N}^+ \longrightarrow$

28

$\{0, \ldots, b-1\}$ that yields the ith digit of the base-b expansion of α, that is, $E_b^\alpha(i) = D_i$ when $(0.D_1D_2\ldots)_b$ is the base-b expansion of α. $\qquad\square$

It is easy to see that a base-b expansion can be subrecursively converted into a Cauchy sequence: If $(0.D_1D_2\ldots)_b$ is the base-b expansion of α, then

$$(0.D_1)_b \ , \ (0.D_1D_2)_b \ , \ (0.D_1D_2D_3)_b \ , \ \ldots$$

will be the first elements of a Cauchy sequence for α, and thus, we do not need unbounded search to compute a Cauchy sequence if we have access to the base-b expansion.

Lemma 5.2. *Let $b > 1$. There is a parameterized function-oracle Turing machine M with the following properties. Let $E_b^\alpha : \mathbb{N} \longrightarrow \{0, \ldots, b-1\}$ be the base-b expansion of an irrational $\alpha \in (0,1)$. Then*

- $\Phi_M^{E_b^\alpha} : \mathbb{N} \longrightarrow \mathbb{Q}$ *is a Cauchy sequence for α*

- $M^{E_b^\alpha}$ *on input n runs in time* $\mathrm{poly}(n)$ *and uses n oracle calls of input size at most $\log n$.*

Proof. On input n, M performs n oracle queries to E_b^α, of size at most $\log n$ to obtain the first n digits $D_1, \ldots, D_n$ of the base-b expansion of α. Each digit requires $\log b$ space, and computing the rational number $p/q = \sum_{i=1}^n D_i b^{-i}$ can thus be done in time $\mathrm{poly}(n)$. $\qquad\square$

It turns out that we cannot subrecursively convert a Cauchy sequence into a base-b expansion. Neither can we, in general, subrecursively convert a base-b expansion into a base-a expansion. Let us recall a definition from Kristiansen [32].

Definition 5.3. We will use $\mathrm{prim}(b)$ to denote the set of prime factors of the base b, that is, $\mathrm{prim}(b) = \{p \mid p \text{ is a prime and } p|b\}$.

Let a and b be bases such that $\mathrm{prim}(a) \subseteq \mathrm{prim}(b)$. We will now define the *base transition factor* from a to b. Let $b = p_1^{k_1} p_2^{k_2} \ldots p_n^{k_n}$, where p_i is a prime and $k_i \in \mathbb{N}^+$ (for $i = 1, \ldots, n$), be the prime factorization of b. Then, a can be written in the form $a = p_1^{j_1} p_2^{j_2} \ldots p_n^{j_n}$ where $j_i \in \mathbb{N}$ (for $i = 1, \ldots, n$). The *base transition factor* from a to b is the natural number k such that

$$k \ = \ \max\{ \lceil j_i/k_i \rceil \mid 1 \leq i \leq n \} \,.$$

$\qquad\square$

Note that the base transition factor from a to b is defined if and only if $\mathrm{prim}(a) \subseteq \mathrm{prim}(b)$ (the definition does not make sense when $\mathrm{prim}(a) \not\subseteq \mathrm{prim}(b)$). When we assume that the base transition factor from a to b exists, it is understood that we have $\mathrm{prim}(a) \subseteq \mathrm{prim}(b)$.

Theorem 5.4 (The Base Transition Theorem). *Let k be the base transition factor from a to b, and let $(0.D_1 D_2 \ldots)_a$ and $(0.\dot{D}_1 \dot{D}_2 \ldots)_b$ be, respectively, the base-a and base-b expansion of the real number α. Then, for all $n \in \mathbb{N}$, we have*

$$
\begin{aligned}
(0.D_1 \ldots D_n)_a &\leq (0.\dot{D}_1 \ldots \dot{D}_{kn})_b \leq \alpha \\
&< (0.\dot{D}_1 \ldots \dot{D}_{kn})_b + b^{-kn} \leq (0.D_1 \ldots D_n)_a + a^{-n} . \quad \text{(I)}
\end{aligned}
$$

Moreover, for all $n, \ell \in \mathbb{N}$, we have

$$
(0.\dot{D}_1 \ldots \dot{D}_{kn})_b < (0.\dot{D}_1 \ldots \dot{D}_\ell)_b \implies (0.D_1 \ldots D_n)_a < (0.D_1 \ldots D_{\ell m})_a \quad \text{(II)}
$$

where $m = \lceil \log_a b \rceil$.

A proof of the Base Transition Theorem can be found in [32]. Assume that the base transition factor k from base a to base b exists. Then, by clause (I) of the theorem, the first n fractional digits $0.D_1 \ldots D_n$ of the base-a expansion of α will be determined by the first kn fractional digits $0.\dot{D}_1 \ldots \dot{D}_{kn}$ of the base-b expansion of α, and thus, we can subrecursively convert a base-b expansion into a base-a expansion.

Lemma 5.5. *Assume that the base transition factor k from base a to base b exists. There is a parameterized function-oracle Turing machine M with the following properties. Let $E_b^\alpha : \mathbb{N}^+ \longrightarrow \{0, \ldots, b-1\}$ be the base-b expansion of an irrational $\alpha \in (0, 1)$. Then*

- $\Phi_M^{E_b^\alpha} : \mathbb{N}^+ \longrightarrow \{0, \ldots, a-1\}$ *is the base-a expansion of α*

- $M^{E_b^\alpha}$ *on input n runs in time $\mathrm{poly}(n)$ and uses kn oracle calls, each of input size at most $O(\log n)$.*

Proof. Note that the first kn digits of the b-ary expansion of α can be computed using kn oracle calls each of size at most $\log kn = O(\log n)$. Converting these kn digits to a number of the form $q \cdot b^{-kn}$ can be done in time $\mathrm{poly}(kn) = \mathrm{poly}(n)$, and hence computing $p = \lfloor a^n \cdot q / b^{kn} \rceil$ can be done in time $\mathrm{poly}(kn) = \mathrm{poly}(n)$. By Clause (I) of the Base Transition Theorem, $(0.D_1 \ldots D_n)_a = p/a^n$, and as each base-$a$ digit $D_1, \ldots, D_n$ is in $\{0, \ldots, a-1\}$ and $p/a^n = \sum_{i=1}^n D_i a^{-i}$, a simple greedy algorithm may compute the digits $D_1, \ldots, D_n$ of $(0.D_1 \ldots D_n)_a$ in increasing order in time $\mathrm{poly}(n)$ when p has been computed. $\square$

If the base transition factor from a to b does not exist, then we cannot compute E_a^α subrecursively in E_b^α even if we assume that α is irrational. This is proved formally in [32], but intuitively it is not very hard to see why this is the case: Consider an irrational α which lies very close to the rational number $(0.1)_{10}$. We have $(0.1)_{10} = (0.0(0011)^*)_2$, and let us say that the base-2 expansion of α starts with

$$
\alpha = 0.00011001100110011001100110011001100110011001100110011\ldots
$$

Given the digits of the base-2 expansion displayed above, we cannot tell if the first fractional digit of the base-10 expansion should be 0 or 1. We need more digits of the base-2 expansion to the determine the first digit of the base-10 expansion. Now, α cannot equal $(0.1)_{10}$ since we have assumed that α is irrational. Thus, sooner or later we will find a digit in the base-2 expansion which allows to determine the first digit of the base-10 expansion, but we need unbounded search to find that digit.

In general, if the base transition factor from a to b does not exist, that is, if $\mathrm{prim}(a) \not\subseteq \mathrm{prim}(b)$, we cannot subrecursively compute E_a^α in E_b^α. It follows that we cannot compute E_b^α subrecursively in a Cauchy sequence for α (for any base b). Assume for the sake of a contradiction that we can, that is, assume that we can compute E_b^α subrecursively in an arbitrary Cauchy sequence for α. Pick a base b_0 such that $\mathrm{prim}(b) \not\subseteq \mathrm{prim}(b_0)$. By Lemma 5.2, we can subrecursively compute a Cauchy sequence C for α in $E_{b_0}^\alpha$. By our assumption we can subrecursively compute E_b^α in C. Hence, we can subrecursively compute E_b^α in $E_{b_0}^\alpha$ which is impossible as the base transition factor from b to b_0 does not exist.

5.2   Base-b sum approximations

Base-b sum approximations (from below and above) were introduced by Kristiansen in [31] and studied further in [32] and, with Georgiev and Stephan, in [20].

Definition 5.6. Let $(0.\mathtt{D}_1\mathtt{D}_2\ldots)_b$ be the base-b expansion of the irrational $\alpha \in (0,1)$ (thus, we have $E_b^\alpha(n) = \mathtt{D}_n$).

The *base-b sum approximation from below* of α is the function $\hat{A}_b^\alpha : \mathbb{N} \longrightarrow \mathbb{Q}$ defined by $\hat{A}_b^\alpha(0) = 0$ and $\hat{A}_b^\alpha(n+1) = E_b^\alpha(m)b^{-m}$ where m is the least m such that

$$\sum_{i=0}^{n} \hat{A}_b^\alpha(i) \quad < \quad (0.\mathtt{D}_1\ldots\mathtt{D}_m)_b$$

that is, $\hat{A}_b^\alpha(n)$ is the value represented by the nth non-zero digit of the base-b expansion of α.

Let $\overline{\mathtt{D}}$ denote the *complement digit* of the base-b digit $\mathtt{D}$, that is, let $\overline{\mathtt{D}} = (b-1) - \mathtt{D}$ (observe that we have $(0.\mathtt{D}_1\mathtt{D}_2\mathtt{D}_3\ldots)_b + (0.\overline{\mathtt{D}}_1\overline{\mathtt{D}}_2\overline{\mathtt{D}}_3\ldots)_b = 1$ for any base b and any base-b expansion $(0.\mathtt{D}_1\mathtt{D}_2\ldots)_b$).

The *base-b sum approximation from above* of α is the function $\check{A}_b^\alpha : \mathbb{N} \longrightarrow \mathbb{Q}$ defined by $\check{A}_b^\alpha(0) = 0$ and $\check{A}_b^\alpha(n+1) = \overline{E_b^\alpha(m)}b^{-m}$ where m is the least m such that

$$1 - \sum_{i=0}^{n} \check{A}_b^\alpha(n) \quad > \quad 1 - (0.\overline{\mathtt{D}}_1\ldots\overline{\mathtt{D}}_m)_b .$$

$\square$

31

The functions $\hat{A}_b^\alpha$ and $\check{A}_b^\alpha$ are not defined if α is rational. When we use the notation it is understood that α is irrational. It is fairly straightforward to prove that

$$\sum_{i=0}^\infty E_b^\alpha(i)b^{-i} \;\;=\;\; \sum_{i=0}^\infty \hat{A}_b^\alpha(i) \;\;=\;\; 1 \;-\; \sum_{i=0}^\infty \check{A}_b^\alpha(i)\,.$$

A detailed proof can be found in [32].

We cannot subrecursively compute $\hat{A}_b^\alpha$ in $\check{A}_b^\alpha$, and neither can we subrecursively compute $\check{A}_b^\alpha$ in $\hat{A}_b^\alpha$ (for any base b). The following growth argument (see page 8) explains why we cannot subrecursively compute $\hat{A}_2^\alpha$ in $\check{A}_2^\alpha$: Let t be any time bound, and let $f : \mathbb{N} \longrightarrow \mathbb{N}$ be a strictly increasing function that grows faster than any function computable in time $O(t)$, moreover, let the graph of f be computable in polynomial time, that is, the relation $f(x) = y$ can be decided in time polynomial in the size of the natural numbers x and y. It is straightforward to see that such an f exists, and that we may further assume that $2^x \le f(x)$. Consider the irrational number β given by $\hat{A}_2^\beta(n) = 2^{-f(n)}$. We have

$$\beta = (0.0\ldots 010\ldots 010\ldots 010\ldots 010\ldots)_2$$

where the sequences $0\ldots 0$ of zeros are getting longer and longer. Now

$$\beta = (0.0\ldots 010\ldots 010\ldots 010\ldots)_2 = 1 - (0.1\ldots 101\ldots 101\ldots 101\ldots)_2$$

and thus $\check{A}_2^\beta(n) = 2^{-g(n)}$ where $g : \mathbb{N} \longrightarrow \mathbb{N}$ is a slow growing function (we have $g(x) \le 2x$), indeed, $g(x)$ is computable in polynomial time as the graph of f is computable in polynomial time. Since we can compute g in polynomial time, we can also compute $\check{A}_2^\beta$ in polynomial time. Obviously, we cannot compute $\hat{A}_2^\beta$ in time $O(t)$, if we could, then we could also compute f in time $O(t)$, contrary to our assumption.

Hence we conclude that for any time-bound t there exists an irrational β such that $\check{A}_2^\beta$ is computable in polynomial time whereas $\hat{A}_2^\beta$ is not computable in time $O(t)$. This shows that we cannot subrecursively compute $\hat{A}_2^\alpha$ in $\check{A}_2^\alpha$. The argument generalizes easily to work for any base b, and hence, we cannot subrecursively compute $\hat{A}_b^\alpha$ in $\check{A}_b^\alpha$. A symmetric argument will show that we cannot subrecursively compute $\check{A}_b^\alpha$ in $\hat{A}_b^\alpha$. Detailed proofs can be found in [32].

It is also proved in [32] that the base transition factor from a to b exists, if and only if, we can subrecursively compute $\hat{A}_a^\alpha$ in $\hat{A}_b^\alpha$, if and only if, we can subrecursively compute $\check{A}_a^\alpha$ in $\check{A}_b^\alpha$. We have already argued why we cannot subrecursively compute E_a^α in E_b^α when the base transition factor from a to b does not exist (see page 30). The very same argument should also give an intuitive explanation of why we cannot subrecursively compute $\hat{A}_a^\alpha$ in $\hat{A}_b^\alpha$, or $\check{A}_a^\alpha$ in $\check{A}_b^\alpha$, when the base transition factor from a to b does not exist. In the next subsection we will analyze the complexity of computing $\hat{A}_a^\alpha$ in $\hat{A}_b^\alpha$ when the needed base transition factor is available.

32

5.3 From base-b sum approximations to base-a sum approximations

Assume that the base transition factor k from base a to base b exists. We will now give and explain an algorithm for computing $\hat{A}_a^\alpha$ (α's base-a sum approximation from below) using $\hat{A}_b^\alpha$ (α's base-b sum approximation from below) as an oracle. Of course, α's base-a sum approximation from above can be computed from α's base-b sum approximation from above by a symmetric algorithm.

Assume the values of $\hat{A}_a^\alpha(0), \ldots, \hat{A}_a^\alpha(n)$ are already computed. Then, the algorithm computes the value of $\hat{A}_a^\alpha(n+1)$ by carrying out the following instructions:

- Step 1: Compute the rational number $\sum_{i=0}^n \hat{A}_a^\alpha(i)$. The number will be of the form $(0.\mathtt{D}_1 \ldots \mathtt{D}_p)_a$ for some p. Compute that p.

- Step 2: Ask the oracle $\hat{A}_b^\alpha$ for the value of $\hat{A}_b^\alpha(kp+1)$ (where k is the base transition factor). The oracle will yield a rational number of the form $\mathtt{D}b^{-\ell}$ where $\ell \geq kp+1$ and $\mathtt{D}$ is a nonzero base b digit. Compute ℓ.

- Step 3: Use the oracle $\hat{A}_b^\alpha$ to compute the rational number

$$R = \sum_{i=0}^{k\ell m} \hat{A}_b^\alpha(i)$$

  where k is the base transition factor, $m = \lceil \log_a b \rceil$ and ℓ is the value computed in Step 2.

- Step 4: Compute the least $i \geq 1$ such that digit number $p+i$ in the base-a expansion of R is nonzero (where p is the value computed in Step 1 and R is the value computed in Step 3). Give the output $\mathtt{D}a^{-(p+i)}$ where $\mathtt{D}$ is digit number $p+i$ in the base-a expansion of R, that is, we have $\hat{A}_a^\alpha(n+1) = \mathtt{D}a^{-(p+i)}$

We will now argue that the algorithm gives correct output. Let $(0.\mathtt{D}_1\mathtt{D}_2\ldots)_a$ and $(0.\dot{\mathtt{D}}_1\dot{\mathtt{D}}_2\ldots)_b$ be, respectively, the base-a and base-b expansion of α. In Step 1, the algorithm computes $(0.\mathtt{D}_1 \ldots \mathtt{D}_p)_a$. According to the definition of $\hat{A}_a^\alpha$, the output should be $\mathtt{D}_{p+i}a^{-(p+i)}$ where i is the least number such that

$$(0.\mathtt{D}_1 \ldots \mathtt{D}_p)_a < (0.\mathtt{D}_1 \ldots \mathtt{D}_{p+i})_a \ .$$

In Step 2, the algorithm computes ℓ. By the definition of sum approximations,

$$(0.\dot{\mathtt{D}}_1 \ldots \dot{\mathtt{D}}_{kp})_b < (0.\dot{\mathtt{D}}_1 \ldots \dot{\mathtt{D}}_\ell)_b \ . \tag{5.1}$$

Hence, by clause (II) of Base Transition Theorem, we have

$$(0.\mathtt{D}_1 \ldots \mathtt{D}_p)_a < (0.\mathtt{D}_1 \ldots \mathtt{D}_{\ell m})_a \tag{5.2}$$

33

where $m = \lceil \log_a b \rceil$. This shows that the denominator of the next term in the base-a sum approximation is at most $a^{\ell m}$. In Step 3, the algorithm computes R. By (5.2) and clause (I) of the Base Transition Theorem, we have

$$(0.\mathtt{D}_1 \ldots \mathtt{D}_p)_a < (0.\mathtt{D}_1 \ldots \mathtt{D}_{\ell m})_a \leq (0.\dot{\mathtt{D}}_1 \ldots \dot{\mathtt{D}}_{k\ell m})_b \leq R < \alpha$$
$$< (0.\dot{\mathtt{D}}_1 \ldots \dot{\mathtt{D}}_{k\ell m})_b + b^{-k\ell m} \leq (0.\mathtt{D}_1 \ldots \mathtt{D}_{\ell m})_a + a^{-\ell m} . \quad (5.3)$$

Now, (5.3) implies that the first ℓm fractional digits of the base-a expansion of α, that is $\mathtt{D}_1 \ldots \mathtt{D}_{\ell m}$, coincide with the first ℓm fractional digits of the base-a expansion of the rational number R. Thus, the algorithm computes a correct result in Step 4.

Note that in Definition 1.4, the time bound s may depend on the time bound t. Therefore, when we convert a representation to another the time bound of the conversion is allowed to depend on the time it takes to compute the source representation. We use such a bound in the following lemma. In fact, what we need is a bound on the *size* of the oracle result; we use the rule that the execution time of a Turing machine bounds its output size. Such a bound is necessary here (in contrast to most previous conversion results) because the algorithm does a number of steps that depends on the size of the oracle answer (see Step 3 above).

Lemma 5.7. *Assume that the base transition factor k from base a to base b exists. There is a parameterized function-oracle Turing machine M with the following properties. Let $\hat{A}_b^\alpha : \mathbb{N} \longrightarrow \mathbb{Q}$ be the base-b sum approximation from below of an irrational $\alpha \in (0,1)$. Furthermore, assume that $\hat{A}_b^\alpha$ is computable within a time bound s, and let $g(n) = (\lambda x.\|m\| + s(\|k\| + x + 1))^{(n)}(0)$ where $m = \lceil \log_a b \rceil$. Then*

- $\Phi_M^{\hat{A}_b^\alpha} : \mathbb{N} \longrightarrow \mathbb{Q}$ *is the base-a sum approximation from below of α*

- $M^{\hat{A}_b^\alpha}$ *on input n runs in time $\mathrm{poly}(2^{g(n)} + s(\|k\| + g(n)))$ and uses at most $k2^{g(n)}$ oracle calls, each of input size at most $\|k\| + g(n)$.*

Proof. By induction on n, we prove the time bound and also a bound on the exponent of a in the denominator of $\hat{A}_a^\alpha(n)$.

To compute $\hat{A}_a^\alpha(1)$, we call the oracle with input 1 and obtain a position ℓ; we then compute R to obtain from it the first non-zero digit in base a. By assumption, $\hat{A}_b^\alpha(1)$ is computable in time $s(\|1\|)$; this implies that $s(\|1\|)$ is also a bound on the bit-length of the result of this computation, hence on that of ℓ (the result of the computation is the binary representation of $cb^{-\ell}$ for some c). The machine has to access values of $\hat{A}_b^\alpha(i)$ for $i = 1, \ldots, k\ell m$, so it uses $k\ell m \leq km2^{s(\|1\|)} \leq k2^{g(1)}$ oracle calls. Since the size of a result is bounded by its computation time, we have that the size of the largest oracle answer is bounded by $s(\|k\| + g(1))$. It is routine to verify that the execution time is dominated by the expression $k\ell m$, which is the length of the summation in

34

Step 3, times the size of the largest number returned from an oracle call. We have bounded $k\ell m$ by $k2^{g(1)}$, and the size of the last oracle call is bounded by $s(\|k\ell m\|) \leq s(\|k\| + g(1))$, justifying the bound on execution time for $n = 1$. We note that the exponent in the denominator of $\hat{A}_a^\alpha(1)$ is bounded by ℓm (5.2), hence by $2^{g(1)}$.

We turn to the induction step. Assume $n > 1$, and assume inductively that the position p is bounded by $2^{g(n-1)}$. The value ℓ will be bounded by $2^{s(\|kp+1\|)}$ and the largest input to an oracle call will be

$$k\ell m \ \leq \ km2^{s(\|kp+1\|)} \ \leq \ km2^{s(\|k\|+\|p\|+1)} \ \leq \ km2^{s(\|k\|+g(n-1)+1)} \ \leq \ k2^{g(n)}\ .$$

This also bounds the number of oracle calls, since we can store results of previous queries and therefore never query the oracle on the same input twice. A bound on the bit-length of the result of the oracle call is $s(\|k\| + g(n))$. The execution time is polynomial in the sum of this quantity and the number of calls, yielding the bound $\mathrm{poly}(2^{g(n)} + s(\|k\| + g(n)))$, and the position of the last base-a digit is bounded by $\ell m \leq 2^{g(n)}$. $\qquad\square$

We note that the lemma implies that if $\hat{A}_b^\alpha$ is computable by an $O(s)$-time Turing machine, then $\hat{A}_a^\alpha$ is computable by an $O(s')$-time Turing machine where the time-bound s' is primitive recursive in the time bound s.

5.4 From base-b sum approximations to base-a expansions

This is very similar to converting base-b expansions to base-a expansions. As in Section 5.2, we argue that if the base transition factor from a to b does not exist, then conversion from base-b sum approximations to base-a expansions cannot be done subrecursively. In the case that the base transition factor exists, we have a straightforward conversion algorithm.

Lemma 5.8. *Assume that the base transition factor k from base a to base b exists. There is a parameterized function-oracle Turing machine M with the following properties. Let $\hat{A}_b^\alpha : \mathbb{N} \longrightarrow \mathbb{Q}$ be the base-b sum approximation from below of an irrational $\alpha \in (0, 1)$. Then*

- $\Phi_M^{\hat{A}_b^\alpha} : \mathbb{N} \longrightarrow \{0, \ldots, a-1\}$ *is the base-a expansion of α*

- $M^{\hat{A}_b^\alpha}$ *on input n runs in time $\mathrm{poly}(\|\hat{A}_b^\alpha(kn)\|)$ and uses at most kn oracle calls, each of input size $O(\log n)$.*

Proof. As in the proof of Lemma 5.5, computing the nth digit of $(0.\mathtt{D}_1\mathtt{D}_2\mathtt{D}_3\cdots)_a$ is possible using the first kn digits of $(0.\mathtt{D}_1\mathtt{D}_2\mathtt{D}_3\cdots)_b$. By definition of $\hat{A}_b^\alpha$, if $\hat{A}_b^\alpha(n) = cb^{-m}$ for some $m \in \mathbb{N}$ and $c \in \{1, \ldots, b-1\}$, then the nth non-zero digit of $(0.\mathtt{D}_1\mathtt{D}_2\mathtt{D}_3\cdots)_b$ is c. Hence, to find the first kn digits of the base-b expansion, M may simply query $\hat{A}_b^\alpha(1), \hat{A}_b^\alpha(2), \ldots$ in order until the first i is

35

found for which $\hat{A}_b^\alpha(i) = c_i b^{-m_i}$ with $m_i \geq kn$. This is obviously a bounded search as such an m_i will exist for some $i \leq kn$. Once we have $(0.\mathtt{D}_1\mathtt{D}_2\ldots\mathtt{D}_{kn})_b$, we extract $(0.\mathtt{D}_1\mathtt{D}_2\mathtt{D}_3\ldots\mathtt{D}_n)_a$ by repeated multiplication by a and extraction of the integer part. Clearly, this procedure uses at most kn oracle calls, each of size at most $\log kn$. The time complexity of the procedure is dominated by a polynomial in the size of the result of the last oracle call, i.e., $\mathrm{poly}(\|\hat{A}_b^\alpha(kn)\|)$ (note that the denominator of this number is at least b^{kn}). $\qquad\square$

Lemma 5.8 does of course also hold for sum approximations from above.

5.5 Gray codes

The Gray code representation of real numbers was introduced by Tsuiki [39] and studied further in Berger et al. [8]. The representation is subrecursively equivalent to the representation by base-2 expansions.

Definition 5.9. The function $G : \mathbb{N} \longrightarrow \{0,1\}$ is the *Gray code* of the irrational number α if $G(i) = 0$ if there is an even number m with

$$m2^{-i} - 2^{-(i+1)} < \alpha < m2^{-i} + 2^{-(i+1)}$$

and $G(i) = 1$ if the same holds for an odd number m. $\qquad\square$

Gray codes are usually defined as maps $G : \mathbb{N} \longrightarrow \{0,1,\bot\}$ where $G(i) = \bot$ if α is of the form $m2^{-i} - 2^{-(i+1)}$ for some m and i, and hence is rational. As we are only considering irrationals, we do not need $\bot$.

Lemma 5.10 (Tsuiki [39]). *Let E_2^α and G^α, respectively, be the base-2 expansion and the Gray code of the irrational $\alpha \in (0,1)$. Then we have*

$$E_2^\alpha(1) = G^\alpha(0)$$
$$E_2^\alpha(n+1) = E_2^\alpha(n) \oplus G^\alpha(n)$$

where $\oplus$ denotes the XOR function.

Lemma 5.11. *There is a parameterized function-oracle Turing machine M with the following properties. Let $G : \mathbb{N} \longrightarrow \{0,1\}$ be the Gray code of an irrational $\alpha \in (0,1)$. Then*

- $\Phi_M^G : \mathbb{N}^+ \longrightarrow \{0,1\}$ *is the base-2 expansion of α*

- M^G *on input n runs in time $O(n \log n)$ and uses n oracle calls, each of input size $O(\log n)$.*

Proof. Straightforward from the conversion algorithm expressed by Lemma 5.10. $\qquad\square$

Lemma 5.12. *There is a parameterized function-oracle Turing machine M with the following properties. Let $E : \mathbb{N}^+ \longrightarrow \{0,1\}$ be the base-2 expansion of an irrational $\alpha \in (0,1)$. Then*

- *$\Phi_M^E : \mathbb{N} \longrightarrow \{0,1\}$ is the Gray code of α*

- *M^E on input n runs in time $O(\log n)$ and uses 2 oracle calls, each of input size $O(\log n)$.*

Proof. A simple rewrite of the equations in Lemma 5.10 gives:

$$G^\alpha(0) = E_2^\alpha(1)$$
$$G^\alpha(n+1) = E_2^\alpha(n+1) \oplus E_2^\alpha(n+2)$$

The implied algorithm has the complexity stated in the lemma. □

5.6   Summary

In this section, our main results concerned the representations by base-b expansions (Gray codes being equivalent to base-2 expansions), and the representations by base-b sum approximations. These representations form clusters in which not all representations allow for subrecursive conversion from one to the other. Base-b expansions are convertible to base-a expansions when the base transition factor from a to b exists—and then the overhead of the conversion is exponential in the bit-length of the input. Base-b sum approximations from below (above) are convertible to base-a sum approximations from below (above) when the base transition factor from a to b exists—and then the overhead of the conversion involves iteration, which guaranteed that an $O(t)$-time computable base-b sum approximation becomes an $O(s)$-time computable base-a sum approximation with s primitive recursive in t.

6 Representations subrecursively equivalent to Dedekind cuts

6.1 Dedekind cuts

The representation of irrational numbers by Dedekind cuts is not subrecursively equivalent to the representation by base-b expansions, neither is it subrecursively equivalent to the representation by base-b sum approximations from below or above, for any base b. From now on, we will use D^α to denote the Dedkind cut of α, see Definition 1.1.

It is fairly easy to see that we can subrecursively compute base-b expansion E_b^α in D^α, see Section 6.3 below, but it is not possible to subrecursively compute D^α in E_b^α for any fixed base b. An intuitive explanation of why this is impossible is very similar to our explanation of why we cannot subrecursively compute E_{10}^α in E_2^α at page 30: Let α be an irrational whose base-2 expansion starts with

$$\alpha \;=\; 0.000110011001100110011001100110011001100110011\ldots\;.$$

The period 0011 may be repeated arbitrarily many times, and thus, we will need unbounded search to determine if α lies above or below 10^{-1}, that is, we need unbounded search to compute $D^\alpha(10^{-1})$. This simple example should also serve as an intuitive explanation of why we cannot subrecursively compute D^α in α's base-b sum approximation from above or below, that is, $\hat{A}_b^\alpha$ or $\check{A}_b^\alpha$. Neither is it possible to subrecursively compute $\hat{A}_b^\alpha$, or $\check{A}_b^\alpha$, in D^α, but an intuitive explanation of why this is the case is not all that straightforward, and we refer the interested reader to Section 7 and 8 of Kristiansen [32] for more on the relationship between Dedekind cuts and base-b sum approximations.

6.2 Definitions

The next definition gives some representations being subrecursively equivalent to the representation by Dedekind cuts.

Definition 6.1. Let $\alpha \in (0,1)$ be an irrational number.

1. The *Beatty sequence* of α is the function $B^\alpha : \mathbb{N}^+ \longrightarrow \mathbb{N}$ given by

$$\frac{B^\alpha(n)}{n} \;<\; \alpha \;<\; \frac{B^\alpha(n)+1}{n}\;.$$

2. The *general base expansion* of α is the function

$$E^\alpha : (\mathbb{N} \setminus \{0,1\}) \times \mathbb{N}^+ \longrightarrow \mathbb{N}$$

   where $E^\alpha(b,n) = E_b^\alpha(n)$ (recall that E_b^α is the base-b expansion of α, see Definition 5.1).

3. The *Hurwitz characteristic* of α is the function $H^\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$ such that $H^\alpha(0), H^\alpha(1), H^\alpha(2), \ldots$ is a path in the Farey pair tree[3] $\mathcal{T}_F$, and moreover, for all $n \in \mathbb{N}$, we have $\alpha \in \mathcal{T}_F(H^\alpha(n))$.

$\square$

Dedekind cuts [9, 15] and Hurwitz characteristics [23] were known in the 19th century or earlier. Use of the Hurwitz characteristic to represent numbers rather than a stepping stone for other material is a much younger invention, see Lehman [35]. Beatty sequences appear in a problem by Samuel Beatty published in American Mathematical Monthly in 1926 [17], but such sequences were used earlier by Bernard Bolzano [11]. The representation by general base expansions might not have been investigated before, but it is very natural.

This section is based on the conference paper Kristiansen & Simonsen [34].

6.3 Conversion between general base expansions and Dedekind cuts

We will compute $E^\alpha(b, n)$ by computing the digits $\mathtt{D}_1, \mathtt{D}_2, \mathtt{D}_3, \ldots$ of α's base-b expansion one by one. When we have determined the digits $\mathtt{D}_1, \ldots, \mathtt{D}_n$, we know that

$$(0.\mathtt{D}_1 \ldots \mathtt{D}_n)_b < \alpha < (0.\mathtt{D}_1 \ldots \mathtt{D}_n)_b + b^{-n}$$

and then we can split the interval

$$(\ (0.\mathtt{D}_1 \ldots \mathtt{D}_n)_b \, , \ (0.\mathtt{D}_1 \ldots \mathtt{D}_n)_b + b^{-n} \ )$$

into b subintervals, each of length b^{-n-1}, and use the Dedekind cut of α to determine the digit $\mathtt{D}_{n+1}$.

Lemma 6.2. *There is a parameterized function-oracle Turing machine M with the following properties. Let $D : \mathbb{Q} \longrightarrow \{0,1\}$ be the Dedekind cut of an irrational $\alpha \in (0,1)$. Then*

- $\Phi_M^D : (\mathbb{N} \setminus \{0,1\}) \times \mathbb{N}^+ \longrightarrow \mathbb{N}$ *is the general base expansion of α*

- M^D *on input (b, n) runs in time $O(\mathrm{poly}(n)\mathrm{polylog}(b))$ and uses at most $n \log b$ oracle calls, each of input size at most $O(n \log b)$.*

Proof. M constructs the sequence $E_b^\alpha(1), E_b^\alpha(2), \ldots, E_b^\alpha(n)$ inductively by maintaining an open interval $I_i = (v_i, w_i)$ with rational endpoints $v_i, w_i \in \mathbb{Q}$ for each $i \in \{0, \ldots, n-1\}$ such that (i) $\alpha \in I_i$, (ii) v_i is a multiple of b^{-i}, and (iii)

---

[3]Strictly speaking, the classic Hurwitz characteristic corresponds to a path through the full Stern-Brocot tree (not the Farey pair tree as we consider here), and hence the classic Hurwitz characteristic H' of $\alpha \in (0,1)$ is the function defined by $H'(0) = 0$ and $H'(q) = 0 \cdot H(q-1)$ for $q > 0$. This does not change our results in any material way.

$w_i - v_i = b^{-i}$. Initially, $I_0 = (0, 1)$. For each interval I_i, M splits $I_i = (v_i, w_i)$ into b equal-sized intervals

$$(v_i, v_i + b^{-(i+1)}), \ldots, (v_i + (b-1)b^{-(i+1)}, v_i + b^{-i}) = (v_i + (b-1)b^{-(i+1)}, w_i) .$$

The endpoints of all the b intervals do not have to be explicitly computed; in the course of the following binary search we can compute those that we need. This saves some computation time.

For any interval (r_1, r_2), if $D(r_1) = D(r_2) = 0$, then $\alpha > r_2$, and if $D(r_1) = D(r_2) = 1$, then $\alpha < r_1$ (and the case $D(r_1) = 1 \wedge D(r_2) = 0$ is not possible). Thus, M can use D to perform binary search on (the endpoints of) the above set of intervals to find the interval

$$(\ v_i + jb^{-(i+1)} \ , \ v_i + (j+1)b^{-(i+1)} \ ) \tag{6.1}$$

that contains α (observe that, for this interval, $D(v_i + jb^{-(i+1)}) = 0$ and $D(v_i + (j+1)b^{-(i+1)}) = 1$). We then set (v_{i+1}, w_{i+1}) to equal the interval (6.1). By construction, we have $E^\alpha(b, i+1) = j$.

Clearly, in each step i, there are at most $\log b$ oracle calls to D, and the construction of each of the b intervals and writing on the query tape can be performed in time polynomial in the binary representation of the numbers involved, hence in time $O(\text{polylog}(b^i)) = O(\text{poly}(i)\text{polylog}(b))$. Hence, the total time needed to produce $E(b, n)$ is at most $O(\text{poly}(n)\text{polylog}(b))$ with at most $n \log b$ queries to D. In each oracle call, the rational numbers involved are all endpoints of intervals where the endpoints are sums of negative powers of b and where the exponent of all powers is at most n. Hence, all oracle calls can be represented by rational numbers using at most $O(n \log b)$ bits. $\square$

Our algorithm for converting a general base expansion to a Dedekind cut is based on the following observation: For any $n \in \mathbb{N}$ and $m \in \mathbb{N}^+$ such that $0 < n/m < 1$, we have $D^\alpha(n/m) = 0$ iff $n/m < \alpha$ iff $n \leq E^\alpha(m, 1)$.

Lemma 6.3. *There is a parameterized function-oracle Turing machine M with the following properties. Let $E : (\mathbb{N} \setminus \{0, 1\}) \times \mathbb{N}^+ \longrightarrow \mathbb{N}$ be the general base expansion of an irrational $\alpha \in (0, 1)$. Then*

- $\Phi_M^E : \mathbb{Q} \longrightarrow \{0, 1\}$ *is the Dedekind cut of α*

- M^E *on input n/m runs in time $O(\log(\max\{n, m\}))$ and uses exactly one oracle call of input size at most $O(\log m)$.*

Proof. On input $n/m \in \mathbb{Q}$, M first checks if $m = 1$, and outputs 0 if $n \leq 0$ and 1 if $n \geq 1$. Otherwise, $m > 1$, and M computes $E(m, 1)$; by definition, this is an element of $\{0, \ldots, m-1\}$. Thereafter, M outputs 0 if $n \leq E(m, 1)$, and outputs 1 otherwise. M needs to write the (representation of the) pair $(m, 1)$

on the oracle tape and perform a single comparison of numbers of magnitude at most $\max\{n, m\}$, hence M uses time $O(\log \max\{n, m\})$ for the comparison. M uses exactly one oracle call to E with the pair $(m, 1)$, the representation of which uses at most $O(\log m)$ bits. $\qquad\square$

6.4 Conversion between Beatty sequences and Dedekind cuts

It is easy to see how we can convert a Dedekind cut D^α into a Beatty sequence B^α as the value of $B^\alpha(n)$ is the natural number m such that $m/n < \alpha < (m+1)/n$. We may use D^α to search for that unique m.

Lemma 6.4. *There is a parameterized function-oracle Turing machine M with the following properties. Let $D : \mathbb{Q} \longrightarrow \{0, 1\}$ be the Dedekind cut of an irrational $\alpha \in (0, 1)$. Then*

- *$\Phi_M^D : \mathbb{N}^+ \longrightarrow \mathbb{N}$ is the Beatty sequence of α*

- *M^D on input n runs in time $O(\mathrm{polylog}(n))$ and uses at most $\lceil \log n \rceil$ oracle calls, each of input size at most $O(\log n)$.*

Proof. On input n, M finds the least $i \in \{1, \dots, n\}$ such that $D(i/n) = 1$. As $D(i/n) = 1$ and $j > i$ implies $D(j/n) = 1$, the least i can be found by binary search, halving the search range in each step. This can be done by maintaining two integers l and u ranging in $\{0, \dots, n\}$, and requires a maximum of $\log n$ halving steps. In each halving step, M finds the midpoint m between l and u, writes the rational number m/n on the query tape, queries D, and records the answer. Then, l and u are updated using basic binary arithmetic operations on integers, represented by at most $O(\log n)$ bits; if $D(m/n) = 1$, then $u := m$, and if $D(m/n) = 0$, then $l := m$. Eventually we reach a state where $D(l/n) = 0$ and $D(m/n) = D((l+1)/n) = 1$. Clearly, in each step, the arithmetic and update operations can be performed in time polynomial in the size of the representation of the integers, hence in time $\mathrm{polylog}(n)$. As $l/n < \alpha < m/n$, we have $B(n) = l$, and M^D thus returns l. $\qquad\square$

In order to see that our algorithm for converting a Beatty sequence B^α into a Dedekind cut D^α is correct, observe that we have $D^\alpha(n/m) = 0$ iff $n/m < \alpha$ iff $n \leq B^\alpha(m)$, for any $n \in \mathbb{N}$ and any $m \in \mathbb{N}^+$.

Lemma 6.5. *There is a parameterized function-oracle Turing machine M with the following properties. Let $B : \mathbb{N}^+ \longrightarrow \mathbb{N}$ be the Beatty sequence of an irrational $\alpha \in (0, 1)$. Then*

- *$\Phi_M^B : \mathbb{Q} \longrightarrow \{0, 1\}$ is the Dedekind cut of α*

- M^B *on input* n/m *runs in time* $O(\log(\max\{n, m\}))$ *and uses exactly one oracle call of input size* $O(\log m)$.

Proof. On input n/m, M performs the oracle call $B(m)$, resulting in an integer $B(m)$ (where $B(m) \in \{0, 1 \ldots, m-1\}$). If $n \leq B(m)$, then M outputs 0, otherwise, M outputs 1. The comparison $n \leq B(m)$ can be performed bit-wise using the binary representations of n and $B(m)$ which is clearly linear in $\log(\max\{n, B(m)\}) \leq \log\max\{n, m\}$. Writing m on the oracle tape clearly also takes time linear in $\log m$. $\qquad\square$

6.5 Conversion between Hurwitz characteristics and Dedekind cuts

Let us first discuss how we can compute a Dedekind cut D^α using a Hurwitz characteristic H^α as an oracle.

Assume $0 < n/m < 1$ where n and m are relatively prime natural numbers. Let $(a/b, c/d) = \mathcal{T}_F(H^\alpha(n+m))$. By Proposition 2.6, n/m occurs as one of the fractions in a Farey pair in $\mathcal{T}_F$ at depth at most $n+m-2$, and thus exactly one of (i) $n/m \leq a/b$ and (ii) $c/d \leq n/m$ must hold. As $a/b < \alpha < c/d$, we have $D^\alpha(n/m) = 0$ iff $n/m \leq a/b$.

Lemma 6.6. *There is a parameterized function-oracle Turing machine M with the following properties. Let $H : \mathbb{N} \longrightarrow \{0,1\}^*$ be the Hurwitz characteristic of an irrational $\alpha \in (0, 1)$. Then*

- $\Phi_M^H : \mathbb{Q} \longrightarrow \{0, 1\}$ *is the Dedekind cut of* α

- M^H *on input* n/m *runs in time* $\text{poly}(\max\{n, m\})$ *and uses exactly one oracle call of input size at most* $O(\log\max\{n, m\})$.

Proof. If $n/m \leq 0$, then M outputs 0. If $n/m \geq 1$, then M outputs 1. Let $0 < n/m < 1$. We assume that n/m is in its lowest terms. Then M computes $H(n+m)$ using $\text{polylog}(\max\{n, m\})$ operations to compute the binary representation of $n + m$, and then performing a single oracle call; note that the result of the oracle $H(n+m)$ is a bit string of length exactly $n + m \leq \text{poly}(\max\{n, m\})$. M then computes $\mathcal{T}_F(H(n+m))$ to obtain a Farey pair $(a/b, c/d)$ such that $a/b < \alpha < c/d$. If $n/m \leq a/b$, then M outputs 0, otherwise, M outputs 1.

By Proposition 2.4, M can find $(a/b, c/d)$ in time

$$\text{poly}(1 + |H(n+m)|) = \text{poly}(\max\{n, m\})$$

and whether $n/m \leq a/b$ holds can be tested in time $O(\log\max\{a, b, n, m\})$. It is an easy induction on the depth d to see that a numerator or denominator in any fraction occurring in a Farey pair at depth d in $\mathcal{T}_F$ is at most 2^d. Hence, $\max\{a, b, n, m\} \leq 2^{n+m}$, and the test can thus be performed in time $O(n+m) = O(\max\{n, m\})$. Thus, M needs a total time of $\text{poly}(\max\{n, m\})$. $\qquad\square$

Our algorithm for converting a Dedekind cut D^α into a Hurwitz characteristic H^α is not very surprising. The value of $H^\alpha(n)$ is determined by a path of length n in the Farey pair tree $\mathcal{T}_F$ where every interval along the path contains α. We can easily compute such a path when we have access to D^α.

Lemma 6.7. *There is a parameterized function-oracle Turing machine M with the following properties. Let $D : \mathbb{Q} \longrightarrow \{0,1\}$ be the Dedekind cut of an irrational $\alpha \in (0,1)$. Then*

- $\Phi_M^D : \mathbb{N} \longrightarrow \{0,1\}^*$ *is the Hurwitz characteristic of α*

- M^D *on input n runs in time* $\mathrm{poly}(n)$ *and uses exactly n oracle calls, each of input size at most $O(n)$.*

Proof. On input n, M constructs a path of length n in $\mathcal{T}_F$. M does this by starting at $i = 0$ and incrementing i, maintaining a current Farey pair $(a_i/b_i, c_i/d_i)$ such that $\alpha \in (a_i/b_i ; c_i/d_i)$ for $i = 0, \ldots, n$. The mediant of $(a_i/b_i, c_i/d_i)$ gives rise to the two children $(a_i/b_i, (a_i+c_i)/(b_i+d_i))$ and $((a_i+c_i)/(b_i+d_i), c_i/d_i)$ of $(a_i/b_i, c_i/d_i)$ in $\mathcal{T}_F$. The irrational number α will be in exactly one of these two open intervals. Clearly, $\alpha \in (a_i/b_i, (a_i+c_i)/(b_i+d_i))$ iff $D((a_i+c_i)/(b_i+d_i)) = 1$ iff the ith bit of $H(n)$ is 0. Hence, M starts with $(a_0/b_0, c_0/d_0) = (0/1, 1/1)$, and for $i = 1, \ldots, n$, M constructs the interval $(a_i/b_i, c_i/d_i)$ by computing the mediant and querying D. Observe that the query in step i is the (binary representation of the) mediant of a Farey pair at depth $i - 1$, thus its denominator is bounded above by 2^i and its binary representation uses at most $O(\log 2^i) = O(i)$ bits.

As the numerators and denominators at depth i in $\mathcal{T}_F$ are of size at most 2^i (hence representable by i bits), computing the mediant at step i can be done in time at most $O(i) = O(n)$ by two standard schoolbook additions, and the step i contains exactly one query to D. Hence, the total time needed for M to construct $H(n)$ is at most $O(n\,\mathrm{poly}(n)) = \mathrm{poly}(n)$, with exactly n oracle calls, each of size at most $O(\log 2^n) = O(n)$. $\qquad\square$

6.6 Summary

We can now give a summary of our results on the complexity of conversions among representations subrecursively equivalent to representation by Dedekind cuts.

Theorem 6.8. *Consider the representations by (1) Dedekind cuts, (2) general base expansions, (3) Hurwitz characteristics and (4) Beatty sequences, and let R_1 and R_2 be any two of these four representations. Then, for an arbitrary time-bound t, we have*

$$O(t)_{R_2} \subseteq O(\mathrm{poly}(2^{2^n} + t(O(2^n))))_{R_1} \ .$$

43

Proof. As in the proof of Theorem 4.15, we have to consider composition of reductions. In this case all paths go through the representation by Dedekind cuts (see Figure 3). The formulas next to each arc are justified by the corresponding lemma. For example, computation of a general base expansion from a Dedekind cut is described by Lemma 6.2. It states that on input (b, n), the machine runs in time $\mathrm{poly}(n)\mathrm{polylog}(b)$. Let x be the size of the input (in Figure 3 it is denoted by n, but we change here to x to avoid a clash). I.e., $x = \|(b, n)\|$. Then it should be clear that $\mathrm{poly}(n)\mathrm{polylog}(b) = \mathrm{poly}(2^x)$. The reduction uses at most $n \log b$ oracle calls, this is bounded by 2^x. The input size to the oracle call is bounded by $n \log b$, this is $O(2^x)$.

Now, consider the conversion from Hurwitz characteristic to a general base expansion via the Dedekind cut—this turns out to be the costliest path. Let t_D be the time to compute the Dedekind cut from the Hurwitz characteristic. Then computation of the general base expansion takes $\mathrm{poly}(2^n)$ time using 2^n oracle calls of size $O(2^n)$: we get the formula

$$\mathrm{poly}(2^n) + 2^n \cdot t_D(O(2^n)).$$

Now we substitute $t_D(x) = O(2^x) + t(x)$, to get

$$\mathrm{poly}(2^n) + 2^n \cdot (2^{O(2^n)} + t(O(2^n)))$$

which simplifies to

$$\mathrm{poly}(2^{2^n}) + 2^n \cdot t(O(2^n))$$

which again simplifies to $\mathrm{poly}(2^{2^n} + t(O(2^n)))$. $\qquad\square$

7 Representations equivalent to best approximations

7.1 Best approximations

The representation of real numbers by left (or right) best approximations might not be very well known, but it is a natural representation which is easy to understand, and the next definition should not require any explanations.

Definition 7.1. Let α be an irrational number in the interval $(0, 1)$, and let $r = a/b$ where a, b are relatively prime natural numbers.

The rational r is a *left best approximant* of α if we have $c/d \leq a/b < \alpha$ or $\alpha < c/d$ for any natural numbers c, d where $d \leq b$. The rational r is a *right best approximant* of α if we have $\alpha < a/b \leq c/d$ or $c/d < \alpha$ for any natural numbers c, d where $d \leq b$.
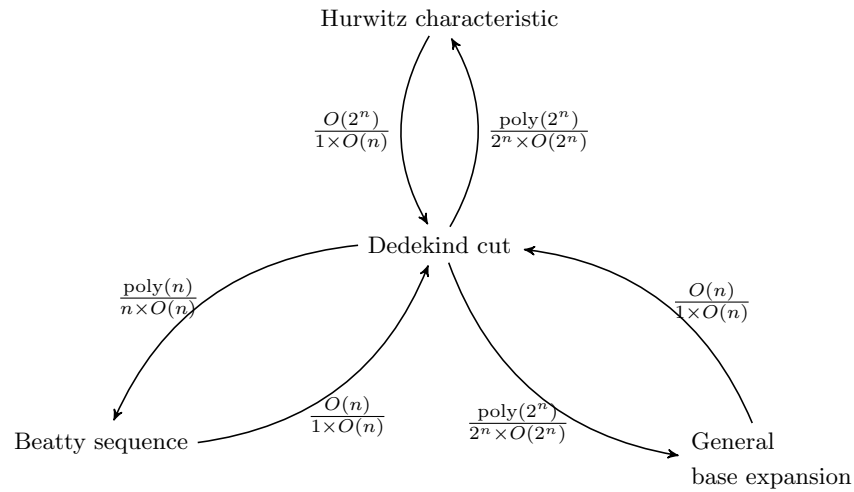
44

Figure 3: Reductions among representations in the cluster of Dedekind Cuts. Next to each arrow representing a reduction is the cost of the reduction (above the line) and the number and size of oracle calls (below).

A *left best approximation* of α is a sequence of rationals $\{r_i\}_{i \in \mathbb{N}}$ such that

$$0 = r_0 < r_1 < r_2 < \dots$$

and each r_i is a left best approximant of α. A *right best approximation* of α is a sequence of rationals $\{r_i\}_{i \in \mathbb{N}}$ such that

$$1 = r_0 > r_1 > r_2 > \dots$$

and each r_i is a right best approximant of α. □

Note that there is at most one left (right) best approximant of α for any given denominator, and thus, a left (right) best approximation of α will converge to α.

In this section we will study a number of representations subrecursively equivalent to the representation by left best approximations and a number of representations subrecursively equivalent to the representation by right best approximations. These two equivalence classes are incomparable to each other, that is, a representation in one of the classes cannot be subrecursively converted to a representation in the other class. We will explain why in Section 8.

Definition 7.2. A left best approximation $\{r_i\}_{i \in \mathbb{N}}$ of α is *complete* if every left best approximant of α occurs in the sequence $\{r_i\}_{i \in \mathbb{N}}$. A right best approximation $\{r_i\}_{i \in \mathbb{N}}$ of α is *complete* if every right best approximant of α occurs in the sequence. □

45

There is a connection between complete best approximations and paths in the Farey pair tree $\mathcal{T}_F$. Let α be an irrational in the interval $(0, 1)$, and let $\sigma_1 \sigma_2 \cdots \in \{0, 1\}^\omega$ be the unique path in $\mathcal{T}_F$ such that

$$\alpha \in \mathcal{T}_F(\sigma_1 \ldots \sigma_n) = (a_n/b_n, c_n/d_n) \tag{7.1}$$

holds for any $n \in \mathbb{N}$. Then, a fraction p/q is a left best approximant of α if and only if p/q occurs in the sequence $a_0/b_0, a_1/b_1, a_2/b_2, \ldots$.

It is obvious that every a_i/b_i in the sequence is a left best approximant of α as any fraction in the interval $(a_i/b_i, c_i/d_i)$ has denominator strictly greater than b_i (see Theorem 2.2). To see that every left best approximant of α indeed occurs in the sequence, assume for the sake of contradiction that a left best approximant p/q is not there. Then we have

$$\frac{a_i}{b_i} \ < \ \frac{p}{q} \ < \ \frac{a_{i+1}}{b_{i+1}} \ = \ \frac{a_i + c_i}{b_i + d_i}$$

for some i. Since p/q is a left best approximant we must have $q < b_{i+1} = b_i + d_i$. But we also have $p/q \in (a_i/b_i, c_i/d_i)$, and that contradicts Theorem 2.2 which states that any fraction in the interval $(a_i/b_i, c_i/d_i)$ has denominator greater than or equal to $b_i + d_i$.

By the same token, p/q is a right best approximant of α if and only if p/q occurs in the sequence $c_0/d_0, c_1/d_1, c_2/d_2, \ldots$. Hence, we have the next lemma.

Lemma 7.3. *Let α be an irrational number such that (7.1) holds. (i) The sequence $\{a_i/b_i\}_{i \in \mathbb{N}}$ contains all the left best approximants of α and nothing but left best approximants of α. (ii) The sequence $\{c_i/d_i\}_{i \in \mathbb{N}}$ contains all the right best approximants of α and nothing but right best approximants of α.*

We can use $\mathcal{T}_F$ to subrecursively compute the complete left best approximation $\{a_i/b_i\}_{i \in \mathbb{N}}$ of α from an arbitrary left best approximation $\{\hat{a}_i/\hat{b}_i\}_{i \in \mathbb{N}}$ of α. Observe that we have $a_i/b_i \leq \hat{a}_i/\hat{b}_i < \alpha$ for any $i \in \mathbb{N}$. Thus, there will be at least n left best approximants of α that are smaller than or equal to $\hat{a}_n/\hat{b}_n$. Hence we can find a_n/b_n by constructing a path $\sigma_1 \ldots \sigma_m$ in $\mathcal{T}_F$ such that $\hat{a}_n/\hat{b}_n$ is the left endpoint of $\mathcal{T}_F(\sigma_1 \ldots \sigma_m)$. By Lemma 7.3, every left best approximant of α smaller than or equal to $\hat{a}_n/\hat{b}_n$ will occur along the path $\sigma_0 \ldots \sigma_m$. We have argued that there will be at least n of them, and thus we can pick the nth one. By Proposition 2.6, we have $m \leq \hat{a}_n + \hat{b}_n - 2$ (except for $\hat{a}_0 = 0$ where we have $m = 0$), and thus, unbounded search is not required. A symmetric algorithm will compute the complete right best approximation from an arbitrary right best approximation.

Lemma 7.4. *There is a parameterized function-oracle Turing machine M with the following properties. Let $L : \mathbb{N} \longrightarrow \mathbb{Q}$ be a left best approximation of an irrational $\alpha \in (0, 1)$. Then*

- $\Phi_M^L : \mathbb{N} \longrightarrow \mathbb{Q}$ *is the complete left best approximation of α*

46

- M^L *on input n runs in time* $\text{poly}(2^{\|L(n)\|})$ *and uses exactly one oracle call of input size* $\|n\|$.

Proof. By the algorithm shown above, noting that we trace in the Farey tree a path of length at most $\max(0, \hat{a}_n + \hat{b}_n - 2)$. The arithmetic operations performed by the algorithm take polynomial time in the length of the operands. □

Given our discussion above, it is not very hard to see that the representation by Dedekind cuts is subrecursive in the representation by left best approximations and also in the representation by right best approximations: If $\{a_i/b_i\}_{i \in \mathbb{N}}$ is a left best approximation of α, then we have $b_n > n$, and thus also $m/n < \alpha$ iff $m/n < a_n/b_n$. If $\{c_i/d_i\}_{i \in \mathbb{N}}$ is a right best approximation of α, then we have $d_n > n$, and thus also $\alpha < m/n$ iff $c_n/d_n < m/n$.

Lemma 7.5. *There is a parameterized function-oracle Turing machine M with the following properties. Let $L : \mathbb{N} \longrightarrow \mathbb{Q}$ be a left best approximation of an irrational $\alpha \in (0, 1)$. Then*

- $\Phi^L_M : \mathbb{Q} \longrightarrow \{0, 1\}$ *is the Dedekind cut of α*

- M^L *on input m/n runs in time* $\text{poly}(\|L(n)\|)$ *and uses exactly one oracle call of input size* $\|n\|$.

Proof. We ask the oracle for the mth left best approximant a_n/b_n. Then we have $m/n < a_n/b_n < \alpha$ or $\alpha < m/n$. Thus, using a single oracle call and a comparison of rationals, we can decide whether $m/n < \alpha$. □

7.2 Definitions

We will now define and explain a few representations subrecursively equivalent to left, or right, best approximations.

The base-b sum approximation from below (above) of α, denoted $\hat{A}^\alpha_b$ ($\check{A}^\alpha_b$), is defined and discussed in Section 5, see Definition 5.6. The general sum approximation from below (above) of α encompasses the base-b sum approximation from below (above) of α for any base b. The formal definition follows.

Definition 7.6. The *general sum approximation from below* of α is the function $\hat{G}^\alpha : (\mathbb{N} \setminus \{0, 1\}) \times \mathbb{N} \longrightarrow \mathbb{Q}$ given by $\hat{G}^\alpha(b, n) = \hat{A}^\alpha_b(n)$. The *general sum approximation from above* of α is the function $\check{G}^\alpha : (\mathbb{N} \setminus \{0, 1\}) \times \mathbb{N} \longrightarrow \mathbb{Q}$ given by $\check{G}^\alpha(b, n) = \check{A}^\alpha_b(n)$. □

What we will call a *Baire sequence* is an infinite sequence of natural numbers. Such a sequence $a_0, a_1, a_2, \ldots$ represents an irrational number α in the interval $(0, 1)$. We split the interval $(0, 1)$ into infinitely many open subintervals with rational endpoints. Specifically, we use the splitting

$$(\, 0/1 \, , \, 1/2 \,) \, (\, 1/2 \, , \, 2/3 \,) \, (\, 2/3 \, , \, 3/4 \,) \, \ldots \, (\, n/(n+1) \, , \, (n+1)/(n+2) \,) \ldots .$$

The first number of the sequence a_0 tells us in which of these intervals we find α. Thus if $a_0 = 17$, we find α in the interval $(17/18, 18/19)$. Then we split the interval $(17/18, 18/19)$ in a similar way. The second number of the sequence a_1 tells us in which of these intervals we find α, and thus we proceed.

In general, in order to split the interval (q, r), we need a strictly increasing sequence of rationals $s_0, s_1, s_2, \ldots$ such that $s_0 = q$ and $\lim_i s_i = r$. We will use the splitting $s_i = (a + ic)/(b + id)$ where a, b are (the unique) relatively prime natural numbers such that $q = a/b$ and c, d are (the unique) relatively prime natural numbers such that $r = c/d$ (let $0 = 0/1$ and $1 = 1/1$). This particular splitting ensures that every interval induced by a Baire sequence can be found in the Farey pair tree $\mathcal{T}_F$.

We will say that the Baire sequences explained above are *standard*. The standard Baire sequence of the irrational number α will lexicographically precede the standard Baire sequence of the irrational number β iff $\alpha < \beta$. We will also work with what we call *dual* Baire sequences. The dual sequence of α will lexicographically precede the dual sequence of β iff $\alpha > \beta$. We get the dual sequences by using decreasing sequences of rationals to split intervals, e.g., the interval $(0, 1)$ will be split into the intervals

$$(\; 1/1 \; , \; 1/2 \;) \; (\; 1/2 \; , \; 1/3 \;) \; (\; 1/3 \; , \; 1/4 \;) \; \ldots \; (\; 1/n \; , \; 1/(n+1) \;) \; \ldots \quad .$$

Definition 7.7. Let $f : \mathbb{N} \longrightarrow \mathbb{N}$ be any function, and let $n \in \mathbb{N}$. We define the interval I_f^n by $I_f^0 = (0/1, 1/1)$ and

$$I_f^{n+1} \;\; = \;\; \left(\; \frac{a + f(n)c}{b + f(n)d} \; , \; \frac{a + f(n)c + c}{b + f(n)d + d} \; \right)$$

if $I_f^n = (a/b, c/d)$. We define the interval J_f^n by $J_f^0 = (0/1, 1/1)$ and

$$J_f^{n+1} \;\; = \;\; \left(\; \frac{a + f(n)a + c}{b + f(n)b + d} \; , \; \frac{f(n)a + c}{f(n)b + d} \; \right)$$

if $J_f^n = (a/b, c/d)$. The function $B : \mathbb{N} \longrightarrow \mathbb{N}$ is the *standard Baire representation* of the irrational number $\alpha \in (0, 1)$ if we have $\alpha \in I_B^n$ for every n. The function $A : \mathbb{N} \longrightarrow \mathbb{N}$ is the *dual Baire representation* of the irrational number $\alpha \in (0, 1)$ if we have $\alpha \in J_A^n$ for every n. □

Unit fractions, that is, fractions with numerator 1, were studied in ancient Egypt, see e.g. [2], and are also known as Egyptian fractions. In the literature, an *Egyptian fraction expansion* may refer to any representation of a number as a sum of fractions with numerator 1. The definition we give below ensures that any irrational number in the interval $(0, 1)$ has a unique Egyptian fraction expansion, see Cohen [14].

Definition 7.8. The function $E^\alpha : \mathbb{N}^+ \longrightarrow \mathbb{N}$ is the *Egyptian fraction expansion* for α if

$$\alpha = \sum_{i=1}^{\infty} \left(\prod_{j=1}^{i} E^\alpha(j) \right)^{-1}$$

48

and $E^\alpha(i) \le E^\alpha(i+1)$ (for all $i \in \mathbb{N}^+$). $\qquad\qquad\square$

We have e.g.

$$\sqrt{2} - 1 = \frac{1}{3} + \frac{1}{3 \cdot 5} + \frac{1}{3 \cdot 5 \cdot 5} + \frac{1}{3 \cdot 5 \cdot 5 \cdot 16} + \frac{1}{3 \cdot 5 \cdot 5 \cdot 16 \cdot 18} + \ldots$$

and this is the unique Egyptian fraction expansion of $\sqrt{2} - 1$. Another possible representation of irrationals based on Egyptian fractions is related to left best approximation, see Beck et al. [6]. In fact, from Theorem 2.2 and the relation of left best approximations to the Farey pair tree, (Lemma 7.3) it is easy to deduce that the difference between consecutive fractions in a complete left best approximation is a unit fraction. Thus a complete left best approximation of α induces a series of unit fractions that adds up to α.

Definition 7.9. A sequence $\{q_i\}_{i \in \mathbb{N}}$ of non-negative unit fractions is the Farey-Egyptian expansion of $\alpha \in (0,1)$ if the sequence $\{\sum_{i=0}^{j} q_i\}_{j \in \mathbb{N}}$ is a complete left best approximation of α (which entails $\sum_{i=1}^{\infty} q_i = \alpha$). $\qquad\square$

For example

$$\sqrt{2} - 1 = \frac{0}{1} + \frac{1}{3} + \frac{1}{15} + \frac{1}{85} + \frac{1}{493} + \ldots$$

is a Farey-Egyptian expansion, associated with the complete left best approximation $0/1,\ 1/3,\ 2/5,\ 7/17, \ldots$.

The representation by Farey-Egyptian expansions is closely related to the representation by complete left best approximations, and it is easy to convert Farey-Egyptian expansion into a complete left best approximation, and vice versa.

General sum approximations (from above and below) were introduced in Kristiansen [31] and studied further in Georgiev et al. [20]. Left and right best approximations are studied in [20], and standard and dual Baire sequences are studied in Kristiansen [33]. The computational complexity of representations by Egyptian fraction expansions is studied for the first time in this paper.

7.3 Conversion between general sum approximations and best approximations

Let $\{a_i/b_i\}_{i \in \mathbb{N}}$ be a left best approximation of β. We will give an algorithm for computing the general sum approximation from below of β, that is $\hat{G}^\beta$, using $\{a_i/b_i\}_{i \in \mathbb{N}}$ as an oracle. The ith iteration of the algorithm generates $\hat{G}^\beta(b,i)$. Having computed $\hat{G}^\beta(b,i)$ for all $i < n$, the algorithm will also have computed the sum

$$\frac{c}{d} = \sum_{i=1}^{n-1} \hat{G}^\beta(b,i) .$$

Then the algorithm computes $\hat{G}^\beta(b,n)$ by executing the following instructions.

49

- Step 1: Ask the oracle for the value of a_d/b_d. Let $c'/d' = a_d/b_d$. Note that d is a power of b; and that we have $d' > d$, and $c/d < c'/d' < \beta$.

- Step 2: Compute $m := \lceil \log_b d'd \rceil$. *Comment*: We have $c'/d' - c/d > 0$, and thus also

$$\frac{c'}{d'} - \frac{c}{d} = \frac{c'd - cd'}{d'd} \;\geq\; \frac{1}{d'd} \; .$$

It follows that $c/d + 1/b^m \leq c'/d' < \beta$.

- Step 3: Ask the oracle for the value of a_{b^m}/b_{b^m} and compute (as further explained below) the least $k \leq m$ and base-b digit $\mathtt{D} > 0$ so that

$$\frac{c}{d} + \frac{\mathtt{D}}{b^k} \;<\; \beta \;<\; \frac{c}{d} + \frac{\mathtt{D}}{b^k} + \frac{1}{b^k} \; . \tag{7.2}$$

Give the output $\mathtt{D}/b^k$.

Comments on Step 3: We have $i < b_i$ for any left best approximant a_i/b_i. Hence, for any fraction p/b^k, with $k \leq m$, we have $p/b^k < \beta$ iff $p/b^k \leq a_{b^m}/b_{b^m}$, and (7.2) is equivalent, for such k, to

$$\frac{c}{d} + \frac{\mathtt{D}}{b^k} \;\leq\; \frac{a_{b^m}}{b_{b^m}} \;<\; \frac{c}{d} + \frac{\mathtt{D}}{b^k} + \frac{1}{b^k} \; . \tag{7.3}$$

The least k that satisfies (7.3) for some $\mathtt{D} > 0$ is the least k that satisfies

$$\frac{c}{d} + \frac{1}{b^k} \;\leq\; \frac{a_{b^m}}{b_{b^m}} \; .$$

In order to find that k, we rewrite the inequality as

$$\frac{1}{b^k} \;\leq\; \frac{a_{b^m}}{b_{b^m}} - \frac{c}{d}$$

which again can be rewritten as

$$b^k \;\geq\; \frac{d b_{b^m}}{d a_{b^m} - c b_{b^m}} \; .$$

Hence, the desired k is

$$k = \left\lceil \log_b \frac{d b_{b^m}}{d a_{b^m} - c b_{b^m}} \right\rceil \; .$$

Having computed k, we look for a value of $\mathtt{D}$ such that (7.3) holds, and it should be clear that

$$\mathtt{D} = \left\lfloor \left(\frac{a_{b^m}}{b_{b^m}} - \frac{c}{d} \right) \cdot b^k \right\rfloor \; .$$

The correctness of the algorithm follows straightforwardly from the comments on Steps 1–3 and the definition of a general sum approximation from below. A right best approximation can be converted into a general sum approximation from above by a symmetric algorithm.

50

Lemma 7.10. *There is a parameterized function-oracle Turing machine M with the following properties. Let $L : \mathbb{N} \longrightarrow \mathbb{Q}$ be a left best approximation of an irrational $\alpha \in (0, 1)$. Assume L is computable within a time bound s, and let $f(b, n) = (\lambda x.(\|b\| + 2s(x)))^{(n)}(1)$. Then*

- $\Phi_M^L : (\mathbb{N} \setminus \{0, 1\}) \times \mathbb{N} \longrightarrow \mathbb{Q}$ *is the general sum approximation from below of α*

- M^L *on input b, n runs in time $\mathrm{poly}(s(f(b, n)))$ and uses at most $2n$ oracle calls, each of input size at most $f(b, n)$.*

Proof. The proof is, of course, inductive. We claim that the bit-lengths of d, b^m are bounded by $f(b, n)$. Consequently, by assumption and the rule that the running time of a machine bounds the size of its output, we also have that the bit-lengths of the denominators b_d, b_{b^m} are bounded by $s(f(b, n))$. The induction step works as follows: The algorithm first sets $d' = b_d$. We inductively assume the bound $f(b, n-1)$ for the bit-length of the denominators of each of $\hat{G}^\alpha(b, i)$, $i < n$. The common denominator d is the denominator of the last term, and so its bit-length is bounded by $f(b, n-1)$. Applying our assumption of the time bound s for the computation of b_d, we have $\|d'\| \leq s(\|d\|) \leq s(f(b, n-1))$. We bound the size of the next denominator, d_{b^m} as follows. First,

$$
\begin{aligned}
\|b^m\| = \|b^{\lceil \log_b(dd') \rceil}\| &\leq \|bdd'\| \\
&\leq \|b\| + \|d\| + \|d'\| \leq \|b\| + f(b, n-1) + s(f(b, n-1)) \leq \\
&\qquad \|b\| + 2s(f(b, n-1)) = f(b, n)
\end{aligned}
$$

and once more, by the assumption of the time bound s, we have

$$
\|d_{b^m}\| \leq s(\|b^m\|) \leq s(f(b, n)) .
$$

The execution time is polynomial in the size of the numbers manipulated, hence polynomial in $s(f(b, n))$. $\qquad \square$

Next we give an algorithm for computing a complete left best approximation $\{a_i/b_i\}_{i \in \mathbb{N}}$ of β which uses $\hat{G}^\beta$ (the general sum approximation from below of β) as an oracle. Observe that the oracle makes it easy to compute the Dedekind cut D^β of β: for $d > 1$, we have $D^\beta(c/d) = 0$ iff $\hat{G}^\beta(d, 1) \geq c/d$.

To compute a_0/b_0 is trivial since we have $a_0/b_0 = 0/1$ by convention. In order to compute a_1/b_1 the algorithm asks the oracle for the value of $\hat{G}^\beta(2, 1)$. We have $\hat{G}^\beta(2, 1) = 1/M$ for some M. Then the algorithm uses the Dedekind cut of β to search for $N \leq M$ such that $1/N < \beta < 1/(N-1)$ and sets a_1/b_1 to $1/N$. (Such an N will exist since $1/M < \beta$. Note that $2/N \geq 1/(N-1) > \beta$, and hence, the algorithm computes a_1/b_1 correctly.)

When $n \geq 1$, the algorithm computes a_{n+1}/b_{n+1} by the following procedure:

51

- Let

$$\frac{c'}{d'} \;=\; \frac{a_n}{b_n} \;+\; \hat{G}^{\beta}(b_n, 2)$$

with relatively prime c' and d'. We have $d' > b_n$ because a_n/b_n is a best approximant.

- Use the Dedekind cut of β to search for c and the smallest d such that $b_n < d \le d'$ and $a_n/b_n < c/d < \beta < (c+1)/d$. Let $a_{n+1}/b_{n+1} = c/d$.

In order to see that the algorithm is correct, observe that $a_n/b_n = \hat{G}^{\beta}(b_n, 1)$. Hence, we have $a_n/b_n < c'/d' < \beta < (c'+1)/d'$, which ensures that the denominator b_{n+1} is at most d'. Therefore we find a complete best approximation.

A general sum approximation from above can be converted into a complete right best approximation by a symmetric algorithm.

Lemma 7.11. *There is a parameterized function-oracle Turing machine M with the following properties. Let $G : (\mathbb{N} \setminus \{0,1\}) \times \mathbb{N} \longrightarrow \mathbb{Q}$ be the general sum approximation from below of an irrational $\beta \in (0,1)$. Assume that G is computable within a time bound s, and let $f(n) = (\lambda x.s(2x))^{(n)}(1)$. Then*

- $\Phi_M^G : \mathbb{N} \longrightarrow \mathbb{Q}$ *is the complete left best approximation of β*

- M^G *on input n runs in time* $\mathrm{poly}(2^{f(n)})$ *and uses at most* $O(2^{f(n)})$ *oracle calls, each of input size at most $f(n)$.*

Proof. First we explain the implementation of the search for c, d in the inductive step, when computing a_n/b_n. The condition we test is $b_{n-1} < d \le d'$ and $a_{n-1}/b_{n-1} < c/d < \beta < (c+1)/d$. The oracle can be used to replace the second conjunct by $a_{n-1}/b_{n-1} < c/d = G(d, 1)$. It follows that to perform the search, we first ask for $G(b_{n-1}, 2)$, then for $G(d, 1)$ for successive values of d, from b_{n-1} upwards, and at most up to the denominator of $G(b_{n-1}, 2)$. We conclude that the largest number involved in computing a_n/b_n is $G(b_{n-1}, 2)$.

Now, consider the whole process of computing $a_0/b_0, \dots, a_n/b_n$: in the process of computing a_{i+1}/b_{i+1} we make a single oracle call to bound the search, and then we search from $b_i + 1$ up to b_{i+1} using a single oracle call for each test. It is easy to see that the total number of oracle calls is $n + b_n$. To get a bound in terms of the input to the algorithm, we note that d' in the induction step is at most the denominator of $G(b_{n-1}, 2)$. By assumption, $G(b_{n-1}, 2)$ occupies at most $s(2\|b_{n-1}\|)$ bits. Hence

$$b_n < 2^{s(2\|b_{n-1}\|)}$$

so $\|b_n\| \le s(2\|b_{n-1}\|)$. This gives the bound $f(n)$ on the bit-length of the largest number involved in the computation. The value of this number, bounded by $2^{f(n)}$, bounds the number of *steps* in the computation, and hence the execution time up to a polynomial. $\square$

7.4 Conversion between Baire sequences and best approximations

Lemma 7.12. *We have*

$$I_f^{n+1} = \mathcal{T}_F(1^{f(0)}01^{f(1)}0\ldots 1^{f(n)}0) \tag{i}$$

and

$$J_f^{n+1} = \mathcal{T}_F(0^{f(0)}10^{f(1)}1\ldots 0^{f(n)}1) . \tag{ii}$$

Proof. We prove (i). The proof of (ii) is symmetric. Let $\sigma = 1^{f(0)}01^{f(1)}0\ldots 1^{f(n-1)}0$. Observe that we have $\mathcal{T}_F(\sigma) = (0/1, 1/1) = I_f^0$ when σ is the empty sequence. Assume that $\mathcal{T}_F(\sigma) = I_f^n = (a/b, c/d)$. We need to prove that

$$\mathcal{T}_F(\sigma 1^{f(n)}0) = I_f^{n+1} . \tag{7.4}$$

Let $k = f(n)$.

Assume $k = 0$. By Definition 2.3, we have

$$\mathcal{T}_F(\sigma 1^{f(n)}0) = \mathcal{T}_F(\sigma 1^0 0) = \mathcal{T}_F(\sigma 0) = (\, a/b, (a+c)/(b+d) \,) .$$

By Definition 7.7, we have

$$I_f^{n+1} = (\, (a+kc)/(b+kd), (a+kc+c)/(b+kd+d) \,) = (\, a/b, (a+c)/(b+d) \,) .$$

Thus, (7.4) holds when $f(n) = 0$.

Next, we prove by induction on k that

$$\mathcal{T}_F(\sigma 1^k 0) = \left(\frac{a+kc}{b+kd}, \frac{a+kc+c}{b+kd+d} \right) . \tag{7.5}$$

Observe that the right-hand side of (7.5) is the definition of I_f^{n+1} with k for $f(n)$. Now, by (7.5) and Definition 2.3, we have

$$\mathcal{T}_F(\sigma 1^k) = \left(\frac{a+kc}{b+kd}, \frac{c}{d} \right) . \tag{7.6}$$

Furthermore, by (7.6) and Definition 2.3, we have

$$\mathcal{T}_F(\sigma 1^{k+1}) = \left(\frac{a+kc+c}{b+kd+d}, \frac{c}{d} \right) = \left(\frac{a+(k+1)c}{b+(k+1)d}, \frac{c}{d} \right) \tag{7.7}$$

and by (7.7) and Definition 2.3, we have

$$\mathcal{T}_F(\sigma 1^{k+1}0) = \left(\frac{a+(k+1)c}{b+(k+1)d}, \frac{a+(k+1)c+c}{b+(k+1)d+d} \right) . \tag{7.8}$$

Observe that the right-hand side of (7.8) is the definition of I_f^{n+1} with $k+1$ for $f(n)$. This proves that (7.4) holds. $\qquad\square$

53

Given the lemma above it is easy to see how we can convert a standard Baire sequence B into a complete right best approximation. We use B to compute an interval I in the Farey pair tree such that $I = \mathcal{T}_{\mathrm{F}}(1^{B(0)}01^{B(1)}0\ldots 1^{B(n)}0)$. By Lemma 7.12 (i), we have $I = I_B^{n+1}$. It follows that the right endpoint of I is the $n+1$st approximant in the complete right best approximation of α (see Lemma 7.3).

By Lemma 7.12 (ii), we have a symmetric algorithm for converting a dual Baire sequence into a complete left best approximation.

Lemma 7.13. *There is a parameterized function-oracle Turing machine M with the following properties. Let $B : \mathbb{N} \longrightarrow \mathbb{N}$ be the standard Baire sequence of an irrational $\alpha \in (0,1)$. Then*

- $\Phi_M^B : \mathbb{N} \longrightarrow \mathbb{Q}$ *is the complete right best approximation of α*

- M^B *on input $n \in \mathbb{N}$ runs in time* $\mathrm{poly}(n + \sum_{i=0}^{n-1} B(i))$ *and uses n oracle calls, each of input size $O(\log n)$.*

Proof. We have to compute $I = \mathcal{T}_{\mathrm{F}}(1^{B(0)}01^{B(1)}0\ldots 1^{B(n-1)}0)$. By Proposition 2.4 this is polynomial in $n + \sum_{i=0}^{n-1} B(i)$.[4] $\qquad\square$

Lemma 7.12 (i) also yields an algorithm for converting complete right best approximations into standard Baire sequences: Given the complete right best approximation $\{a_i/b_i\}_{i\in\mathbb{N}}$ of α, we can compute a (unique) string of the form $1^{k_0}01^{k_1}0\ldots 1^{k_n}0$ such that the right endpoint of the interval $\mathcal{T}_{\mathrm{F}}(1^{k_0}01^{k_1}0\ldots 1^{k_i}0)$ equals $\{a_{i+1}/b_{i+1}\}$ (for all $i \leq n$). By Lemma 7.12 (i), we have $B(i) = k_i$ where B is the standard Baire sequence of α. Lemma 7.12 (ii) yields an algorithm for converting complete left best approximations into dual Baire sequences.

Lemma 7.14. *There is a parameterized function-oracle Turing machine M with the following properties. Let $R : \mathbb{N} \longrightarrow \mathbb{Q}$ be the complete right best approximation of an irrational $\alpha \in (0,1)$. Then*

- $\Phi_M^R : \mathbb{N} \longrightarrow \mathbb{N}$ *is the standard Baire sequence of α*

- M^R *on input n runs in time* $\mathrm{poly}(2^{\|R(n+1)\|})$ *and uses n oracle calls, each of input size $O(\log n)$*

Proof. The machine has to trace a path in $\mathcal{T}_{\mathrm{F}}$ from the root up to the first occurrence of $R(n+1) = a_{n+1}/b_{n+1}$. By Proposition 2.6, this happens at most at depth $\max(0, a_{n+1} + b_{n+1} - 2) < 2^{\|R(n+1)\|}$. The branch taken at each level is dictated by the corresponding best approximant, so the machine has to compute $R(0)$ through $R(n)$. The work at each level is polynomial in the depth and the size of the numbers involved (see Lemma 2.7), which are all polynomial in $a_{n+1} + b_{n+1}$, hence in $2^{\|R(n+1)\|}$.[5] $\qquad\square$

---

[4] If B is computable within the time bound s, then $s(\|n\|)$ can be used to bound the size of the function values, and consequently M^B runs in time $\mathrm{poly}(n2^{s(\|n\|)})$.

[5] By the standard argument, this shows that if R is computable within the time bound s, then M^R runs in time $\mathrm{poly}(2^{s(\|n\|)})$.

7.5 From general sum approximation from below to Egyptian fraction expansions

Let $\hat{G}^\alpha : (\mathbb{N} \setminus \{0,1\}) \times \mathbb{N} \longrightarrow \mathbb{Q}$ be the general sum approximation from below of $\alpha \in (0,1)$. We show how to convert $\hat{G}^\alpha$ to the Egyptian fraction expansion E^α.

The algorithm works recursively. For the base case, assume $\hat{G}^\alpha(2,1) = 1/b$ (where b is some power of 2). Then we search for the least $2 \leq d \leq b$ such that the denominator of $\hat{G}^\alpha(d,1)$ equals d. This means that $1/d < \alpha < 1/(d-1)$, and we set $E^\alpha(1) = d$. In the general case, first the algorithm computes $E^\alpha(i)$ for all $i \leq n$, then the algorithm computes $E^\alpha(n+1)$ as follows:

- Step 1: Compute the sum of the first n terms in the Egyptian fraction expansion
$$\frac{c}{d} = \sum_{i=1}^{n} \left(\prod_{j=1}^{i} E^\alpha(j) \right)^{-1} .$$

- Step 2: Let $c'/d' = \hat{G}^\alpha(d,1) + \hat{G}^\alpha(d,2)$. Now d' is some power of d, and $c/d < c'/d' < \alpha$.

- Step 3: Search for the least m such that $c/d + (dm)^{-1} < \alpha$. This search can be performed by the Dedekind cut of α, in turn simulated using $\hat{G}^\alpha$ as already shown. The search is bounded since $E^\alpha(n) \leq m \leq d'/d$. Return m.

Except for the effort to bound the searches, our algorithm is the natural greedy algorithm which is known to compute the Egyptian fraction expansion, see Cohen [14].

Lemma 7.15. *There is a parameterized function-oracle Turing machine M with the following properties. Let $G : (\mathbb{N} \setminus \{0,1\}) \times \mathbb{N} \longrightarrow \mathbb{Q}$ be the general sum approximation from below of an irrational number $\alpha \in (0,1)$. Assume that G is computable within a time bound s, and let $f(n) = (\lambda x.s(2x))^{(n)}(2)$. Then*

- *$\Phi_M^G : \mathbb{N}^+ \longrightarrow \mathbb{N}$ is the Egyptian fraction expansion of α*

- *M^G on input n runs in time $\mathrm{poly}(2^{f(n)})$ and uses at most $2^{f(n)+1}$ oracle calls, each of input size at most $2f(n)$.*

Proof. The crucial quantity that determines the complexity of this algorithm is the size (bit-length) of the largest denominator encountered, denoted d' in the above algorithm. We bound the size of d' by $f(n)$ as follows. In the general case, d' is obtained by calling $G(d,2)$ where d comes from the previous iteration, so we assume for induction that $\|d\| \leq f(n-1)$. The input $d,2$ is represented in twice the bit-length of d (assuming that a pair of integers is represented by zipping two binary numbers, as suggested in Section 2), so the size of the oracle

55

input is bounded by $2f(n-1)$ and the output of the oracle has size bounded by $s(2f(n-1)) = f(n)$. In the base case, we query the oracle for $G(2,1)$: the size of the result is bounded by $s(4) = (\lambda x.s(2x))(2)$.

Thus $2f(n)$ bounds the size of any oracle input throughout the algorithm. The number of oracle calls in the last iteration of the general case is bounded by $1 + (E(n) - E(n-1) + 1)$, where the first call is to determine d', and the rest are in the search for $E(n)$. It easily follows that the *total* number of calls is bounded by $2n + E(n)$, which we bound by $2 \cdot 2^{f(n)}$ since $E(n)$ is clearly bounded by $d' < 2^{f(n)}$, and $2n \leq 2^{f(n)}$ which is easily proved by induction. $\square$

7.6 From Egyptian fraction expansions to left best approximations

The next lemma indicates how we can subrecursively convert the representation by Egyptian fraction expansion into a Dedekind cut. Here we shall use the Egyptian fraction expansion of a rational number. A rational number may have two expansions, one finite and one infinite (e.g., $1/9$ is also $1/10 + 1/100 + 1/1000 + \dots$). The algorithm we give uses the finite one. In the proof of the next lemma, we write the expansion as an infinite sequence anyway, for uniformity of notation; assume the missing fractions to be zero, identified with $1/\infty = 0$ (e.g., $1/9 + 0 + 0 + \dots$).

Lemma 7.16. *Let x, y be two different real numbers (possibly rational) in the interval $(0, 1)$, and let E^x, E^y be their respective Egyptian fraction expansions. Then, $x < y$ if and only if E^y precedes E^x in lexicographic order.*

Proof. Suppose that $E^x(i) = E^y(i)$ for all $i < n$, and n is the position of the first difference. Then regarding the nth element we have the following cases: either one of the sequences terminates, in which case it is clear that the number expressed by the other sequence is larger; or both continue. In the latter case, assume w.l.o.g. that $E^y(n) < E^x(n)$. Denote $d = \prod_{j=1}^{n-1} E^x(j) = \prod_{j=1}^{n-1} E^y(j)$.

56

Then

$$
\begin{aligned}
y - x &= \sum_{i=n}^{\infty} \left(\prod_{j=1}^{i} E^y(j) \right)^{-1} - \sum_{i=n}^{\infty} \left(\prod_{j=1}^{i} E^x(j) \right)^{-1} \\
&\geq \frac{1}{dE^y(n)} - \sum_{i=n}^{\infty} \left(\prod_{j=1}^{i} E^x(j) \right)^{-1} \\
&= \frac{1}{dE^y(n)} - \frac{1}{d} \sum_{i=n}^{\infty} \left(\prod_{j=n}^{i} E^x(j) \right)^{-1} \\
&\geq \frac{1}{dE^y(n)} - \frac{1}{d} \sum_{i=n}^{\infty} (E^x(n))^{n-i-1} = \frac{1}{dE^y(n)} - \frac{1}{d} \cdot \frac{1}{E^x(n) - 1} \geq 0 \; .
\end{aligned}
$$

We conclude that $x \leq y$, but they are known to differ, so $x < y$. $\qquad\square$

We obtain the following algorithm for computing $D^\alpha(q)$ in E^α: Compute the finite Egyptian fraction expansion of q and compare it lexicographically to E^α. For computing the expansion of q we use the algorithm of [14], which we review below. Importantly, from the algorithm it is easy to see that the expansion $E^{p/q}$ of a rational number p/q includes at most p terms and each term $E^{p/q}(i)$ is bounded by q.

Expansion algorithm: We define an auxiliary function $Div(q, p)$ that for positive integers q, p returns a pair of non-negative integers $n \leq q$ and $r < p$ such that np is the smallest multiple of p with $np \geq q$, and $r = np - q$. Given a fraction p/q, we construct sequences $n_1, n_2, \ldots$ and $r_0, r_1, \ldots$ by setting $r_0 = p$ and $(n_{i+1}, r_{i+1}) = Div(q, r_i)$ until we reach $r_j = 0$. The Egyptian fraction expansion is given by $E^{p/q}(i) = n_i$ for $i = 1, \ldots, j$.

Lemma 7.17. *There is a parameterized function-oracle Turing machine M with the following properties. Let $E : \mathbb{N}^+ \longrightarrow \mathbb{N}$ be the Egyptian fraction expansion of an irrational number $\alpha \in (0, 1)$. Then*

- $\Phi_M^E : \mathbb{Q} \longrightarrow \{0, 1\}$ *is the Dedekind cut of α*

- M^E *on input m/n runs in time $m \cdot \mathrm{poly}(\|n\|, \|E(m)\|)$ and uses at most m oracle calls, each of input size at most $1 + \log m$.*

Proof. The computation of the expansion of m/n, according to [14], makes at most m iterations, and in each iteration $O(1)$ arithmetic operations are performed on numbers bounded by n. For our purpose, we compare the ith number in the expansion, $E^{m/n}(i)$, with $E(i)$. Recalling that E is a non-decreasing function, the complexity bounds stated follow. $\qquad\square$

Let E^α be the Egyptian fraction expansion of α. We will give an algorithm for computing a complete left best approximation L^α of α, using E^α as an oracle. The algorithm works recursively. The base case is the trivial approximation $0/1$. In the general case, for $n \geq 1$, the algorithm first computes $a_{n-1}/b_{n-1} = L^\alpha(n-1)$. Then the algorithm computes $L^\alpha(n)$ as follows:

- Step 1: Compute m terms of the Egyptian Fraction expansion for the least m such that
$$\sum_{i=1}^{m} \left(\prod_{j=1}^{i} E^\alpha(j) \right)^{-1} > \frac{a_{n-1}}{b_{n-1}}$$
  and let $d' = \prod_{i=1}^{m} E^\alpha(i)$.

- Step 2: Using an implementation of the Dedekind cut D^α as described above, search for the least $d > b_{n-1}$ such that for some $c < d$, the fraction $c/d < \alpha$ is closer to α than a_{n-1}/b_{n-1}. Choose the largest such c. Return c/d as $L^\alpha(n)$. Note that d will be at most d'.

Step 1 may seem like an unbounded search. But in fact, since the Egyptian fraction expansion of a_{n-1}/b_{n-1} comprises at most a_{n-1} terms, we know, by Lemma 7.16, that $m \leq a_{n-1} + 1$.

Lemma 7.18. *There is a parameterized function-oracle Turing machine M with the following properties. Let $E : \mathbb{N}^+ \longrightarrow \mathbb{N}$ be the Egyptian fraction expansion of an irrational number $\alpha \in (0,1)$. Assume E is computable within a time bound s, and let $f(n) = (\lambda x.2^{x \log x + s(\|x\|)})^{(n)}(1)$. Then*

- $\Phi_M^E : \mathbb{N} \longrightarrow \mathbb{Q}$ *is the complete left best approximation of α*

- M^E *on input n runs in time $\mathrm{poly}(f(n))$ and uses at most $f(n)$ oracle calls, each of input size at most $\|f(n)\|$.*

Proof. We will see that the crucial quantity in the analysis of this algorithm is b_n. We claim that $b_n \leq f(n)$, which follows by induction from
$$b_n \leq 2^{b_{n-1} \log b_{n-1} + s(\|b_{n-1}\|)} .$$

To justify this, note that
$$b_n \leq d' \leq \prod_{i=1}^{m} E(i) . \tag{7.9}$$

Now $m \leq a_{n-1} + 1 \leq b_{n-1}$, as argued before the lemma. Moreover, the first $m-1$ terms $E(1), \ldots, E(m-1)$ are a prefix of the Egyptian fraction expansion of a_{n-1}/b_{n-1}, which implies that their value is at most b_{n-1}, hence their product at most $b_{n-1}^{b_{n-1}}$. To bound $E(m)$, we recall that $E(m)$ is computable in time

58

$s(\|m\|)$, and therefore bounded by $2^{s(\|m\|)} \leq 2^{s(\|b_{n-1}\|)}$. We deduce from (7.9) that

$$b_n \leq 2^{b_{n-1} \log b_{n-1} + s(\|b_{n-1}\|)} \ .$$

This completes the induction.

Now, to bound the oracle input size we claim that b_n bounds the largest value passed to the oracle throughout the algorithm; this is easy enough to verify. The bound on the number of oracle calls also follows, since the Turing machine can record answers from the oracle, and hence, we can avoid querying the same input twice.

To bound the execution time, we bound the time for the last iteration; the total time is polynomial in this quantity, since the bound on the last iteration is the largest. In the last iteration we compute terms of the Egyptian fraction expansion of a_{n-1}/b_{n-1} and compare it lexicographically with the sequence $\{E(i)\}_{i \in \mathbb{N}^+}$ to determine m; this takes $O(m) = O(b_{n-1})$ arithmetic operations on numbers bounded by b_n. We have proved that $b_n \leq f(n)$, and the time for the computation is polynomial in $\|b_n\|$ times $O(b_{n-1})$. Thus the time required by Step 1 will be $\mathrm{poly}(f(n-1) \log f(n)) = \mathrm{poly}(f(n))$.

Step 2 is implemented as follows: for $d = b_{n-1} + 1, b_{n-1} + 2, \ldots$ we compute $c = 1 + \lfloor (a_{n-1}d)/b_{n-1} \rfloor$, as this gives the smallest nominator such that $c/d > a_{n-1}/b_{n-1}$. We test if $c/d < \alpha$ using the Dedekind cut. The whole process is $\mathrm{poly}(f(n))$ since $d \leq f(n)$. When we find the first such c, d, we search for the largest c such that c/d is still below α. This involves less than d applications of the Dedekind cut; again we remain within $\mathrm{poly}(f(n))$ time. $\qquad\square$

7.7 Summary

We will now give a summary of this section along the lines we have given summaries of Section 4 (page 26) and Section 6 (page 43).

Theorem 7.19. *Consider the representations by (1) right best approximations, (2) complete right best approximations (3) standard Baire sequences and (4) general sum approximations from above, and let R_1 and R_2 be any two of these four representations. Then, for an arbitrary time-bound t, there exists a time-bound s primitive recursive in t such that $O(t)_{R_2} \subseteq O(s)_{R_1}$.*

A comment meant for the readers familiar with the Grzegorczyk hierarchy: Our results are a bit stronger than what the theorem above asserts. One can easily check that the time-bound s, for any $i \geq 3$, will be in the Grzegorczyk class $\mathcal{E}_{i+1}$ if the time-bound t is in the Grzegorczyk class $\mathcal{E}_i$. The same goes for the next theorem if we leave out the representation by Egyptian fraction expansions.

Theorem 7.20. *Consider the representations by (1) left best approximations, (2) complete left best approximations (3) dual Baire sequences, (4) general sum*

59

approximations from below and (5) Egyptian fraction expansions, and let R_1 and R_2 be any two of these five representations. Then, for an arbitrary time-bound t, there exists a time-bound s primitive recursive in t such that $O(t)_{R_2} \subseteq O(s)_{R_1}$.

8   A Little Bit on the Degrees of Representations

Recall the definition of the relation $\preceq_S$ (Definition 1.4), and recall the definition of the relations $\equiv_S$ and $\prec_S$ (Definition 1.5). Furthermore, recall that the equivalence relation $\equiv_S$ induces a degree structure on the representations, where a degree is simply an $\equiv_S$-equivalence class, see Page 11. We will use the standard terminology of degree theory and say that a degree $\mathbf{a}$ lies *(strictly) below* a degree $\mathbf{b}$ if we have $(R_1 \prec_S R_2)$ $R_1 \preceq_S R_2$ whenever $R_1 \in \mathbf{a}$ and $R_2 \in \mathbf{b}$. We say that $\mathbf{b}$ lies *(strictly) above* $\mathbf{a}$ if $\mathbf{a}$ lies (strictly) below $\mathbf{b}$. Beware that Figure 1 shows an upside-down picture of the world, that is, if a degree $\mathbf{a}$ lies below a degree $\mathbf{b}$, then $\mathbf{a}$ is depicted above $\mathbf{b}$ in the figure.

Figure 1 shows that the degree of the representation by left best approximations is incomparable to, that is, lies neither above nor below, the degree of the representation by right best approximations. That this indeed is the case can be established by a growth argument (see page 8). Let us see how such an argument works.

In Section 7 we saw that the representation by dual Baire sequences is subrecursively equivalent to the representation by left best approximations and that the representation by standard Baire sequences is subrecursively equivalent to the representation by right best approximations. In order to make our growth argument transparent, we will consider dual and standard Baire sequences in place of left and right best approximations. Let $\mathcal{B}$ and $d\mathcal{B}$ denote the representations by standard and dual Baire sequences, respectively. We will argue that $\mathcal{B} \not\preceq_S d\mathcal{B}$ and $d\mathcal{B} \not\preceq_S \mathcal{B}$.

Let s be any time bound. We will prove that there exists an irrational α in the interval $(0, 1)$ such that

$$\text{the standard Baire sequence of } \alpha \text{ is not computable in time } O(s) \qquad (8.1)$$

but still

$$\text{the dual Baire sequence of } \alpha \text{ is computable in time } \mathrm{poly}(2^{2^n}) \qquad (8.2)$$

where n is the length of the input.

Consider a very fast increasing function $B : \mathbb{N} \longrightarrow \mathbb{N}$ with a simple graph. Specifically, we assume that the graph of B, that is the relation $B(x) = y$, is decidable in time $\mathrm{poly}(\max(x, y))$, but still, B increases too fast to be computable in time $O(s)$. Such a B will always exist, and for convenience, we will also choose B so that $2^x \leq B(x)$ and $B(x) \leq B(x+1)$. Now, B is the standard Baire sequence of some irrational number α, and since an irrational number has exactly one standard Baire sequence, the standard Baire sequence of α is not

computable in time $O(s)$. Thus, (8.1) holds. It remains to prove that (8.2) holds.

Let $a_n = B(0) + \sum_{i=1}^{n}(B(i) + 1)$. Let $A(x) = 1$ if $x = a_n$ for some n; otherwise, let $A(x) = 0$. This defines a function A. We will prove that A is the dual Baire sequence of α, but first we will argue that $A(x)$ is computable in time $\text{poly}(2^{\|x\|})$: Observe that n will be smaller than x whenever $x = a_n$ holds. Thus, we can check if there exists n such that $x = a_n$ by checking

$$(\exists n, y_0, \ldots, y_n < x)[\ B(0) = y_0 \ \wedge \ \ldots \ \wedge \ B(n) = y_n$$
$$\wedge \ x = y_0 + \sum_{i=1}^{n}(y_i + 1)\ ]\ . \quad (8.3)$$

By assumption we can decide in time $\text{poly}(\max(z_1, z_2))$ if the relation $B(z_1) = z_2$ holds. Thus we can check in time $\text{poly}(x)$ if the matrix of (8.3) holds for given $y_0, \ldots, y_n$ and a search over all such sequences can be done with at most 2^x such checks (this bounds the number of ordered partitions of x). This shows that $A(x)$ is computable in time $2^x \text{poly}(x)$, and thus in time $\text{poly}(2^{2^{\|x\|}})$, since $A(x)$ equals 1, if (8.3) holds, and 0 if (8.3) does not hold.

For any natural number n, we define the strings σ_n and τ_n by

$$\sigma_n = 0^{A(0)}10^{A(1)}1\ldots0^{A(a_n-1)}10^{A(a_n)} \text{ and } \tau_n = 1^{B(0)}01^{B(1)}0\ldots1^{B(n)}0\ .$$

We prove by induction on n that $\sigma_n = \tau_n$. Let $n = 0$. We have $a_0 = B(0)$, and thus, by the definition of A, we have

$$\sigma_0\ =\ 0^{A(0)}10^{A(1)}1\ldots0^{A(a_0-1)}10^{A(a_0)}\ =\ 1^{a_0}0\ =\ 1^{B(0)}0\ =\ \tau_0\ .$$

Let $n > 0$. By the definition of a_n, we have $a_n = a_{n-1} + B(n) + 1$, and thus $B(n) = a_n - (a_{n-1} + 1)$. Furthermore, we have

$$\sigma_n\ \overset{(1)}{=}\ \sigma_{n-1}10^{A(a_{n-1}+1)}10^{A(a_{n-1}+2)}\ldots10^{A(a_n-1)}10^{A(a_n)}\ \overset{(2)}{=}$$
$$\sigma_{n-1}1^{a_n-(a_{n-1}+1)}0\ \overset{(3)}{=}\ \sigma_{n-1}1^{B(n)}0\ \overset{(4)}{=}\ \tau_{n-1}1^{B(n)}0\ \overset{(5)}{=}\ \tau_n$$

where (1) holds by the definition of σ_n; (2) holds by the definition of A; (3) holds by the definition of a_n; (4) holds by the induction hypothesis; and (5) holds by the definition of τ_n. This proves that $\sigma_n = \tau_n$ for any n, and by Lemma 7.12, we have

$$\mathcal{T}_{\text{F}}(\sigma_n)\ =\ \mathcal{T}_{\text{F}}(\tau_n)\ =\ I_B^n$$

and also

$$J_A^{a_n}\ \supseteq\ \mathcal{T}_{\text{F}}(\sigma_n) \supseteq J_A^{a_n+1}$$

for any n. By the definition of standard and dual Baire sequences (Definition 7.7), it follows that A is the dual Baire sequence of α. This completes our proof of (8.2).

61

It follows from (8.1) and (8.2) that we have

$$\mathrm{poly}(2^{2^n})_{d\mathcal{B}} \;\not\subseteq\; O(s)_{\mathcal{B}}$$

for any time bound s, and thus we have $\mathcal{B} \npreceq_S d\mathcal{B}$ by our definition of the ordering relation $\preceq_S$ (Definition 1.4). A symmetric proof yields $d\mathcal{B} \npreceq_S \mathcal{B}$.

Our growth argument shows that the degree of the representation by left best approximations (which is also the degree of $d\mathcal{B}$) is incomparable to the degree of the representation by right best approximations (which is also the degree of $\mathcal{B}$). We have seen that we can subrecursively compute the Dedekind cut of $\alpha \in (0,1)$ if we have access to a left, or to a right, best approximation of α (Lemma 7.5). Thus our two degrees of representations by best approximations will both lie above the degree of the representation by Dedekind cuts, and since the two degrees are incomparable, they have to lie *strictly* above. We cannot subrecursively convert the representation by Dedekind cuts into the representation by left, or the representation by right, best approximations. In the next section we will see that both degrees lie below the degree of the representation by continued fractions, and since the two degrees are incomparable, we can conclude that they lie strictly below. We cannot subrecursively convert the representation by left, or the representation by right, best approximations into the representation by continued fractions (but we will see in the next section that we indeed can subrecursively compute the continued fraction of α if we have access to *both a left and a right* best approximation of α). See Figure 1.

9 Representations equivalent to continued fractions

9.1 Continued fractions

We may assume some familiarity with continued fractions, but we will state and explain some of their properties below. For more on continued fractions see Khintchine [24] or Richards [37]. The latter is a very readable paper which carefully explains the relationship between continued fractions and Farey pairs.

Let $a_0, a_1, a_2, \ldots$ be an infinite sequence of integers where $a_1, a_2, a_3, \ldots$ are positive. The *continued fraction* $[a_0; a_1, a_2, \ldots]$ is defined by

$$[\,a_0; a_1, a_2, a_3, \ldots\,] \;=\; a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ldots}}}$$

We will work with continued fraction representations of irrational numbers between 0 and 1. Every irrational number α in the interval $(0,1)$ can be written uniquely in the form $\alpha = [0; a_1, a_2, \ldots]$ where $a_1, a_2, a_3, \ldots$ are positive integers. Hence, the next definition makes sense.

62

Definition 9.1. Let α be an irrational number in the interval $(0,1)$, and let $\alpha = [0; a_1, a_2, \ldots]$. The *continued fraction* of α is the function $C : \mathbb{N}^+ \longrightarrow \mathbb{N}$ given by $C(i) = a_i$. □

We define $[a_0; a_1, \ldots, a_n]$ by induction on n. If $n = 0$, let $[a_0; \ ] = a_0$. If $n > 0$, let

$$[\, a_0; a_1, \ldots, a_n \,] \;=\; a_0 \;+\; \cfrac{1}{[\, a_1; a_2, \ldots, a_n \,]} \,.$$

The rational number $[a_0; a_1, \ldots, a_n]$ is known as the *nth convergent* of the infinite continued fraction $[a_0; a_1, a_2, \ldots]$.

Let

$$\frac{p_0}{q_0} = \frac{0}{1} \,, \quad \frac{p_1}{q_1} = \frac{1}{a_1} \quad \text{and} \quad \frac{p_{k+2}}{q_{k+2}} = \frac{p_k + a_{k+2} p_{k+1}}{q_k + a_{k+2} q_{k+1}} \,. \tag{9.1}$$

It is well known that p_n/q_n equals the nth convergent of $[0; a_1, a_2, a_3, \ldots]$, that is, $p_n/q_n = [0; a_1, \ldots, a_n]$. It is also well known that

$$\frac{p_0}{q_0} \;<\; \frac{p_2}{q_2} \;<\; \frac{p_4}{q_4} \;<\; \ldots \;<\; \alpha \;< \ldots \;<\; \frac{p_5}{q_5} \;<\; \frac{p_3}{q_3} \;<\; \frac{p_1}{q_1}$$

where $\alpha = [0; a_1, a_2, \ldots]$, that is, the even convergents approach the number represented by the continued fraction from below whereas the odd convergents approach it from above. Every convergent is a (left or right) best approximant, but the converse it not true, a best approximant will not necessarily be a convergent. The next theorem relates the convergents of a continued fraction and the Farey pair tree $\mathcal{T}_F$.

Theorem 9.2 (Hurwitz [23])**.** *Let $[0; a_1, a_2, \ldots]$ be a continued fraction, and let the convergent p_n/q_n be defined by (9.1) above. We have*

$$\mathcal{T}_F(\, 0^{a_1-1} 1^{a_2} \ldots 1^{a_{n-1}} 0^{a_n} \,) \;=\; \left(\frac{p_{n-1}}{q_{n-1}}, \frac{p_n}{q_n} \right) = ([0; a_1, \ldots, a_{n-1}], [0; a_1, \ldots, a_n])$$

when n is odd, and we have

$$\mathcal{T}_F(\, 0^{a_1-1} 1^{a_2} \ldots 0^{a_{n-1}} 1^{a_n} \,) \;=\; \left(\frac{p_n}{q_n}, \frac{p_{n-1}}{q_{n-1}} \right) = ([0; a_1, \ldots, a_n], [0; a_1, \ldots, a_{n-1}])$$

when n is even.

9.2 Definitions

We will now define some representations which turn out to be subrecursively equivalent to the representation by continued fractions.

Definition 9.3. A function $T : [0,1] \cap \mathbb{Q} \longrightarrow \mathbb{Q}$ is a *trace function* for the irrational number $\alpha \in (0,1)$ if we have $|\alpha - r| > |\alpha - T(r)|$ for every $r \in [0,1] \cap \mathbb{Q}$. □

We will say that a trace function T *moves* r *to the right (left)* if $r < T(r)$ ($T(r) < r$). The easiest way to realize that a trace function indeed defines a unique real number, is probably to observe that a trace function T for α yields the Dedekind cut of α: if T moves r to the right, then we know that r lies below α; if T moves r to the left, then we know that r lies above α. Obviously, T cannot yield the Dedekind cut for any number other than α.

Intuitively, a *contractor* is a function that moves two (rational) numbers closer to each other. We will see that contractors can be used to represent irrational numbers.

Definition 9.4. A function $F : [0, 1] \cap \mathbb{Q} \longrightarrow (0, 1) \cap \mathbb{Q}$ is a *contractor* if we have $F(r) \neq r$ and $|F(r_1) - F(r_2)| < |r_1 - r_2|$ for any rationals r, r_1, r_2 where $r_1 \neq r_2$. □

Lemma 9.5. *Any contractor is a trace function for some irrational number.*

Proof. Let F be a contractor. If F moves r to the right (left), then F also moves any rational less (greater) than r to the right (left); otherwise F would not be a contractor. We define two sequences $r_0, r_1, r_2, \ldots$ and $s_0, s_1, s_2, \ldots$ of rationals. Let $r_0 = 0$ and $s_0 = 1$. Let $r_{i+1} = (r_i + s_i)/2$ if F moves $(r_i + s_i)/2$ to the right; otherwise, let $r_{i+1} = r_i$. Let $s_{i+1} = (r_i + s_i)/2$ if F moves $(r_i + s_i)/2$ to the left; otherwise, let $s_{i+1} = s_i$ (Definition 9.4 requires that a contractor moves any rational number). Obviously, we have $\lim_i r_i = \lim_i s_i$, and obviously, this limit is an irrational number α. It is easy to see that F is a trace function for α. □

The previous lemma shows that the next definition makes sense.

Definition 9.6. A contractor F is a *contractor for the irrational number* α if F is a trace function for α. □

The representation by trace functions was introduced in Kristiansen [31], and the representation by contractors was introduced in Kristiansen [33]. Lehman [35] has studied the representation by continued fractions. He proved that the class real numbers with primitive recursive continued fractions is strictly included in the class of real numbers with primitive recursive Dedekind cuts. Ko [25] [26] has studied conversions between continued fractions and other representations from a complexity-theoretic point of view.

9.3 From complete best approximation to continued fractions

By Theorem 9.2, the continued fraction $[0; x_1, x_2, \ldots]$ of $\alpha \in (0, 1)$ can be viewed as the infinite path $0^{x_1 - 1} 1^{x_2} 0^{x_3} 1^{x_4} \ldots$ in the Farey pair tree $\mathcal{T}_F$. By Lemma 7.3, we can construct the path $0^{x_1 - 1} 1^{x_2} 0^{x_3} 1^{x_4} \ldots$ if we have access to the complete left and the complete right best approximations of α. This insight yields an algorithm for converting complete best approximations into a

64

continued fraction. The algorithm, which is given by pseudocode in Figure 4, uses the complete left best approximation of α, denoted $\{a_i/b_i\}_{i\in\mathbb{N}}$, and the complete right best approximation of α, denoted $\{c_i/d_i\}_{i\in\mathbb{N}}$, as oracles. The algorithm outputs the nth element x_n of α's continued fraction $[0; x_1, x_2, \ldots]$ (the input is n). The comments embraced by $(*\ldots*)$ explain how the algorithm works, and $m(a/b, c/d)$ denotes the mediant of the fractions a/b and c/d, that is, $m(a/b, c/d) = (a + c)/(b + d)$.

The first while-loop in Figure 4 counts consecutive zeros found in the path $0^{x_1-1}1^{x_2}0^{x_3}1^{x_4}\ldots$ given by the best approximations, whereas the second while-loop counts consecutive ones. It turns out that the actual counting is superfluous and that these two loops can be eliminated: We can directly compute the number of times a while-loop will be executed from values available when the loop starts. Let us consider the first while-loop:

$$\texttt{WHILE}\quad m\left(\frac{a_\ell}{b_\ell}, \frac{c_r}{d_r}\right) = \frac{c_{r+1}}{d_{r+1}}\quad \texttt{DO}\quad \texttt{BEGIN}\quad r := r+1;\; x_i := x_i+1\quad \texttt{END}$$

No variables except r and x_i are modified during the execution of this loop. Let

- S be the value of r when the execution of the loop starts

- T be the value of r when the execution of the loop terminates

- X be the value of x_i when the execution of the loop terminates (observe that x_i is 1 when the execution starts, and thus $X = 1 + T - S$).

For $r = S, \ldots, T-1$, we have

$$\frac{c_{r+1}}{d_{r+1}} \;=\; m\left(\frac{a_\ell}{b_\ell}, \frac{c_r}{d_r}\right) \;=\; \frac{a_\ell + c_r}{b_\ell + d_r}\;.$$

Thus, when the loop terminates, we have

$$\frac{c_T}{d_T} \;=\; \frac{a_\ell(X-1) + c_S}{b_\ell(X-1) + d_S}\;. \tag{9.2}$$

When the loop terminates, we also have

$$\frac{c_{T+1}}{d_{T+1}} \;\neq\; m\left(\frac{a_\ell}{b_\ell}, \frac{c_T}{d_T}\right)$$

but then, as the mediant $m(a_\ell/b_\ell, c_T/d_T)$ is not the next fraction in the complete list of right best approximants, it will be fraction number $\ell + 1$ in the complete list of left best approximants. That is, we have:

$$\frac{a_{\ell+1}}{b_{\ell+1}} \;=\; m\left(\frac{a_\ell}{b_\ell}, \frac{c_T}{d_T}\right) \;=\; \frac{a_\ell + c_T}{b_\ell + d_T}\;. \tag{9.3}$$

By (9.2) and (9.3), we have

$$\frac{a_{\ell+1}}{b_{\ell+1}} \;=\; \frac{a_\ell + a_\ell(X-1) + c_S}{b_\ell + b_\ell(X-1) + d_S}\;. \tag{9.4}$$

65

$\ell := 0; \ r := 0; \ i := 1; \ x_i := 1;$

(∗ we have $a_\ell/b_\ell = 0/1$ and $c_r/d_r = 1/1$ ∗)

more: (∗ α lies in $(a_\ell/b_\ell, c_r/d_r)$ ∗)

WHILE $m\left(\dfrac{a_\ell}{b_\ell}, \dfrac{c_r}{d_r}\right) = \dfrac{c_{r+1}}{d_{r+1}}$ DO

   BEGIN (∗ α lies in $(a_\ell/b_\ell, c_{r+1}/d_{r+1})$ and the path branches left ∗)

   $r := r + 1; \ x_i := x_i + 1$

   END (∗ we have determined x_i where i is odd ∗)

IF $i = n$ THEN $\langle$give output x_i and halt$\rangle$

(∗ α lies in $(a_{\ell+1}/b_{\ell+1}, c_r/d_r)$ since the loop terminated ∗)

$i := i + 1; \ x_i := 1; \ \ell := \ell + 1;$ (∗ α lies in $(a_\ell/b_\ell, c_r/d_r)$ ∗)

WHILE $m\left(\dfrac{a_\ell}{b_\ell}, \dfrac{c_r}{d_r}\right) = \dfrac{a_{\ell+1}}{b_{\ell+1}}$ DO

   BEGIN (∗ α lies in $(a_{\ell+1}/b_{\ell+1}, c_r/d_r)$ and the path branches right ∗)

   $\ell := \ell + 1; \ x_i := x_i + 1$

   END (∗ we have determined x_i where i is even ∗)

IF $i = n$ THEN $\langle$give output x_i and halt$\rangle$

(∗ α lies in $(a_\ell/b_\ell, c_{r+1}/d_{r+1})$ since the loop terminated ∗)

$i := i + 1; \ x_i := 1; \ r := r + 1;$ (∗ α lies in $(a_\ell/b_\ell, c_r/d_r)$ ∗)

GOTO   *more*

Figure 4: ALGORITHM

66

```
1                 ℓ := 0;  r := 0;  i := 1;
2      more:     x_i := (b_{ℓ+1} − d_r)/b_ℓ;  r := r + (x_i − 1);
3                 IF  i = n  THEN  ⟨give output x_i and halt⟩
4                 i := i + 1;  ℓ := ℓ + 1;
5                 x_i := (d_{r+1} − b_ℓ)/d_r;  ℓ := ℓ + (x_i − 1);
6                 IF  i = n  THEN  ⟨give output x_i and halt⟩
7                 i := i + 1;  r := r + 1;
8                 GOTO  more
```

Figure 5: ALGORITHM

Now, (9.4) yields the equation

$$X = \frac{b_{\ell+1} - d_S}{b_\ell}$$

where d_S is the value of d_r when the execution of the loop starts and X is the value of x_i when the execution of the loop terminates. Hence, the loop above can be replaced by

$$x_i := (b_{\ell+1} - d_r)/b_\ell;\ r := r + (x_i - 1)$$

(recall that x_i is 1 when the loop starts and hence r will be incremented exactly $x_i - 1$ times before the loop terminates). A symmetric argument shows that the loop

$$\texttt{WHILE}\ \ m\left(\frac{a_\ell}{b_\ell}, \frac{c_r}{d_r}\right) = \frac{a_{\ell+1}}{b_{\ell+1}}\ \ \texttt{DO}\ \ \texttt{BEGIN}\ \ \ell := \ell + 1;\ x_i := x_i + 1\ \ \texttt{END}$$

can be replaced by the program

$$x_i := (d_{r+1} - b_\ell)/d_r;\ \ell := \ell + (x_i - 1).$$

Thus, we have the algorithm in Figure 5.

Lemma 9.7. *There is a parameterized function-oracle Turing machine M with the following properties. Let $L : \mathbb{N} \longrightarrow \mathbb{Q}$ and $R : \mathbb{N} \longrightarrow \mathbb{Q}$ be the complete left and right, respectively, best approximations of an irrational $\alpha \in (0, 1)$. Assume that both L and R are computable within a time-bound s, and let $f(n) = \lambda x.(s(1 + x))^{(n)}(0)$. Then*

- *$\Phi_M^{L,R} : \mathbb{N}^+ \longrightarrow \mathbb{N}$ is the continued fraction of α*

- *$M^{L,R}$ on input n runs in time $\mathrm{poly}(f(n))$ and uses exactly $2n$ oracle calls, each of input size at most $f(n-1) + 1$.*

67

Proof. The Turing machine M will iterate the body of the loop implemented by the goto-statement in Figure 5 no more than $\lceil n/2 \rceil$ times. To implement Line 2, it asks the oracle L for the values of a_ℓ/b_ℓ and $a_{\ell+1}/b_{\ell+1}$. Then M can carry out the assignment as the value of d_r is already known to M and will be stored at the work tape (except for the very first time Line 2 is reached, then we have $r = 0$ and $d_r = 1$ by convention). Similarly, to implement Line 5, it asks the oracle R for c_r/d_r and c_{r+1}/d_{r+1}. The value of b_ℓ is already known to M. Thus, M consults each oracle twice each time the loop's body is iterated, and when M outputs x_n and halts, exactly $2n$ oracle calls have been made.

The time required to perform the arithmetic operations is polynomial in the size of the (representation of) the involved values. We do an induction to prove bounds on the bit-lengths of ℓ, r throughout the algorithm. Our induction claim is the conjunction of the following two statements: (1) whenever Line 6 is reached, we have

$$\ell \ \le \ d_{r+1} \quad \text{and} \quad \|\ell\| \ \le \ s(1 + f(i-1)) \tag{9.5}$$

and (2) whenever Line 3 is reached, we have

$$r \ \le \ b_{\ell+1} \quad \text{and} \quad \|r\| \ \le \ s(1 + f(i-1)) \, . \tag{9.6}$$

Note that these bounds, which we prove next, complete the justification of the lemma regarding the time complexity and oracle input size.

Consider the first tour through the loop body. We reach Line 3 with

$$r = (b_1 - d_0)/b_0 - 1 = (b_1 - 1)/1 - 1 = b_1 - 2 \le b_1$$

and we reach line 6 with $\ell = (d_1 - b_1)/d_0 = d_1 - b_1 \le d_1$. It is easy to verify that both (9.5) and (9.6) hold.

In general, suppose that we have reached Line 6 for the kth time. Now $i = 2k$. Moving on towards Line 2, we increment i and r. The value of x_i computed at Line 2 is bounded by

$$x_i \ = \ (b_{\ell+1} - d_r)/b_\ell \ \le \ b_{\ell+1} - b_\ell,$$

where the last inequality uses the fact that $(x-1)/y \le x-y$ holds for all integers where $x > y > 0$. We compare the new value of r after Line 2 to the value r' which it had when we last visited Line 3. We have

$$r \ = \ r' + x_i \le b_{\ell'+1} + b_{\ell+1} - b_\ell \ \le \ b_{\ell+1}$$

where ℓ' is the value ℓ had when we last visited Line 3 (clearly $\ell' \le \ell - 1$).

This proves that $r \le b_{\ell+1}$, and hence the first conjunct of (9.6) holds when Line 3 is reached. In order to verify that the second conjunct of (9.6) also holds, observe that $b_{\ell+1}$ is obtained by an oracle query, and hence we have $\|b_{\ell+1}\| \le$

68

$s(\|\ell + 1\|)$. By our induction hypothesis (9.5), we have $\|\ell\| \leq s(1 + f(i-2))$ when Line 3 is reached. Thus, as $r \leq b_{\ell+1}$, we have

$$\|r\| \;\leq\; \|b_{\ell+1}\| \;\leq\; s(\|\ell + 1\|) \;\leq\; s(1 + \|\ell\|) \;\leq$$
$$s(1 + s(1 + f(i-2))) \;=\; s(1 + f(i-1))$$

(the final equality holds by the definition of f). A symmetric argument justifies (9.5) in the inductive case. $\qquad\square$

9.4 From continued fractions to complete best approximations

Our algorithm for converting a continued fraction into a complete left (or right) best approximation is pretty straightforward: Let $[0; x_1, x_2, \cdots]$ be the continued fraction of α. By Theorem 9.2, $0^{x_1-1}1^{x_2}0^{x_3}\cdots$ is the unique path of α in the Farey pair tree $\mathcal{T}_F$. By Lemma 7.3, complete left and right best approximations of α can be read off, in order, from the Farey pairs along the path $0^{x_1-1}1^{x_2}0^{x_3}\cdots$. Every time a 1 occurs in the path (branching right), a new left best approximant will show up; every time a 0 occurs in the path (branching left), a new right best approximant will show up.

Lemma 9.8. *There is a parameterized function-oracle Turing machine M with the following properties. Let $C : \mathbb{N}^+ \longrightarrow \mathbb{N}$ be the continued fraction of the irrational number $\alpha \in (0,1)$. Then*

- *$\Phi_M^C : \mathbb{N} \longrightarrow \mathbb{Q}$ is the complete left best approximation of α*

- *M^C on input n runs in time $\mathrm{poly}(\sum_{j=1}^{2n} C(j))$ and makes at most $2n$ oracle calls, each of input size at most $2 + \log n$.*

Proof. Let $\underline{\sigma}$ denote the path $0^{C(1)-1}1^{C(2)}0^{C(3)}\cdots$. The nth element of the complete best left approximation is obtained by finding the index i_n such that the nth occurrence of 1 in $\underline{\sigma}$ occurs at index i_n. Thereafter M computes the Farey pair at position $\underline{\sigma}_1 \cdots \underline{\sigma}_{i_n}$ and simply returns that pair's left component.

Observe that as $C(2), C(4), \ldots \geq 1$, we have $i_n \leq \sum_{j=1}^{2n} C(j)$. Hence, M needs to perform at most $2n$ oracle calls of input size at most $1 + \lfloor \log 2n \rfloor \leq 2 + \log n$ (as the input needs to represent numbers of size at most $2n$). After having found i_n, M determines the Farey pair at the position $\underline{\sigma}_1 \cdots \underline{\sigma}_{i_n}$. This can be accomplished in time $\mathrm{poly}(i_n) \leq \mathrm{poly}(\sum_{j=1}^{2n} C(j))$ by Proposition 2.4. Now it is easy to see that the total time needed for computing the arguments to the oracle calls, reading off the results, and computing the relevant Farey pair is bounded above by $\mathrm{poly}(\sum_{j=1}^{2n} C(j))$. $\qquad\square$

9.5 Conversion between contractors, trace functions and complete best approximations

Lemma 9.9. *(i) Let $\{\hat{r}_i\}_{i\in\mathbb{N}}$ be a complete left best approximation of α. For any $i \in \mathbb{N}$, we have*

$$\hat{r}_{i+1} - \hat{r}_i \; > \; \hat{r}_{i+2} - \hat{r}_{i+1} \;.$$

(ii) Let $\{\check{r}_i\}_{i\in\mathbb{N}}$ be a complete right best approximation of α. For any $i \in \mathbb{N}$, we have

$$\check{r}_i - \check{r}_{i+1} \; > \; \check{r}_{i+1} - \check{r}_{i+2} \;.$$

Proof. We prove (ii); the proof of (i) is symmetric. By Lemma 7.3, $\check{r}_i$ will be the right endpoint of some interval in the tree $\mathcal{T}_{\mathrm{F}}$. We consider the last interval of this kind, that is, we let σ be the longest binary string with $\alpha \in \mathcal{T}_{\mathrm{F}}(\sigma)$ and such that $\check{r}_i$ is the right endpoint of the interval $\mathcal{T}_{\mathrm{F}}(\sigma)$. Let us write $\mathcal{T}_{\mathrm{F}}(\sigma) = (a/b, c/d)$. Then $\check{r}_i = c/d$. Furthermore, $\alpha \in \mathcal{T}_{\mathrm{F}}(\sigma 0) = (a/b, (a+c)/(b+d))$ and $\check{r}_{i+1} = (a+c)/(b+d)$.

By Theorem 2.2 we have

$$\check{r}_i - \check{r}_{i+1} = \frac{c}{d} - \frac{a+c}{b+d} = \frac{1}{d(b+d)}. \tag{9.7}$$

By Lemma 7.3 there exists m such that $\check{r}_{i+2}$ is the right endpoint of the interval $\mathcal{T}_{\mathrm{F}}(\sigma 0 1^m 0)$. We can assume that $m = 0$ since this case yields the maximal distance between $\check{r}_{i+1}$ and $\check{r}_{i+2}$. Thus, by the definition of $\mathcal{T}_{\mathrm{F}}$, we have

$$\mathcal{T}_{\mathrm{F}}(\sigma 00) = \left(\frac{a}{b}, \frac{2a+c}{2b+d} \right) \quad \text{and} \quad \check{r}_{i+2} = \frac{2a+c}{2b+d} \;.$$

Moreover, again by the definition of $\mathcal{T}_{\mathrm{F}}$, we have

$$\mathcal{T}_{\mathrm{F}}(\sigma 0) = \left(\frac{a}{b}, \frac{a+c}{b+d} \right) \quad \text{and} \quad \mathcal{T}_{\mathrm{F}}(\sigma 01) = \left(\frac{2a+c}{2b+d}, \frac{a+c}{b+d} \right) \;.$$

This shows that $((2a+c)/(2b+d), (a+c)/(b+d))$ is an interval in $\mathcal{T}_{\mathrm{F}}$, and thus, by Theorem 2.2, we have

$$\frac{a+c}{b+d} - \frac{2a+c}{2b+d} \; = \; \frac{1}{(b+d)(2b+d)} \;. \tag{9.8}$$

Now we can conclude our proof of (ii) with

$$\check{r}_i - \check{r}_{i+1} \; = \; \frac{c}{d} - \frac{a+c}{b+d} \; \overset{(9.7)}{=} \; \frac{1}{d(b+d)} \; > \; \frac{1}{(b+d)(2b+d)} \; \overset{(9.8)}{=}$$
$$\frac{a+c}{b+d} - \frac{2a+c}{2b+d} \; = \; \check{r}_{i+1} - \check{r}_{i+2} \;.$$

$\square$

It is necessary to assume in Lemma 9.9 that the best approximations are complete. The lemma does not hold for best approximations in general: E.g., let $(1000)^{-1} > \alpha > 0$. Then we have $1 > 1/2 > 1/3 > 1/99 > \ldots > \alpha$ where $1/2$, $1/3$ and $1/99$ all are right best approximants of α, but it is false that $1/2 - 1/3 > 1/3 - 1/99$. A complete right best approximation of α has the form

$$1 > \frac{1}{2} > \frac{1}{3} > \frac{1}{4} > \ldots > \frac{1}{i} > \frac{1}{i+1} > \ldots > \frac{1}{1000} > \ldots > \alpha \,.$$

Let $\{\hat{r}_i\}_{i\in\mathbb{N}}$ be a complete left best approximation of α, and let $\{\check{r}_i\}_{i\in\mathbb{N}}$ be a complete right best approximation of α. Furthermore, let

$$\hat{k}_i = (\hat{r}_{i+2} - \hat{r}_{i+1})/(\hat{r}_{i+1} - \hat{r}_i) \quad \text{and} \quad \check{k}_i = (\check{r}_{i+1} - \check{r}_{i+2})/(\check{r}_i - \check{r}_{i+1}) \,.$$

By the previous lemma we have $\hat{k}_i < 1$ and $\check{k}_i < 1$ (for all i), and thus we can define a contractor F for α by

$$F(x) = \begin{cases} \hat{r}_{i+1} + \hat{k}_i(x - \hat{r}_i) & \text{if } \hat{r}_i \leq x < \hat{r}_{i+1} \\ \check{r}_{i+1} + \check{k}_i(x - \check{r}_i) & \text{if } \check{r}_{i+1} < x \leq \check{r}_i \,. \end{cases} \tag{9.9}$$

It order to verify that F is indeed a contractor for α, we will prove that we have

$$|F(x) - F(y)| < |x - y| \tag{9.10}$$

whenever $x \neq y$. The proof splits into several cases. We can w.l.o.g. assume that $x < y$.

Case (i) There exist i, j such that $\hat{r}_i \leq x < \hat{r}_{i+1}$ and $\check{r}_{j+1} < y \leq \check{r}_j$. Then F moves x to the right and y to the left, and hence (9.10) holds.

Case (ii) There exists i such that $\hat{r}_i \leq x < y < \hat{r}_{i+1}$. Then we have $|F(x) - F(y)| = \hat{k}_i|x - y|$, and (9.10) holds since $\hat{k}_i < 1$.

Case (iii) There exist i, j (where $i < j$) such that $\hat{r}_i \leq x < \hat{r}_{i+1}$ and $\hat{r}_j \leq y < \hat{r}_{j+1}$. Now, we form the telescopic sum

$$F(y) - F(x) = F(y) - F(\hat{r}_j) + F(\hat{r}_j) - \ldots - F(\hat{r}_{i+1}) + F(\hat{r}_{i+1}) - F(x)$$

Now, since $\hat{r}_j \leq y < \hat{r}_{j+1}$, we have

$$F(y) - F(\hat{r}_j) = \hat{r}_{j+1} + \hat{k}_j(y - \hat{r}_j) - \hat{r}_{j+1} < y - \hat{r}_j$$

and similarly for the following differences. So we have

$$\begin{aligned} F(y) - F(x) &< (y - \hat{r}_j) + (\hat{r}_j - \hat{r}_{j-1}) + \ldots + (\hat{r}_{i+1} - x) \\ &= y + (-\hat{r}_j + \hat{r}_j - \hat{r}_{j-1} + \ldots + \hat{r}_{i+1}) - x \\ &= y - x. \end{aligned}$$

71

Case (iv) There exists i such that $\check{r}_{i+1} < x < y \le \check{r}_i$. This case is symmetric to (ii). Use that $\check{k}_i < 1$.

Case (v) There exist i, j (where $i < j$) such that $\check{r}_{i+1} < x \le \check{r}_i$ and $\check{r}_{j+1} < y \le \check{r}_j$. This case is symmetric to (iii).

This completes the proof of (9.10).

Lemma 9.10. *There is a parameterized function-oracle Turing machine M with the following properties. Let $L : \mathbb{N} \longrightarrow \mathbb{Q}$ and $R : \mathbb{N} \longrightarrow \mathbb{Q}$ be the complete left and right, respectively, best approximations of an irrational $\alpha \in (0, 1)$. Let $L(i) = a_i/b_i$ and $R(i) = c_i/d_i$. Then*

- $\Phi_M^{L,R} : [0, 1] \cap \mathbb{Q} \longrightarrow (0, 1) \cap \mathbb{Q}$ *is a contractor for α.*

- $M^{L,R}$ *on input n/m runs in time $O(\log m \log^2 \max\{b_{m+1}, d_{m+1}\})$ and uses at most $2 + \log m$ oracle calls, each of input size at most $\|m + 1\|$.*

Proof. Define M to be the Turing machine that, when given oracle access to L, R computes the contractor F given by (9.9). On input $n/m \in \mathbb{Q}$, the fact that L and R yield best approximations implies that there is an $i < m$ such that

$$L(i) \;=\; \frac{a_i}{b_i} \;\le\; \frac{n}{m} \;<\; \frac{a_{i+1}}{b_{i+1}} \;=\; L(i+1)$$
$$\text{or} \quad R(i+1) \;=\; \frac{c_{i+1}}{d_{i+1}} \;<\; \frac{n}{m} \;\le\; \frac{c_i}{d_i} \;=\; R(i) \,.$$

The machine first queries $L(m)$ which will be above n/m if and only if $n/m < \alpha$. Then we know wether to continue the search with L or with R. As L and R are strictly increasing, resp. decreasing, M can find i as above by binary search, requiring at most $1 + \log m$ steps, and each step requires 4 comparisons of rational numbers that can be performed in time $O(m_0^2)$ where m_0 is the length of the largest binary representation of the integer components of the rational numbers (because $n/m < a/b$ iff $nb < am$, and schoolbook multiplication can be done in quadratic time in the size of the representation). The largest integer occurring in the comparison above is bounded above by

$$\max\{n, m, a_{i+1}, b_{i+1}, c_{i+1}, d_{i+1}\}_{i=1}^m \le \max\{b_{m+1}, d_{m+1}\}$$

where the inequality follows as all fractions are bounded above by 1 and $\{a_i/b_i\}_{i\in\mathbb{N}}$ and $\{c_i/d_i\}_{i\in\mathbb{N}}$ are best approximations. Hence, the total time needed to compute i is $O(\log m \log^2 \max\{b_{m+1}, d_{m+1}\})$ using $1 + \log m$ oracle calls, each of size at most $\|m+1\|$. Once i has been found, M first computes $F(n/m)$ as a fraction (not necessarily in lowest terms) using one additional oracle query (necessary for computing the slope $\hat{k}_i$ or $\check{k}_i$), and a constant number of additions, subtractions

and multiplications of numbers representable by at most $\log \max\{b_{m+1}, d_{m+1}\}$ bits, hence using at most $O(\log^2 \max\{b_{m+1}, d_{m+1}\})$ operations and resulting in a number representable using $O(\log \max\{b_{m+1}, d_{m+1}\})$ bits. Obtaining a reduced fraction can be done by first computing the gcd of the numerator and the denominator, and then performing the 2 requisite integer divisions, for a total of $O(\log^2 \max\{b_{m+1}, d_{m+1}\})$ further operations. $\qquad\square$

Before we give our algorithm for converting trace functions (and thus also contractors) into best approximations, we will make a couple of observations. The first observation is trivial: If a trace function for α moves a rational number to the right (left), then the rational number lies below (above) α. Hence, if we have access to a trace function for α, we can easily compute the Dedekind cut D^α of α. The next observation is slightly more sophisticated: Let $T_0(r) = (r + T(r))/2$ where T is a trace function for α. Then, T_0 will also be a trace function for α, moreover, we have

$$T_0(r) < \alpha \text{ if } r < \alpha \qquad (9.11)$$

and

$$T_0(r) > \alpha \text{ if } r > \alpha. \qquad (9.12)$$

In order to see that (9.11) holds, assume that $r < \alpha$ and $T_0(r) = (r + T(r))/2 \geq \alpha$. Then, we have $T(r) - \alpha \geq \alpha - r > 0$, and thus $|T(r) - \alpha| = T(r) - \alpha \geq \alpha - r = |\alpha - r|$, contradicting that T is a trace function. A symmetric argument shows that (9.12) holds.

Our algorithm for converting a trace function T for α into the complete left best approximation $\{a_i/b_i\}_{i \in \mathbb{N}}$ of α uses the Dedekind cut D^α and the trace function T_0. When $n = 0$, the algorithm simply lets $a_n/b_n = 0/1$. When $n > 0$, the algorithm performs the following steps.

- Step 1: Recursively, compute the value a'/b' of $T_0(a_{n-1}/b_{n-1})$.

- Step 2: Using D^α, search for the least natural number $b'' \leq b'$ such that for some a'', $a_{n-1}/b_{n-1} < a''/b'' < \alpha$.

- Step 3: Using D^α, find the greatest $a'' < b''$ such that $a''/b'' < \alpha$.

- Step 4: Set a_n/b_n to a''/b''.

Such a'' and b'' will for sure exist as, if no $b'' < b'$ satisfies the requirement in Step 2, then b' itself satisfies it. It is easy to see that a''/b'' will be the smallest left best approximant of α that is strictly greater than a_{n-1}/b_{n-1}.

A trace function can of course be converted into a right best approximation by a symmetric algorithm.

Lemma 9.11. *There is a parameterized function-oracle Turing machine M with the following properties. Let $T : [0,1] \cap \mathbb{Q} \longrightarrow \mathbb{Q}$ be a trace function for an irrational number $\alpha \in (0,1)$. Assume $T(x)$ is computable in time $s(\|x\|)$, and let $f(n) = \lambda x.(2s(x))^{(n)}(2)$. Then*

- $\Phi_M^T : \mathbb{N} \longrightarrow \mathbb{Q}$ *is the complete left best approximation of α*

- M^T *on input n runs in time $O(2^{f(n)}) + \mathrm{poly}(s(f(n)))$ and uses $O(2^{f(n)})$ oracle calls, each of input size at most $f(n)$.*

Proof. Observe that computing $T_0(r)$ can be performed with a single oracle call to $T(r)$ followed by three arithmetical operations, hence in time polynomial in the size of the representations of r and $T(r)$; the latter is bounded by $s(\|r\|)$. We first bound the size of b': by assumption, a'/b' is computable in time $s(\|a_{n-1}/b_{n-1}\|)$, so this is also a bound on its size, and the size of b'' is by definition at most that of b'; the same goes for a''. A straightforward induction on n will show that $\|a_n/b_n\| \leq f(n)$.

The search for b'' in Step 2 requires at most b' calls to T (in order to compute the Dedekind cut of α), each with arguments consisting of a rational with representation size at most $2\|b'\|$. We use a rough bound on b', namely $2^{\|a'/b'\|}$, to estimate the number of calls. Step 3 has the same bound.

Regarding execution time: arithmetic operations, tests etc. add an overhead polynomial in the number of oracle calls, and we should also take into account that the result of the call may be bigger than its argument and another application of s is necessary to cover this cost. Hence we arrive at the expression $O(2^{s(f(n))} + s(f(n)))$. This leads to the conclusions stated in the lemma. $\square$

The brute-force search in the proof of Lemma 9.11 can also be performed by using the Stern-Brocot tree, but we have been unable to derive better bounds for this approach.

9.6 Summary

We can now give a summary of our results on the complexity of the conversions presented in this section in the same style as we have done in earlier sections.

Theorem 9.12. *Consider the representations by (1) continued fractions, (2) complete left best approximations together with complete right best approximations, (3) trace functions and (4) contractors, and let R_1 and R_2 be any two of these four representations. Then, for an arbitrary time-bound t, there exists a time-bound s primitive recursive in t such that $O(t)_{R_2} \subseteq O(s)_{R_1}$.*

The reader familiar with the Grzegorczyk hierarchy can easily check that the time-bound s in Theorem 9.12, for any $i \geq 3$, will be in the Grzegorczyk class $\mathcal{E}_{i+2}$ if the time-bound t is in the Grzegorczyk class $\mathcal{E}_i$.

References

[1] Sanjeev Arora and Boaz Barak, *Computational complexity: A modern approach*, Cambridge University Press, 2009.

[2] Unknown author, *Rhind mathematical papyrus*, approximately 1500 BC.

[3] A. Avizienis, *Signed-digit numbe representations for fast parallel arithmetic*, IRE Transactions on Electronic Computers (1961), no. 10, 389–400.

[4] B. Bates, M. Bunder, and K. Tognetti, *Locating terms in the Stern-Brocot tree*, European Journal of Combinatorics **31** (2010), no. 3, 1020 – 1033.

[5] A. Bauer, M. Hötzel, and A. Simpson, *Comparing functional paradigms for exact real-number computation*, ICALP 2002: Automata, Languages and Programming, LNCS, vol. 2380, Springer-Verlag, 2002, pp. 488–500.

[6] Anatole Beck, Michael N Bleicher, and Donald W Crowe, *Excursions into mathematics: The millennium edition*, ch. 7. Egyptian Fractions, pp. 421–434, CRC Press, 2000.

[7] Amir Ben-Amram and Lars Kristiansen, *A degree structure on representations of irrational numbers*, Accepted for publication in Journal of Logic and Analysis (2024).

[8] Ulrich Berger, Kenji Miyamoto, Helmut Schwichtenberg, and Hideki Tsuiki, *Logic for Gray-code computation*, Concepts of Proof in Mathematics, Philosophy, and Computer Science, Ontos Mathematical Logic, vol. 6, de-Gruyter, 2016.

[9] Joseph Bertrand, *Traité d'arithmétique*, 1849.

[10] Hans-Juergen Boehm, Robert Cartwright, Mark Riggle, and Michael J. O'Donnell, *Exact real arithmetic: A case study in higher order programming*, Proceedings of the 1986 ACM Conference on LISP and Functional Programming (William L. Scherlis, John H. Williams, and Richard P. Gabriel, eds.), ACM, 1986, pp. 162–173.

[11] B. Bolzano, *Pure theory of numbers*, Oxford University Press, 2004, In the *Mathematical Works of Bernard Bolzano* edited and translated by Steve Russ, pp. 355-428.

[12] V. Brattka, P. Hertling, and K. Weihrauch, *A tutorial on computable analysis*, New computational paradigms: changing conceptions of what is computable (S. B. Cooper, B. Löwe, and A. Sorbi, eds.), Springer New York, 2008, pp. 425–491.

[13] Augustin-Louis Cauchy, *Exercises de mathématiques*, Bure Frères, 1826.

[14] Robert Cohen, *Egyptian fraction expansions*, Mathematics Magazine **46** (1973), no. 2, 76–80.

[15] Richard Dedekind, *Stetigkeit und irrationale Zahlen*, Braunschweig: Vieweg, 1872.

[16] Ding-Zhu Du and Ker-I Ko, *Theory of computational complexity*, Wiley Interscience, 2000.

[17] B. F. Finkel, O. Dunkel, and H. L. Olson, *Problems and solutions: 3173-3180.*, The American Mathematical Monthly **33** (1926), no. 3, 159.

[18] I. Georgiev, *Subrecursive graphs of representations of irrational numbers*, CiE 2023: Unity of Logic and Computation (G. Della Vedova, B. Dundua, S. Lempp, and F. Manea, eds.), LNCS, vol. 13967, Springer, Cham., 2023, pp. 154–165.

[19] _____, *Interplay between insertion of zeros and the complexity of dedekind cuts*, Computability **13** (2024), no. 2, 135–159.

[20] I. Georgiev, L. Kristiansen, and F. Stephan, *Computable irrational numbers with representations of surprising complexity*, Annals of Pure and Applied Logic **172** (2021), no. 2, 102893.

[21] G. H. Hardy and E. M. Wright, *Theory of numbers*, Oxford at the Clarendon Press, 1975.

[22] K. Hiroshima and A. Kawamura, *Elementarily traceable irrational numbers*, CiE 2023: Unity of Logic and Computation (G. Della Vedova, B. Dundua, S. Lempp, and F. Manea, eds.), LNCS, vol. 13967, Springer, Cham., 2023, pp. 135–140.

[23] A. Hurwitz, *Ueber die angenäherte Darstellung der Irrationalzahlen durch rationale Brüche*, Mathematische Annalen **39** (1891), 279–284.

[24] A. Ya. Khintchine, *Continued fractions.*, P. Noordhoff, Ltd., 1963, Translated by P. Wynn.

[25] K. Ko, *On the continued fraction representation of computable real numbers*, Theoretical Computer Science **47** (1986), 299–313.

[26] _____, *Corrigenda: On the continued fraction representation of computable real numbers*, Theoretical Computer Science **54** (1987), 341–343.

[27] _____, *Complexity theory of real functions*, Birkhäuser, 1991.

[28] K. Ko and H. Friedman, *Computational complexity of real functions*, Theoretical Computer Science **20** (1982), no. 3, 323 – 352.

[29] Ker-I Ko, *Complexity theory of real functions*, Birkhauser Boston Inc., Cambridge, MA, USA, 1991.

[30] Christoph Kreitz and Klaus Weihrauch, *Theory of representations*, Theoretical Computer Science **38** (1985), 35–53.

[31] L. Kristiansen, *On subrecursive representability of irrational numbers*, Computability **6** (2017), 249–276.

[32] _____, *On subrecursive representability of irrational numbers, part ii*, Computability **8** (2019), 43–65.

[33] _____, *On subrecursive representation of irrational numbers: Contractors and baire sequences*, CiE 2021: Connecting with Computability, LNCS, vol. 12813, Springer-Verlag, 2021, pp. 308–317.

[34] L. Kristiansen and J. G. Simonsen, *On the complexity of conversions between classic real number representations*, CiE 2020: Beyond the Horizon of Computability, LNCS, vol. 12089, Springer-Verlag, 2020, pp. 75–86.

[35] R. S. Lehman, *On primitive recursive real numbers*, Fundamenta Mathematica **49** (1961), no. 2, 105–118.

[36] A Mostowski, *On computable sequences*, Fundamenta Mathematica **44** (1957), 37–51.

[37] I. Richards, *Continued fractions without tears*, Mathematics Magazine **54** (1981), no. 4, 163–172.

[38] Michael Sipser, *Introduction to the theory of computation*, PWS Publishing Company, 1997.

[39] Hideki Tsuiki, *Real number computation through Gray code embedding*, Theoretical Computer Science **284** (2002), no. 2, 467–485.

[40] K. Weihrauch, *The degrees of discontinuity of some translators between representations of real numbers*, Tech. report, Fernuniversität Hagen, 1992.

[41] _____, *Computable analysis*, Springer Verlag, 2000.

[42] Klaus Weihrauch, *Type-2 recursion theory*, Theoretical Computer Science **38** (1985), 17–33.

[43] Klaus Weihrauch and Christoph Kreitz, *Representations of the real numbers and of the open subsets of the set of real numbers*, Annals of Pure and Applied Logic **35** (1987), 247–260.