Beta-Shifts, their Languages, and Computability

Jakob Grue Simonsen
Department of Computer Science
University of Copenhagen (DIKU)
Universitetsparken 1, DK-2100 Copenhagen Ø
Denmark
simonsen@diku.dk

Abstract

For every real number $\beta > 1$, the β-shift is a dynamical system describing iterations of the map $x \mapsto \beta x \bmod 1$ and is studied intensively in number theory. Each β-shift has an associated language of finite strings of characters; properties of this language are studied for the additional insight they give into the dynamics of the underlying system.

We prove that the language of the β-shift is recursive iff β is a computable real number. That fact yields a precise characterization of the reals: The real numbers β for which we can compute arbitrarily good approximations—hence in particular the numbers for which we can compute their expansion to some base—are precisely those for which there exists a program that decides for any finite sequence of digits whether the sequence occurs as a subword of some element of the β-shift.

While the "only if" part of the proof of the above result is constructive, the "if" part is not. We show that no constructive proof of the "if" part exists. Hence, there exists no *algorithm* that transforms a program computing arbitrarily good approximations of a real number β into a program deciding the language of the β-shift.

Keywords: Computability theory; dynamical systems; number theory; computable analysis

1 Introduction

Symbolic dynamics is a vast field of research having a wide variety of applications [23, 12, 2, 22]. A well-known class of symbolic dynamical systems is that of the *β-shifts* introduced by Rényi [26], developed by Parry in the seminal paper [25], and studied intensively thereafter, see for example [15, 28, 37, 21, 5, 10, 29, 30, 8, 7, 1].

Given a non-integral real number $\beta > 1$, write real numbers to base β; any non-negative real number x has a unique greedy expansion in powers of β in the same way as we ordinarily expand a number in powers of 10.

In case $x \in [0, 1)$, we may thus write:

$$x = \sum_{n=1}^{\infty} d_\beta(x)_n \beta^{-n}$$

where $d_\beta(x)_n \in \{0, \ldots, \lfloor \beta \rfloor\}$ is obtained by applying the "greedy algorithm": letting $\tau_\beta(y) \triangleq \beta y \bmod 1$, and setting $\tau_\beta^0(x) = x$, we define:

$$d_\beta(x)_n \triangleq \lfloor \beta \tau_\beta^{n-1}(x) \rfloor, \text{ for } n \geq 1$$

Observe that multiplication modulo 1 of x by β corresponds to left-shifting the integer sequence $(d_\beta(x))_{n\in\mathbb{N}}$ by one place, that is, "chopping off" the first element of the sequence:

$$\beta x \bmod 1 = \sum_{n=1}^{\infty} d_\beta(x)_{n+1}(x)\beta^{-n}$$

The symbolic dynamics associated to iterations of the map $x \mapsto \beta x \bmod 1$ — appropriately called the β-shift — is of significant interest in number theory.

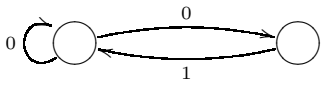
β-shifts have the additional interesting property of being one of the few "natural" classes of dynamical systems for which one of the most common topological invariants — the topological entropy — can assume any desired positive real value [16]; indeed, the topological entropy of the β-shift is $\log(\beta)$.

Any symbolic dynamical system has an associated *language* of finite strings of characters. Properties of this language are usually studied for the additional insight they give into the dynamics of the system; for instance, the topological entropy can be characterized by the language of the system alone.

The language, $\mathcal{L}(X_\beta)$, of the β-shift is the subset of the set of finite strings over the alphabet $\{0, \ldots, \lfloor\beta\rfloor\}$ such that $s \in \mathcal{L}(X_\beta)$ iff s occurs as a substring in the greedy expansion of some real number $x \in [0, 1)$.

The computational properties of β-shifts with particularly well-behaved languages have been studied extensively [10, 11, 9, 3], but most of this effort has been confined to β-shifts with quite simple languages, indeed to β-shifts of finite type and the larger class of sofic shifts — the latter corresponding to β-shifts whose languages are *regular*, that is, recognized by a regular expression, equivalently, recognized by a finite automaton.

Example 1. Consider the Golden ratio $\Phi = 1.681\cdots$ defined as the largest root of the monic polynomial $x^2 - x - 1$. The language, $\mathcal{L}(X_\Phi)$, of the Φ-shift consists of the set of finite binary strings x such that no matter how many initial bits are removed, the resulting string is lexicographically smaller than or equal to the infinite sequence $10101010\cdots$. That is, $\mathcal{L}(X_\Phi)$ is the set of strings generated by the cover automaton



It is well known that if β is a Pisot-Vijayaraghavan number (as, for instance, the Golden mean), then the β-shift is sofic, hence has language generated by a finite automaton as above [5].

However, there are more general languages available. In computability theory, a language L for which there is a Turing machine (or, by the Church-Turing thesis, any program in a sufficiently expressive programming language) that decides, given a finite input string, whether that string is in L, is called *decidable* (aka. *recursive*).

For symbolic dynamical systems more general than the β-shifts, progress has been made recently in finding good classifications of what computational properties of their languages are needed to ensure computability of topological invariants, especially the topological entropy [32, 35, 14, 13]. However, for β-shifts, not much is known when the shifts are no longer assumed to be sofic, bar some very interesting, but little known, results by Johnson [17]. There is no a priori connection between the — necessarily

countable — set of reals β associated to β-shifts with recursive languages and the — uncountable — set of non-integral reals > 1.

Ideally, if a β-shift has recursive language, we would expect that β should have a corresponding, nice computational property. At the very least, it should be possible to compute successively more accurate rational approximations to β, for instance by computing the digits of the expansion of β in some integer base. Reals β for which computing such rational approximations are possible — introduced by Turing in the papers that first defined the Turing machine [38, 39] — are known as *computable reals*.

We shall prove in this paper that there *is* a remarkable connection between decidability of $\mathcal{L}(X_\beta)$ and computability of β: Indeed, $\mathcal{L}(X_\beta)$ is a recursive set iff β is a computable real number.

For the purpose of truly *practical* computation, the correspondence stated above between recursive sets and computable reals is not wholly satisfactory: indeed, there should be programs *witnessing* the correspondence (in the language of computability theory: The correspondence should be uniformly recursive). Intuitively, there should be a *program* that converts programs deciding the language of a β-shift into a program computing approximations to the computable real β, and vice versa. Perhaps surprisingly, this turns out to be *impossible* for the "vice versa" part. Indeed, we prove that there is *no constructive proof* (hence no program) that converts its input—a program yielding successively more accurate approximations to some computable real β—into a program deciding the language of the corresponding β-shift. However, if we restrict our attention to the class of reals β for which 1 has an *infinite* greedy β-expansion, it *is* possible to find a constructive proof.

Remark 1. After the publication of the conference version [31] of this paper, the author became aware that Kimberly Johnson at an earlier time has proved some of our results in her dissertation [17]. Unfortunately, Johnson never published her results elsewhere[1]. While her proof methods are different from ours, we note that Johnson can rightfully claim to be the first to prove that the language of the β-shift is recursive iff the greedy expansion $d_\beta(1)$ of 1 is generated by a Turing machine. Indeed, we prompt the reader to peruse Johnson's dissertation — it contains many other results of interest, for instance the delightfully surprising theorem that the language of a β-shift is context-free iff it is regular. Using the new methods in the present paper, we are able to answer some of Johnson's open questions, see page 10. Note that the main contribution of this paper—the correspondence between computability of β and decidability of the language of the β-shift—is new.

As notions from the fairly disparate fields of dynamical systems and computability theory are employed, we provide a more thorough list of definitions than common in most papers. Readers with background in dynamical systems may safely skip Section 2, whereas readers with background in computability theory and computable analysis may safely skip Section 3.

[1] As of the time of writing, the reference [17] is freely available on CiteSeer.

2 Preliminaries on languages and β-shifts

We now move on to preliminary definitions.

Definition 1: *Let Σ be a nonempty, finite set of symbols. The set of all finite strings of elements of Σ is denoted by Σ^*, the set of all right-infinite strings by $\Sigma^{\mathbb{N}}$. The set of all bi-infinite sequences of elements of Σ is denoted by $\Sigma^{\mathbb{Z}}$. The empty string is denoted by λ. The concatenation of a finite string $a \in \Sigma^*$ and finite or infinite string $b \in \Sigma^{\mathbb{N}} \cup \Sigma^*$ is denoted by $a \cdot b$ or simply ab. If $a \in \Sigma^*$, the right-infinite string constructed by an infinite concatenation of copies of a is denoted by a^ω.*

A language *over Σ is a subset $L \subseteq \Sigma^*$. Language L is said to be* factorial *iff $v \cdot w \in L$ (where $v, w \in \Sigma^*$) implies $v, w \in L$. L is said to be* extensible *if $w \in L$ implies existence of $c, d \in \Sigma$ s.t. $c \cdot w \cdot d \in L$.*

Assuming Σ to be ordered, we define the lexicographic order $\leq_{lex}$ on Σ^* and $\Sigma^{\mathbb{N}}$ as usual; we extend the lexicographic order to the set $\Sigma^* \cup \Sigma^{\mathbb{N}}$ as follows: let c be the least element of Σ, let $\hat{x} = x \cdot c^\omega$ for all $x \in \Sigma^*$, let $\hat{y} = y$ for all $y \in \Sigma^{\mathbb{N}}$, and set $a \leq_{lex} b$ for $a, b \in \Sigma^* \cup \Sigma^{\mathbb{N}}$ iff $\hat{a} \leq_{lex} \hat{b}$.

Definition 2: *The (one-sided)* shift map *$\sigma : \Sigma^* \longrightarrow \Sigma^*$ is defined by $\sigma(\lambda) = \lambda$ and $\sigma(a \cdot b) = b$ where $a \in \Sigma$. The shift map extends naturally to a map $\sigma : \Sigma^{\mathbb{N}_0} \longrightarrow \Sigma^{\mathbb{N}_0}$ by $\sigma(\lambda) = \lambda$ and $\sigma(a \cdot b) = b$ where $a \in \Sigma$ and $b \in \Sigma^{\mathbb{N}_0}$.*

We now turn to β-shifts, starting with greedy expansions:

Definition 3: *Let $\beta > 1$ be a real number. The* greedy expansion of *1* in powers of β^{-1} *is the sequence $d_\beta(1) = (d_\beta(1)_n)_{n \in \mathbb{N}}$ where $d_\beta(1)_1 = \lfloor \beta \rfloor$, and $d_\beta(1)_n = \lfloor \beta^n - \sum_{i=1}^{n-1} d_\beta(1)_i \beta^{n-i} \rfloor$ for $n > 1$. If there is an $m \in \mathbb{N}$ such that for all $n \geq m$, we have $d_\beta(1)_n = 0$, then the expansion is said to be* finite.

It is easy to see that $1 = \sum_{n=1}^{\infty} d_\beta(1)_n \beta^{-n}$, and that β is the unique positive solution of $1 = \sum_{n=1}^{\infty} d_\beta(1)_n x^{-n}$. Observe that if $k = \lfloor \beta \rfloor + 1$, then $0 \leq d_\beta(1)_n \leq k - 1$ for all $n \in \mathbb{N}$, and thus $d_\beta(1)$ is an element of the full shift on k letters, that is, the set of all right-infinite sequences of words from a k-letter alphabet, which is necessarily unique up to one-to-one renaming of the letters.

We define both the "naïve" W-β-shift, which admits certain finite greedy expansions, and the β-shift proper.

Definition 4: *Let $\beta > 1$ be a non-integral real number. If $d_\beta(1)$ is finite, that is,*

$$d_\beta(1) = d_\beta(1)_1 d_\beta(1)_2, \ldots d_\beta(1)_k 0^\omega$$

such that $d_\beta(1)_k > 0$, define

$$d^*_\beta(1) \triangleq (d_\beta(1)_1 d_\beta(1)_2 \cdots d_\beta(1)_{k-1}(d_\beta(1)_k - 1))^\omega$$

. Otherwise, define $d^*_\beta(1) = d_\beta(1)$.

The one-sided W-β-shift *is the subset $\tilde{X}_\beta$ of $\{0, \ldots, \lfloor \beta \rfloor\}^{\mathbb{N}}$ containing exactly those b such that, for all $n \in \mathbb{N}_0$, we have $\sigma^n(b) \leq_{lex} d_\beta(1)$, that is, all suffixes of b are lexicographically smaller than or equal to $d^*_\beta(1)$.*

The one-sided β-shift*, denoted X_β, is the subset X_β of $\{0, \ldots, \lfloor \beta \rfloor\}^{\mathbb{N}}$ containing exactly those b such that for all $n \in \mathbb{N}_0$, we have $\sigma^n(b) \leq_{lex} d^*_\beta(1)$.*

The two-sided *W-β-shift is the subset of $\{0, 1\}^{\mathbb{Z}}$ containing exactly those b such that, for all $i \in \mathbb{Z}$, we have $b_i b_{i+1} b_{i+2} \cdots \in \tilde{X}_\beta$. The two-sided β-shift is defined analogously, using X_β.*

The reader will notice that we could have extended the definition to integers $\beta > 1$ by defining $\tilde{X}_\beta = X_\beta = \{0, \ldots, \beta - 1\}^{\mathbb{N}}$ in that case.

Note that if $d_\beta(1)$ is not finite, then $d^*_\beta(1) = d_\beta(1)$ and hence $\tilde{X}_\beta = X_\beta$. It is easy to see that both the one- and two-sided (W-)β-shifts are shift-invariant subsets of $\{0, \ldots, \lfloor\beta\rfloor\}^{\mathbb{N}}$ and $\{0, \ldots, \lfloor\beta\rfloor\}^{\mathbb{Z}}$, that is, $\sigma(\tilde{X}_\beta) = \tilde{X}_\beta$ and $\sigma(X_\beta) = X_\beta$.

The "W-β-shift" corresponds to a shift with "uncorrected" greedy expansions of 1 and does not correspond to the dynamics of the map $x \mapsto \beta x \bmod 1$. It is sometimes, confusingly, simply called the β-shift, for example in [40].

Definition 5: *Let $\beta > 1$ The* language *of the β-shift X_β, denoted $\mathcal{L}(X_\beta)$, is the set of finite strings occurring as substrings of elements of X_β. The language of the W-β-shift is defined analogously.*

The language of the two-sided β-shift is defined exactly as for the one-sided β-shift. It is easy to see that the one- and two-sided β-shifts have identical languages, and as we are concerned mainly with the language and not the shift itself, we may thus restrict ourselves to the one-sided β-shift from now on, as it is easier to deal with in our particular technical exposition. It is easy to see that the languages of all the mentioned varieties of the β-shift are all factorial and extensible.

The *topological entropy* of a β-shift is defined as $\mathrm{h_{top}}(X_\beta) = \lim_{n\to\infty} |\mathcal{L}(X_\beta) \cap \Sigma^n|/n$ [16]. Both the W-β-shift and the β-shift satisfy the following property [40, 8]:

Theorem 1: *Let $\beta > 1$ be real number. Then $\mathrm{h_{top}}(X_\beta) = \mathrm{h_{top}}(\tilde{X}_\beta) = \log(\beta)$.*

We shall need the following fundamental result by Parry:

Theorem 2: *[Parry [25]] For any $\beta > 1$, the greedy expansion $d_\beta(1)$ satisfies $\sigma^k(d_\beta(1)) <_{lex} d_\beta(1)$ for all $k \geq 1$. Conversely, if $\gamma \in \Sigma^{\mathbb{N}}$ satisfies $\sigma^k(\gamma) <_{lex} \gamma$ for all $k \geq 1$, then $\gamma = d_\beta(1)$ for some real number $\beta > 1$.*

Corollary 1: *$\sigma^k(d^*_\beta(1)) \leq_{lex} d^*_\beta(1)$ for all $k \geq 0$. If $\sigma^k(\gamma) \leq_{lex} \gamma$ for all $k \geq 0$, then $\gamma = d^*_\beta(1)$ for some $\beta > 1$.*

We consequently obtain straightforward characterisations of $\mathcal{L}(\tilde{X}_\beta)$ and $\mathcal{L}(X_\beta)$:

Corollary 2: *For any non-integral real $\beta > 1$, it is the case that $y \in \mathcal{L}(\tilde{X}_\beta)$ iff for all $n \in \{0, \ldots, |y|\}$ we have $\sigma^n(y) \leq_{lex} d_\beta(1)_1 \cdots d_\beta(1)_{|y|}$. Correspondingly, we have $y \in \mathcal{L}(X_\beta)$ iff $\sigma^n(y) \leq_{lex} d^*_\beta(1)_1 \cdots d^*_\beta(1)_{|y|}$.*

3 Computability theory

We give only the briefest of introductions; ample introductory material on computability theory can be found in standard textbooks, for example [27, 24, 18]. Introductory material on computable reals can be found in [19, 41].

Recall that a partial function $f : A \longrightarrow C$ is a function $g : B \longrightarrow C$ defined on a subset $B \subseteq A$. We have:

Definition 6:

A partial function $f : \mathbb{N} \longrightarrow \mathbb{N}$ is said to be (partial) recursive *if there exists a Turing machine with index i that halts exactly on the elements of $dom(f)$ and for all $n \in dom(f)$ outputs $f(n)$ in finite time, in which case we write $f = \phi_i$. A function $f : \mathbb{N} \longrightarrow \mathbb{N}$ is said to be* total recursive *if it is partial recursive and $dom(f) = \mathbb{N}$. The nomenclature is also used with any occurrence of $\mathbb{N}$ replaced by Σ^*, $\mathbb{Q}$ or any finite set.*

Note that there is a computable encoding of Turing machines M as indices $i \in \mathbb{N}$ with computable inverse [27]. For most purposes, we may thus switch at will between considering M and its index i.

Definition 7: *A set $A \subseteq \mathbb{N}$ is* recursive *(aka.* decidable*) if there is a total recursive function $\phi_i : \mathbb{N} \longrightarrow \{0,1\}$ with $\phi_i(n) = 1$ iff $n \in A$. As before, we shall use the same nomenclature with $\mathbb{N}$ replaced by Σ^*, $\mathbb{Q}$, or any finite set.*

A set $A \subseteq \mathbb{N}$ is recursively enumerable *(abbreviated* r.e.*) if there is a total recursive function $\phi : \mathbb{N} \longrightarrow \mathbb{N}$ with $\phi(\mathbb{N}) = A$. The set A is* co-recursively enumerable *(abbreviated* co-r.e.*) if its complement $\mathbb{N} - A$ is r.e.*

For readers unfamiliar with computability theory, the essence of the above definition is that a subset A of the natural numbers (or set of all finite strings Σ^*) is recursive iff there is a program that on input n outputs "yes" or "no" in finite time depending on whether n is in A or not.

It is easy to see that a set is recursive iff it is both r.e. and co-r.e.

We shall have occasion to refer to the standard notion of the *jump* $\emptyset' = \{i \in \mathbb{N} : \phi_i(i) \text{ is defined}\}$. It is straightforward to see that $\emptyset'$ is r.e., but not co-r.e. (hence not recursive).

3.1 Computable real numbers

With modern notation, we may introduce the computable reals as follows:

Definition 8: *A real number α is* computable *if there exists a total recursive function $\phi_i : \mathbb{N} \longrightarrow \mathbb{Q}$ such that, for all $n \in \mathbb{N}$, we have $|\alpha - \phi(n)| < 2^{-n}$. In this case, the index i is called a* computable name *of α.*

Thus, a real number is computable if there is a program computing rapidly converging rational approximations to it. Any rational number, any algebraic number, π and e, as well as any number for which we can compute its digits to some base[2], are computable reals.

A moment's thought reveals that $\alpha \in \mathbb{R}$ is computable iff there are total recursive functions $\phi : \mathbb{N} \longrightarrow \mathbb{Q}$ and $\xi : \mathbb{N} \longrightarrow \mathbb{N}$ such that, for each $n \in \mathbb{N}$, $m > \xi(n)$ implies that $|\alpha - \phi(m)| < 2^{-n}$ (in which case ξ is called a *computable modulus of convergence* for ϕ).

Definition 9: *A real number α is* left- *(resp.* right-*)computable if there exists a total recursive function $\phi_i : \mathbb{N} \longrightarrow \mathbb{Q}$ such that $\alpha = \sup_n \phi_i(n)$ (resp. $\alpha = \inf_n \phi_i(n)$).*

It is not hard to prove that a real number α is computable iff it is both left- and right-computable. Also, a moment's thought reveals that, in the definition of left-(resp. right-)computability, we may wlog. assume ϕ_i to be non-decreasing (resp. non-increasing).

The following two auxiliary propositions follow straightforwardly by reduction from the Halting Problem[3]

Proposition 1: *There is no recursive set D such that, for all i where ϕ_i is total, $i \in D$ iff $\exists n.\phi_i(n) = 1$.*

Equivalently, the question of whether there is a k such that the value of a total recursive function ϕ_i on k is 1 (as opposed to 0) is undecidable.

Proposition 2: *For any computable real x, the following problems are all undecidable: Given a name of a computable real y, (1) decide whether $y \leq x$ ($x \leq y$), (2) whether $y < x$ ($x < y$), (3) whether $x = y$. Identical results hold if x and y are restricted to be elements of any non-degenerate interval on the real line.*

[2] "compute the digits" here means: There is a program that on input natural number n in finite time yields the nth digit of some expansion of the number to whatever integer base we consider. Note that conversely, if a real number is computable, we cannot necessarily compute the canonical expansion of a number, but may have to "switch" between several possible expansions (for example between "$0.99999\cdots$" and "1") depending on the current digit n.

[3] One may in fact give an even more elementary treatment of the two results without referring to the Halting Problem explicitly: Proposition 1 is essentially a rephrasing of "The Limited Principle of Omniscience" ("LPO") in constructive mathematics and Proposition 2 a rephrasing of some of the standard consequences that follow if LPO does *not* hold. See for example [6, Ch.1,3] or [4, Ch.1,2].

4 Correspondence results for $d_\beta(1)$, $d^*_\beta(1)$ and $\mathcal{L}(X_\beta)$

The rest of the paper concerns exposition of our new results. We begin with a simply stated lemma.

Lemma 1: *Let $\beta > 1$ and let, for each $a \in \{1, \ldots, \lfloor\beta\rfloor\}$, the set $\{n : d^*_\beta(1)_n = a\}$ be recursively enumerable. Then, $\mathcal{L}(X_\beta)$ is recursively enumerable.*

Proof: Let $i_{(a)}$ be the index of a Turing machine enumerating $\{n : d^*_\beta(1)_n = a\}$.

By definition of the β-shift, we have that $a \neq b$ implies that $\phi_{i_{(a)}}(k) \neq \phi_{i_{(b)}}(m)$ for all $k, m \in \mathbb{N}$ (otherwise $a = d^*_\beta(1)_{\phi_{i_{(a)}}(k)} = d^*_\beta(1)_{\phi_{i_{(b)}}(m)} = b$, a contradiction).

For each k, we may compute $\phi_{i_{(a)}}(1), \ldots, \phi_{i_{(a)}}(k)$ in finite time. Performing this computation for each a in the range $1, \ldots, \lfloor\beta\rfloor$ thus allows us to compute the finite set $N_k \subseteq \mathbb{N} \times \{1, \ldots, \lfloor\beta\rfloor\}$ defined by:

$$N_k = \{(\phi_{i_{(1)}}(1), 1), \ldots, (\phi_{i_{(1)}}(k), 1)\} \cup \cdots \cup \{(\phi_{i_{(\lfloor\beta\rfloor)}}(1), \lfloor\beta\rfloor), \ldots, (\phi_{i_{(\lfloor\beta\rfloor)}}(k), \lfloor\beta\rfloor)\}$$

Order N_k by the first component of its pairs, and write the thus ordered set as $\{(p_1, h_1), \ldots, (p_l, h_l)\}$, where $0 < p_1 < p_2 < \cdots < p_l$ and $h_a \in \{1, \ldots, \lfloor\beta\rfloor\}$ for all $a \in \{1, \ldots, l\}$.

Consider the finite string $a(k) \triangleq 0^{p_1-1}h_1 0^{p_2-p_1-1}h_2 \cdots 0^{p_l-p_{l-1}-1}h_l$ (where we take $0^0 = \lambda$, the empty string).

The string $a(k)$ is thus an "approximation" to $d^*_\beta(1)_1 \cdots d^*_\beta(1)_{p_l}$—it is intuitively obtained by taking the string 0^{p_l} and "plugging in" the h_a (corresponding to the $d^*_\beta(1)_n$ that we know so far) at the correct places.

All steps in the above are computable, and there is thus a Turing machine that takes as input k and outputs the string $a(k)$.

By construction, the rth element of $a(k)$ is h_a (> 0) *only if* the rth element of $d^*_\beta(1)_1 \cdots d^*_\beta(1)_{p_l}$ is h_a. Thus, $\sigma^r(a(k) \cdot 0^\omega) \leq_{\text{lex}} \sigma^r(d^*_\beta(1)) \leq_{\text{lex}} d^*_\beta(1)$, whence the set $A_k \subseteq \{0, \ldots, \lfloor\beta\rfloor\}^*$ defined by

$$A_k \triangleq \{a_1 \cdots a_j : \forall r \in \{0, \ldots, j\}.\sigma^r(a_1 \cdots a_j) \leq_{\text{lex}} a(k) \cdot 0^\omega\}$$

satisfies $A_k \subseteq \mathcal{L}(X_\beta)$.

Again, there is a Turing machine taking k as input and outputting a representation of the set A_k.

As $a(k) \cdot 0^\omega \leq_{\text{lex}} a(k+1)0^\omega$, it follows from the definition of A_k that $A_k \subseteq A_{k+1}$ for all $k \in \mathbb{N}$.

We claim that $\mathcal{L}(X_\beta) = \bigcup_{k\in\mathbb{N}} A_k$. The inclusion $\bigcup_{k\in\mathbb{N}} A_k \subseteq \mathcal{L}(X_\beta)$ is immediate. To see that $\mathcal{L}(X_\beta) \subseteq \bigcup_{k\in\mathbb{N}} A_k$, let $x \in \mathcal{L}(X_\beta)$. By Parry's Theorem, cf. Corollary 1, we have $\sigma^r(x) \leq_{\text{lex}} d^*_\beta(1)_1 \cdots d^*_\beta(1)_{|x|}$ for all $r \in \{1, \ldots, |x|\}$. There exists a $k \in \mathbb{N}$ such that all the positions in $d^*_\beta(1)_1 \cdots d^*_\beta(1)_{|x|}$ that are *not* zero have been enumerated in N_k[4]. But then $x \in A_k$, showing the desired inclusion.

Each of the sets A_k is recursively enumerable; hence so is $\mathcal{L}(X_\beta) = \bigcup_{k\in\mathbb{N}} A_k$, as desired. □

We have a corresponding lemma for co-r.e. sets:

Lemma 2: *Let $\beta > 1$ and let, for each $a \in \{0, \ldots, \lfloor\beta\rfloor - 1\}$, the set $\{n : d^*_\beta(1)_n = a\}$ be co-recursively enumerable (that is, its complement is recursively enumerable). Then, $\mathcal{L}(X_\beta)$ is co-recursively enumerable.*

[4]But not necessarily a k that we can compute given the $i_{(a)}$—existence in this case is inherently non-constructive.

Proof: Let $i^{(a)}$ be the index of a Turing machine enumerating $\{n : d^*_\beta(1)_n \neq a\}$. As in the proof of Lemma 1, consider, for $k \in \mathbb{N}$, the finite set M_k defined by:

$$M_k = \{(\phi_{i^{(0)}}(1), 0), \ldots, (\phi_{i^{(0)}}(k), 0)\} \cup \cdots \cup \{(\phi_{i^{(\lfloor\beta\rfloor-1)}}(k), \lfloor\beta\rfloor), \ldots, (\phi_{i^{(\lfloor\beta\rfloor-1)}}(k), \lfloor\beta\rfloor)\}$$

Order M_k by the first component of its pairs (note that, unlike the proof of Lemma 1, distinct pairs might now share the same first component due to the fact that the machines $i^{(a)}$ all enumerate *complements*). Write the ordered set as $\{(d_1, h_1), \ldots, (d_l, h_l)\}$ (that is, with $d_1 \leq d_2 \leq \cdots \leq d_l$).

For each $r \in \mathbb{N}$, define the (possibly empty) set

$$J_r \triangleq \{a \in \{0, \ldots, \lfloor\beta\rfloor\} : \exists w \in \{1, \ldots, l\}.(d_w, h_w) = (r, j)\}$$

and define $y_r \triangleq \max(\{0, \ldots, \lfloor\beta\rfloor\} \setminus J_r)$

Intuitively, y_r is the "currently known upper bound" on the rth element of $d^*_\beta(1)_n$ when all the (finitely many) machines $i^{(a)}$ have been run on input $1, \ldots, k$.

Consider the finite string $d(k)$ defined by

$$d(k) \triangleq y_1 y_2 \cdots y_r \cdots y_{d_l}$$

One can think of $d(k)$ as obtained by starting with the string $\lfloor\beta\rfloor^{d_l}$ and subsequently, for all elements of the string, replacing the rth element by "the least upper bound on $d^*_\beta(1)_r$ seen so far by applying the $i^{(a)}$ to inputs $1, \ldots, k$".

Observe that for any $r \in \mathbb{N}$, the rth symbol of $d(k) \cdot \lfloor\beta\rfloor^\omega$ is $0 \leq a \leq \lfloor\beta\rfloor$ *only if* the rth element of $d^*_\beta(1)$ is $\leq a$.

Thus, $d^*_\beta(1) \leq_{\text{lex}} \sigma^j(d(k) \cdot \lfloor\beta\rfloor^\omega)$ for all $r \in \mathbb{N}$. For any $y \in \Sigma^{\leq d_l}$, if there exists $r \in \{0, \ldots, d_l\}$ with $\sigma^r(y) >_{\text{lex}} d(k)$, we hence have $y \in \Sigma^* \setminus \mathcal{L}(X_\beta)$.

Thus, the set

$$B_k \triangleq \{a_1 \cdots a_j : \exists r \in \{0, \ldots, j\}.d(k) \cdot \lfloor\beta\rfloor^\omega <_{\text{lex}} \sigma^r(a_1 \cdots a_j)\}$$

satisfies $B_k \subseteq \{0, \ldots, \lfloor\beta\rfloor\}^* \setminus \mathcal{L}(X_\beta)$. B_k is clearly r.e., and hence so is the set $\bigcup_{k\in\mathbb{N}} B_k$.

We claim that $\{0, \ldots, \lfloor\beta\rfloor\}^* \setminus \mathcal{L}(X_\beta) = \bigcup_{k\in\mathbb{N}} B_k$. To see this, first note that the above observations yield that $\bigcup_{k\in\mathbb{N}} B_k \subseteq \{0, \ldots, \lfloor\beta\rfloor\}^* \setminus \mathcal{L}(X_\beta)$.

To obtain the converse inclusion, let $x \in \{0, \ldots, \lfloor\beta\rfloor\}^* \setminus \mathcal{L}(X_\beta)$. Then there exists an $r \in \{0, \ldots, |x|\}$ such that $d^*_\beta(1)_1 \cdots d^*_\beta(1)_{|x|} <_{\text{lex}} \sigma^r(x)$. But by construction, there also exists a $k \in \mathbb{N}$[5] such that, the $y_1, \ldots, y_{d_l}$ in the construction of $d(k)$ satisfy $|x| \leq d_l$ and $y_1 = d^*_\beta(1)_1, \ldots, y_{d_l} = d^*_\beta(1)_{d_l}$. Hence, for this k, we will have $d(k) \cdot \lfloor\beta\rfloor^\omega <_{\text{lex}} \sigma^r(x)$, showing that $x \in B_k$, and we obtain the desideratum. □

By combining Lemmas 1 and 2, we obtain the following corollary.

Corollary 3: *Let $\beta > 1$ and let, for each $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, the set $\{n : d^*_\beta(1)_n = a\}$ be recursive. Then $\mathcal{L}(X_\beta)$ is recursive.*

[5]Again, there is not necessarily an algorithm to *compute* k given x, but that is irrelevant to the correctness of the proof.

We do not know whether the assumption that $\mathcal{L}(X_\beta)$ is r.e. (resp. co-r.e.) implies that $\{j : d^*_\beta(1)_j = i\}$ is r.e. (resp. co-r.e.) for all $i \in \{1, \ldots, \lfloor\beta\rfloor\}$. However, if we make the stronger assumption that $\mathcal{L}(X_\beta)$ is recursive, we obtain the following:

Lemma 3: *Let $\beta > 1$ and let $\mathcal{L}(X_\beta)$ be recursive. Then, for all $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, the set $\{n : d^*_\beta(1)_n = a\}$ is recursive.*

Proof: Observe that, for any $n \in \mathbb{N}$, $d^*_\beta(1)_1 \cdots d^*_\beta(1)_n$ satisfies $d^*_\beta(1)_1 \cdots d^*_\beta(1)_n \in \mathcal{L}(X_\beta)$. For any $x \in \mathcal{L}(X_\beta) \cap \{0, \ldots, \lfloor\beta\rfloor\}^n$ and $k \in \{0, \ldots, n\}$, we have $\sigma^k(x) \leq_{\text{lex}} d^*_\beta(1)_1 \cdots d^*_\beta(1)_n$. Thus, $y \in \mathcal{L}(X_\beta) \cap \{0, \ldots, \lfloor\beta\rfloor\}^n$ satisfies $y = d^*_\beta(1)_1 \cdots d^*_\beta(1)_n$ iff for all $k \in \{0, \ldots, n\}$ and $x \in \mathcal{L}(X_\beta) \cap \{0, \ldots, \lfloor\beta\rfloor\}^n$ we have $\sigma^k(x) \leq_{\text{lex}} y$.

As $\mathcal{L}(X_\beta)$ is recursive, there is a Turing machine with index i such that $\phi_i : \{0, \ldots, \lfloor\beta\rfloor\}^* \longrightarrow \{0, 1\}$ is a total function and $\phi_i(x) = 1$ iff $x \in \mathcal{L}(X_\beta)$. Using the machine with index i, we may then computably obtain, for each $l \in \mathbb{N}$, the unique element $y \in \mathcal{L}(X_\beta) \cap \{0, \ldots, \lfloor\beta\rfloor\}^l$ satisfying $\sigma^k(x) \leq_{\text{lex}} y$ for all $k \in \{0, \ldots, l\}$ and all $x \in \mathcal{L}(X_\beta) \cap \{0, \ldots, \lfloor\beta\rfloor\}^n$.

The nth element of y is $d^*_\beta(1)_n$, and using the machine with index i we may thus decide whether $d^*_\beta(1)_j = a$, for all $a \in \{0, \ldots, \lfloor\beta\rfloor\}$. Hence, $\{n : d^*_\beta(1)_n = a\}$ is recursive for all $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, concluding the proof. □

By the above lemma, if $\mathcal{L}(X_\beta)$ is recursive, we may compute the elements of the sequence $d^*_\beta(1)$.

We now have our first theorem:

Theorem 3: *Let $\beta > 1$. Then, $\mathcal{L}(X_\beta)$ is recursive iff for all $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, the set $\{n : d^*_\beta(1)_n = a\}$ is recursive.*

Proof: By Corollary 3 and Lemma 3. □

In colloquial terms, Theorem 3 may be stated as: "It is as hard to produce the digits of $d^*_\beta(1)$ in ascending order as it is to decide $\mathcal{L}(X_\beta)$."

As corollaries of Theorem 3, we can now give alternative proofs of the following two results, originally due to Johnson [17]:

Corollary 4: *[[17, Sec. 4.5, implicit]] Let $\beta > 1$ be a non-integral real. Then there exists a Turing machine with index i such that, for each $n \in \mathbb{N}$, $\phi_i(n) = d^*_\beta(1)_n$ iff $\mathcal{L}(X_\beta)$ is recursive.*

Proof: There is a Turing machine with index i such that, for each $k \in \mathbb{N}$, $\phi_i(k) = d^*_\beta(1)_k$ iff the sets $\{j : d^*_\beta(1)_j = a\}$ are recursive for each $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, and the result follows. □

Correspondingly, for the greedy expansion $d_\beta(1)$, we have the below corollary:

Corollary 5: *[[17, Cor. 4.5.8]] Let $\beta > 1$ be a non-integral real. Then there is a Turing machine with index i such that, for each $n \in \mathbb{N}$, $\phi_i(n) = d_\beta(1)_n$ iff $\mathcal{L}(X_\beta)$ is recursive.*

Proof: If $d_\beta(1)$ is not finite, then $d_\beta(1) = d^*_\beta(1)$ and the result follows from Corollary 4. If $d_\beta(1)$ is finite, then the sets $\{n : d^*_\beta(1)_n = a\}$ can all be generated by finite automata, hence are all recursive. □

We could have stated a slightly sharper version of Theorem 3: By inspecting the proofs of the present section, we see that there is a Turing machine that, for each $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, as input takes (the index of) a Turing machine deciding $\mathcal{L}(X_\beta)$ and produces (the index of) a Turing machine that decides $\{n : d^*_\beta(1)_n = a\}$. Conversely, there is a Turing machine that as input takes (the index of) a Turing machine deciding $\{j : d^*_\beta(1)_j = a\}$ and produces (the index of) a Turing machine deciding $\mathcal{L}(X_\beta)$. In computing vernacular, this amounts to the statement that $\mathcal{L}(X_\beta)$ is *uniformly recursive* in the finite collection of sets $\{n : d^*_\beta(1)_n = a\}$ (for $a \in \{0, \ldots, \lfloor\beta\rfloor\}$) and vice versa. Thus, the biimplication of the

theorem is *constructive*: There is an algorithm witnessing that we can go back-and-forth between each side of the biimplication. As we shall see in Section 6, obtaining such an algorithm is not possible for passing back-and-forth from $\mathcal{L}(X_\beta)$ to algorithms for computing β.

In the final part of this section, we establish the following separation result between reals that answers in the positive the open question posed by Johnson [17, Sec. 5.3] whether there exists a β-shift with recursively enumerable, but not recursive language.

Proposition 3: *The following hold:*

1. *There exists a real number $1 < \beta < 2$ such that $\mathcal{L}(X_\beta)$ is r.e., but not co-r.e. (hence not recursive).*
2. *There exists a real number $1 < \beta < 2$ such that $\mathcal{L}(X_\beta)$ is co-r.e., but not r.e. (hence not recursive).*

Proof: Choose a (necessarily infinite) subset, $A \subseteq \mathbb{N}$ that is r.e., but not co-r.e. (for instance the jump $\emptyset'$). Let $\gamma \in \{0,1\}^{\mathbb{N}}$ be such that the 2^jth element of γ is 1 iff $j \in A$ and all other elements are 0. Then, $\sigma^k(\gamma) \leq_{\text{lex}} \gamma$ for all $k \in \mathbb{N}_0$ and Corollary 1 thus ensures existence of a real number $1 < \beta < 2$ such that $\gamma = d^*_\beta(1)$. Now, $\{j : \gamma_j = 1\} = \{2^n : n \in A\}$ is clearly recursively enumerable. If it were also co-r.e., it would be recursive, implying that A would be recursive, hence also co-r.e., contradicting the assumptions.

Consider $\mathcal{L}(X_\beta)$. By Lemma 1, $\mathcal{L}(X_\beta)$ is recursively enumerable. Were $\mathcal{L}(X_\beta)$ also co-r.e., it would be recursive, and Lemma 3 would yield that $\{j : \gamma_j = 1\}$ were recursive, hence in particular co-r.e., a contradiction.

The proof of existence of a β-shift with a language that is co-r.e., but not r.e. is performed in the same way: Letting $A \subseteq \mathbb{N}$ be co-r.e. but not r.e., simply note that $\{j : \gamma_j = 1\} = \{2^n : n \in A\}$ is co-r.e. iff A is co-r.e. □

5 Correspondence Results for $d_\beta(1)$, $d^*_\beta(1)$ and β

We now investigate the correspondence between computability of β and computability of $d_\beta(1)$ and $d^*_\beta(1)$. We first make an observation concerning solutions to equations on the form $1 = \sum_{n=1}^{\infty} a_n x^{-n}$.

Remark 2. For any sequence $(a_n)_{n\in\mathbb{N}}$ in $\{0, \ldots, \lfloor\beta\rfloor\}^{\mathbb{N}}$ that has at least one non-zero element, observe that the equation $1 = \sum_{n=1}^{\infty} a_n x^{-n}$ has exactly one positive real solution.

If all elements of $(a_j)_{j\in\mathbb{N}}$ are zero after a certain step $N \in \mathbb{N}$, we can find the solution of the above effectively by observing that the largest positive real root of the monic polynomial $x^N(1 - \sum_{j=1}^{N} a_j x^{-j})$ (1) is the unique positive solution of $1 = \sum_{j=1}^{N} a_j x^{-j}$, (2) is a computable real number, and (3) given the (finite set of) non-zero coefficients a_n can be found by standard root-finding methods from computational algebra, see for example [43].

Definition 10: *Let $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ be sequences of elements of $\mathbb{N}_0$. Write $(a_n)_{n\in\mathbb{N}} \preceq (b_n)_{n\in\mathbb{N}}$ if, for all $n \in \mathbb{N}$, we have $a_n \leq b_n$.*

Proposition 4: *Let $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ be sequences of elements of $\mathbb{N}_0$ such that (1) for all $n \in \mathbb{N}$, we have $a_n \leq b_n$, and (2) both sequences contain at least one non-zero element. Let the unique positive solutions to $1 = \sum_{n\in\mathbb{N}} a_n x^{-n}$ and $1 = \sum_{n\in\mathbb{N}} b_n x^{-n}$ be x_a and x_b, respectively. Then, $x_a \leq x_b$.*

Proof: Assume $x_a > x_b$. Then $\sum_{n\in\mathbb{N}} b_n x_a^{-n} < 1$. But as $a_n < b_n$ for all $n \in \mathbb{N}$, we have $1 = \sum_{n\in\mathbb{N}} a_n x_a^{-n} \leq \sum_{n\in\mathbb{N}} b_n x_a^{-n} < 1$, a contradiction. □

We can now establish the following.

Lemma 4: *Let, for all $a \in \{1, \ldots, \lfloor\beta\rfloor\}$, the sets $\{n \in \mathbb{N} : d_\beta(1)_n = a\}$ be recursively enumerable. Then the unique positive solution, x_S, to $1 = \sum_{n\in\mathbb{N}} d_\beta(1)_n x^{-n}$ is a left-computable real number.*

Proof: By the assumption that the sets $\{n \in \mathbb{N} : d_\beta(1)_n = a\}$ are r.e., we may construct a Turing machine with index i such that $\phi_i : \mathbb{N} \longrightarrow \mathbb{N} \times \{0, \ldots, \lfloor\beta\rfloor\}$ satisfying the following: (1) if $\phi_i(m) = (r, a)$, then $d_\beta(1)_r = a$, and (2) for all $r \in \mathbb{N}$ with $d_\beta(1)_r > 0$, there is an $m \in \mathbb{N}$ such that $\phi_i(m) = (r, a)$ for some a.

Consider, for each $k \in \mathbb{N}$, the set $S_k = \{\phi_i(1), \ldots, \phi_i(k)\}$ and assume wlog. that $\phi_i(1), \ldots, \phi_i(k)$ are distinct. Observe that $S_k \subseteq S_{k+1}$. The unique positive solution, x_{S_k}, to $1 = \sum_{n\in S_k} d_\beta(1)_n x^{-n}$ is a computable real by Remark 2, and by the same remark, we can thus, given S_k, effectively find an element $p_k/q_k \in \mathbb{Q}$ such that $x_{S_k} - p_k/q_k < 2^{-k}$ and (for $k > 1$) $p_k/q_k \geq p_{k-1}/q_{k-1}, \ldots, p_1/q_1$. By Proposition 4, $x_{S_k} \leq x_{S_{k+1}}$ for all k, and the sequence $(p_k/q_k)_{k\in\mathbb{N}}$ is thus non-decreasing with $\sup_k(p_k/q_k) = x_S$.

As the operations for constructing the p_k/q_k from i are all computable, there exists a Turing machine with index i' halting on all inputs k such that $\phi_{i'}(k) = p_k/q_k$ for all $k \in \mathbb{N}$, whence x_S is left-computable. □

We have a similar correspondence between co-r.e. sets and right-computability:

Lemma 5: *Let, for all $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, the sets $\{n \in \mathbb{N} : d_\beta(1)_n = a\}$ be co-r.e. Then the unique positive solution, x_S, to $1 = \sum_{n\in\mathbb{N}} d_\beta(1)_n x^{-n}$ is a right-computable real number.*

Proof: Let, for each $a \in \{1, \ldots, \lfloor\beta\rfloor\}$, $i_{(a)}$ be the index of a Turing machine that enumerates the following set:

$$\mathbb{N} \setminus \{n \in \mathbb{N} : d_\beta(1)_n = a\} \quad (= \{n \in \mathbb{N} : d_\beta(1) \neq a\})$$

Set $d_k \triangleq \max_{1\leq n\leq k, 1\leq a\leq\lfloor\beta\rfloor}\{\phi_{i_{(a)}}(n)\}$. That is, d_k is the largest index that has occurred when the machines with indices $i_{(1)}, \ldots, i_{(\lfloor\beta\rfloor)}$ have all been run on inputs $1, \ldots, k$.

Consider, now, for each $r \in \mathbb{N}$ the (possibly empty) sets $B_r \triangleq \{j : \phi_{i_{(a)}}(n) = r, 1 \leq a \leq \lfloor\beta\rfloor, 1 \leq n \leq k\}$ and set $b(k)_r \triangleq \max(\{0, \ldots, \lfloor\beta\rfloor\} \setminus B_r)$. Thus, B_r is the set of values that we know $d_\beta(1)_r$ *cannot* assume after running the machines with indices $i_{(1)}, \ldots, i_{(\lfloor\beta\rfloor)}$ on the inputs $1, \ldots, k$.

Now observe that, for all r, we have $b(k)_r \geq d_\beta(1)_r$ and that, for each r, there exists an $N \in \mathbb{N}$ such that for all $k > N$ we have $b(k)_r = d_\beta(1)_r$[6].

Denote by x_k the unique positive solution of $1 = \sum_{n\in\mathbb{N}} b(k)_n x^{-n}$ and observe that for all $k \in \mathbb{N}$, $n > d_k$ implies that $b(k)_n = \lfloor\beta\rfloor$.

We shall establish an algorithm that, on input $k \in \mathbb{N}$, produces a number $p_k/q_k \in \mathbb{Q}$ such that $0 \leq p_k/q_k - x_k \leq 2^{-k}$ and such that for $k > 1$, we have $p_k/q_k \leq p_{k-1}/q_{k-1}, \ldots, p_1/q_1$. Thus, the algorithm produce a non-increasing sequence of non-negative rationals that converges to x_k.

Let M_k be the least natural number such that for all $n > M_k$, we have $b_r = \lfloor\beta\rfloor$. By the above comments, we have $M_k \leq d_k$ and can hence compute M_k. Split the sum $\sum_{n\in\mathbb{N}} b(k)_n x^{-n}$ as:

$$\sum_{n\in\mathbb{N}} b(k)_n x^{-n} = \sum_{n\leq M_k} b(k)_n x^{-n} + \sum_{n>M_k} \lfloor\beta\rfloor x^{-n}$$

[6]But not an N we can necessarily compute. Once again, we do not need to compute it to make the proof work; we simply need existence of N in order to show (non-effective) convergence of a particular sequence of rationals.

The rightmost term above is a geometric series with limit $\lfloor\beta\rfloor x^{-M_k}/(x-1)$. Thus, x_k is the largest positive solution of

$$\frac{x-1-\lfloor\beta\rfloor x^{-M_k}}{x-1} = \sum_{n\leq M_k} b_n x^{-n}$$

hence also the largest positive solution of

$$x^{M_k}(x-1-\lfloor\beta\rfloor x^{-M_k}) - (x-1)x^{M_k}\sum_{n\leq M_k} b_n x^{-n} = 0$$

But the left-hand side above is an integer polynomial of degree M_k+1, and as noted in Remark 2, x_k is thus a computable real. We can hence by the definition of computable reals *effectively* obtain a rational number $p_k/q_k \in \mathbb{Q}$ with $0 < x_k - p_k/q_k \leq 2^{-k}$. By construction and Proposition 4, if $1 \leq l \leq k$, we have $(b(l)_n)_{n\in\mathbb{N}} \preceq (b(k)_n)_{n\in\mathbb{N}}$ and hence $x_k \leq x_l$. We may thus choose p_k/q_k such that $p_k/q_k \leq p_{k-1}/q_{k-1}, \ldots, p_1/q_1$.

We can thus computably obtain p_k/q_k from the machines with indices c_j, and there is hence a Turing machine with index i' such that $\phi_{i'}(k) = p_k/q_k$ for all $k \in \mathbb{N}$.

For increasing k, increasingly longer prefixes of $d_\beta(1)$ occur as the prefixes of $(b_n)_{n\in\mathbb{N}}$, and as the sequence $(p_k/q_k)_{k\in\mathbb{N}}$ is non-increasing and satisfies $p_k/q_k - x_k \leq 2^{-k}$, we have $\inf_{k\in\mathbb{N}} \phi_{i'}(k) = x_S$, showing that x_S is right-computable. □

Corollary 6: *Let $\beta > 1$. If, for each $a \in \{1, \ldots, \lfloor\beta\rfloor\}$, the set $\{n : d_\beta(1)_n = a\}$ is recursive, then β is a computable real.*

Proof: By Lemmas 4 and 5. □

We would like to have a converse result, showing that if β is a computable real, then each of the sets $\{n : d_\beta(1)_n = a\}$ is recursive. This is contained in Lemma 7, the proof of which will need a number of ancillary results that we give in the following. We do not know whether any of these ancillary results can be improved to treat r.e. or co-r.e. sets.

Proposition 5: *If $\beta^k - \sum_{j=1}^{k-1} d_\beta(1)_j \beta^{k-j} \in \mathbb{Z}$ for some $k \geq 1$, then $d_\beta(1)_n = 0$ for all $n > k$, that is, the greedy expansion is finite.*

Proof: This is a straightforward consequence of the greedy algorithm for establishing $d_\beta(1)$. □

Consider a non-integral computable real number $\beta > 1$. If the greedy expansion $d_\beta(1)$ is finite, there obviously exists a Turing machine that, on input n, produces $d_\beta(1)_n$, and another Turing machine that, on input n, produces $d^*_\beta(1)$ (as the latter is periodic if $d_\beta(1)$ is finite and equals $d_\beta(1)$ otherwise). In case $d_\beta(1)$ is not finite, proving existence of a Turing machine computing the elements of $d_\beta(1)$ is somewhat harder and is contained in the following lemma.

Lemma 6: *Let $\beta > 1$ be a non-integral computable real such that $d_\beta(1)$ is not finite. Then there is a Turing machine with index i such that $\phi_i(n) = d_\beta(1)_n$ for all $n \in \mathbb{N}$.*

Proof: By Proposition 5, we have $\beta^n - \sum_{j=1}^{n-1} d_\beta(1)_j \beta^{n-j} \notin \mathbb{Z}$ for all $n \in \mathbb{N}$. Thus, for $n \in \mathbb{N}$, we have $\phi_i(n) = d_\beta(1)_n$ iff $\phi_i(n) < \beta^n - \sum_{j=1}^{n-1} d_\beta(1)_j \beta^{k-j} < \phi_i(n) + 1$.

As β is a computable real, there are Turing machines with indices i' and i'' halting on all inputs such that (1) $\beta = \lim_{n\to\infty} \phi_{i'}(n)$ and (2) for all $m \in \mathbb{N}$, $n \geq \phi_{i''}(m)$ implies $|\beta - \phi_{i'}(n)| < 2^{-m}$.

By standard results [41], the set of computable reals is an algebraic field, and we thus obtain that $\beta^n - \sum_{j=1}^{n-1} d_\beta(1)_j \beta^{n-j}$ is a computable real for all $n \in \mathbb{N}$. Again by standard results [41], the arithmetical operations of multiplication and addition are recursive, and there is thus a Turing machine with index l such that $\phi_l : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{Q}$ and, for all $n, m \in \mathbb{N}$, $|(\beta^n - \sum_{j=1}^{n-1} d_\beta(1)_j \beta^{n-j}) - \phi_l(n,m)| < 2^{-m}$.

Using l, we may construct an algorithm that does the following: For each $k \in \mathbb{N}$, run the machine with index l on input (n, j) for successively greater j until an $N \in \mathbb{N}$ is found such that for all $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, we have $|a - \phi_l(n, N)| > 2^{-(N-1)}$ (the fact that $\beta^n - \sum_{j=1}^{n-1} a_j \beta^{n-j} \notin \mathbb{Z}$ implies existence of such an N).

As $|(\beta^n - \sum_{j=1}^{n-1} d_\beta(1)_j \beta^{n-j}) - \phi_l(n, N)| < 2^{-N}$, we have $a < \phi_l(n, N) < a+1$ iff $a < \beta^n - \sum_{j=1}^{n-1} a_j \beta^{n-j} < a+1$, and the former condition can of course be brute-forced checked by going through all $a \in \{0, \ldots, \lfloor\beta\rfloor\}$.

By the algorithm above, there exists a Turing machine with index i such that, for all $k \in \mathbb{N}$, $\phi_i(n) = a$ iff $a < \beta^n - \sum_{j=1}^{n-1} d_\beta(1)_j \beta^{n-j} < a + 1$, concluding the proof. □

We now proceed to the following crucial lemma:

Lemma 7: *Let $\beta > 1$ and assume that β is a computable real number. Then, for all $a \in \{0, \ldots, \lfloor\beta\rfloor\}$, both $\{n : d_\beta(1)_n = a\}$ and $\{n : d^*_\beta(1)_n = a\}$ are recursive subsets of $\mathbb{N}$.*

Proof: Either $d_\beta(1)$ is finite or it is not[7].

If it is finite, then clearly $\{n : d_\beta(1)_n = 0\}$ is recursive (as the set contains all but a finite number of natural numbers). For $a > 0$, $\{n : d_\beta(1)_n = a\}$ is finite, hence recursive.

By definition, finiteness of $d_\beta(1)$ entails that $d^*_\beta(1)$ is periodic, hence can be generated by a finite automaton, hence is recursive.

If $d_\beta(1)$ is *not* finite, we have $d^*_\beta(1) = d_\beta(1)$, and the result follows from Lemma 6. □

We now prove our main result:

Theorem 4: *Let $\beta > 1$ be a non-integral real. Then β is a computable real number iff $\mathcal{L}(X_\beta)$ (resp. $\mathcal{L}(\tilde{X}_\beta)$) is a recursive subset of $\{0, \ldots, \lfloor\beta\rfloor\}^*$.*

Proof: "If" follows from Theorem 3 and Corollary 6. "Only if" follows from Lemma 7 and Theorem 3. Note that it follows immediately from the definitions that $\mathcal{L}(X_\beta)$ is recursive iff $\mathcal{L}(\tilde{X}_\beta)$ is recursive. □

As every algebraic number is computable [41, 43], we immediately obtain:

Corollary 7: *If $\beta > 1$ is a (real) algebraic number, then $\mathcal{L}(X_\beta)$ is recursive .*

As many well-known transcendental numbers β are expressible in terms of series expansions with predictable rate of convergence, their β-shifts will have recursive language.

Example 2. The π-shift has recursive language, as has the e-shift and the γ-shift (where γ is the Euler-Mascheroni constant).

As negative examples, we have the following two well-known variations on a common theme:

[7] This use of the Principle of the Excluded Middle (PEM) is the essential non-constructive part of our main theorem on the correspondence between $d_\beta(1)$ and β. We are applying the PEM to the undecidable problem of whether a (computable) infinite sequence of bits only has a finite number of ones. Note that the lemma illustrates the difference between merely proving that certain sets are recursive and proving that they are "uniformly computable in some other set", that is, that we can effectively obtain decision procedures for them, given oracles to other sets.

Example 3. The β_s-shift corresponding to Specker's constant

$$\beta_s = 1 + \sum_{n \in \emptyset'} 2^{-n}$$

has non-recursive language, as β_s is not a computable real [36].

Let U be any universal prefix-free Turing machine and let $H_U = \{x \in \{0,1\}^* : U \text{ halts on input } x\}$. Then Chaitin's (U-)constant is the real number

$$\Omega_U = \sum_{x \in H_U} 2^{-|x|}$$

and Ω_U is not a computable real [20]; hence the Ω_U-shift has non-recursive language.

6 Non-constructiveness of the correspondence result for β-shifts is unavoidable

The correspondence theorem for β-shifts was proven by means of two intermediate results: The first result stated that there is an *algorithm* for turning a decision procedure for $\mathcal{L}(X_\beta)$ or $\mathcal{L}(\tilde{X}_\beta)$ into a computable name of the computable real β (that is, the lemma has a constructive proof). The second result had a proof that stated that it is impossible for a computable real *not* to have a β-shift with recursive language; thus, the proof established existence by means of a double negation and was not constructive.

The essential non-constructive technicality of the proof was a flagrant use of the Principle of the Excluded Middle. It is offhand not evident that this should be necessary: A more careful argument could supposedly have yielded a constructive proof that would have supplied an algorithm to convert a computable name of β into a decision procedure for $\mathcal{L}(X_\beta)$.

In this section, we show that *no such constructive proof exists*. For ease of exposition, we consider reals $1 < \beta < 2$; for these, we have $\mathcal{L}(X_\beta) \subseteq \{0,1\}^*$ and thus $d_\beta(1)$ $d^*_\beta(1)$ are thus sequences of bits.

For the W-β-shift and for the set of βs with non-finite $d_\beta(1)$, we can give a surprisingly easy argument employing the Golden ratio showing that existence of a constructive proof would imply solvability of the Halting Problem. We have:

Theorem 5: *There is no program that, on input a computable name of a computable real $1 < \beta < 2$, produces the sequence of bits in $d_\beta(1)$.*

Formally: There is no partial recursive function $\zeta : \mathbb{N} \longrightarrow \mathbb{N}$ such that if $\phi_i : \mathbb{N} \longrightarrow \mathbb{Q}$ is a computable name of a computable real $1 < \beta < 2$, then $i \in dom(\zeta)$ and $\phi_{\zeta(i)} : \mathbb{N} \longrightarrow \{0,1\}$ is a total recursive function satisfying $\phi_{\zeta(i)}(n) = 1$ iff $d_\beta(1)_n = 1$.

Proof: Set $\Phi \triangleq (1+\sqrt{5})/2$ and note that we have $d_\Phi(1) = 110^\omega$. For any $1 < \beta < 2$, we have $d_\beta(1)_1 = 1$ and $d_\beta(1)_2(\beta) = \lfloor \beta^2 - \beta \rfloor$. Thus, $d_\beta(1)_2 = 1$ iff $1 = \lfloor \beta^2 - \beta \rfloor$ iff $1 \leq \beta^2 - \beta$ iff $\Phi \leq \beta$. Thus, if ζ existed, we could feed it any computable name of a computable real $1 < \beta < 2$ and obtain an answer to the question "is $\Phi \leq \beta$?", contradicting Proposition 2. □

Corollary 8: *There is no program that, on input a computable name of a computable real $1 < \beta < 2$, produces a decision procedure for $\mathcal{L}(\tilde{X}_\beta)$.*

Proof: Assume otherwise. The assumption then implies that given $n \in \mathbb{N}$ we can effectively obtain the set $\mathcal{L}(\tilde{X}_\beta) \cap \{0,1\}^n$. The lexicographically greatest element of this finite set is the prefix of $d_\beta(1)$ of

length k. But then we can obviously construct an algorithm that produces the sequence of bits of $d_\beta(1)$, contradicting Theorem 5. □

For the proper β-shift, we would expect a result corresponding to Theorem 5 to hold. Indeed, the two sets X_β and $\tilde{X}_\beta$ differ only when $d_\beta(1)$ is finite. The problem of asserting whether a Turing machine computes a sequence ending in an infinite sequence of zeros is obviously non-recursive. So, intuitively, computing the sequence of digits of $d^*_\beta(1)$ should not be *easier* than computing the sequence of digits of $d_\beta(1)$.

Unfortunately, this ad hoc argument does not lend itself to formalisation, and we are forced to take a more roundabout approach:

Theorem 6: *There is no program that, on input a computable name of a computable real $1 < \beta < 2$, produces the sequence of digits of $d^*_\beta(1)$.*

Formally: There is no partial recursive function $\zeta : \mathbb{N} \longrightarrow \mathbb{N}$ such that if $\phi_i : \mathbb{N} \longrightarrow \mathbb{Q}$ is a computable name of a computable real $1 < \beta < 2$, then $i \in dom(\zeta)$ and $\phi_{\zeta(i)} : \mathbb{N} \longrightarrow \{0,1\}$ is a total recursive function satisfying $\phi_{\zeta(i)}(n) = 1$ iff $d^*_\beta(1)_n = 1$.

Proof: For $n \in \mathbb{N}$, let c_n be the element of $\{0,1\}^{\mathbb{N}}$ having a 1 in the nth place, and 0 in all other places.

Consider the set $E \subseteq \{0,1\}^{\mathbb{N}}$ consisting of all infinite bit strings on the form $11c_n$. For each element $11c_n$, of E, we have $\sigma^l(11c_n) \leq_{\text{lex}} 11c_n$ for all $l \in \mathbb{N}$, whence by Theorem 2, there exists a real number $1 < \beta_n < 2$ with $11c_n = d_{\beta_n}(1)$. Note that by definition of $d_\beta(1)$, we have $d_\beta(1) = 11c$ for some $c \in \{0,1\}^{\mathbb{N}}$.

Set $\Phi \triangleq (1+\sqrt{5})/2$. As β_n is a solution of $1 = x^{-1} + x^{-2} + x^{-n-2}$, we have $1 = \beta_n^2 - \beta_n - \beta_n^{-n}$.

We have (1) $\Phi < \beta_n$ for all n, (2) $\beta_j > \beta_n$ for $j < n$, and (3) $\Phi = \inf_n \beta_n$ (as $1 = \Phi^{-1} + \Phi^{-2}$).

Let $\phi_i : \mathbb{N} \longrightarrow \{0,1\}$ be any total recursive function. We will construct a name p (with $\phi_p : \mathbb{N} \longrightarrow \mathbb{Q}$) of a computable real by setting $\phi_p(n) \triangleq \phi_{p'}(2n+2)$ for an index p' of a Turing machine that does the following:: For each $j \in \mathbb{N}$, if $\phi_i(1) = \cdots = \phi_i(j) = 0$, then let $\phi_{p'}(j)$ be a $p_j/q_j \in \mathbb{Q}$ with $0 < p_j/q_j - \beta_j \leq \Phi^{-j}$. If $\phi_i(j) = 1$ and $\phi_i(1) = \cdots = \phi_i(j-1) = 0$, let, for all $l \geq j$, $\phi_{p'}(l)$ be a $p_l/q_l \in \mathbb{Q}$ such that $p_l/q_l - \beta_j < 2^{-l}$.

Observe that ϕ_p is a total recursive function $\phi_p : \mathbb{N} \longrightarrow \mathbb{Q}$ and that the algorithm above obtains p effectively from i.

Split on cases according to whether there exists a $n \in \mathbb{N}$ such that $\phi_i(n) = 1$:

- Assume that there is no such n.
  As $1 = \Phi^2 - \Phi$, we obtain

  $$\Phi^2 - \Phi = \beta_n^2 - \beta_n - \beta_n^{-n}$$

  and thus

  $$(\Phi - \beta_n)(\Phi + \beta_n) = \Phi - \beta_n - \beta_n^{-n}$$

  whence, by $1 < \Phi < \beta_n$,

  $$|\Phi - \beta^n| = \frac{\beta_n^{-n}}{\Phi - \beta_n - 1} \leq \beta^{-n} \leq \Phi^{-n}.$$

  We then have:

  $$\begin{array}{rcl} |\phi_{p'}(j) - \Phi| & \leq & |\phi_{p'}(j) - \beta_j| + |\beta_j - \Phi| \\ & \leq & \Phi^{-j} + \Phi^{-j} \\ & \leq & 2\Phi^{-j} \end{array}$$

  Hence, $|\phi_p(j) - \Phi| = |\phi_{p'}(2j+2) - \Phi| \leq 2\Phi^{-2j-2} = (\Phi^2)^{-j} < 2^{-j}$ for all $j \in \mathbb{N}$. That is, p is a computable name of Φ.

- Assume that there exists a n with $\phi_i(n) = 1$.

  Straightforward calculations for $j \leq n$ yield:

$$\beta_n^2 - \beta_n - \beta_n^{-n} = \beta_j^2 - \beta_j - \beta_j^{-j}.$$

  That is,

$$(\beta_n - \beta_j)(\beta_n + \beta_j) = \beta_n - \beta_j + \beta_n^{-n} - \beta_j^{-j}$$

  whence, by $1 < \beta_n < \beta_j$,

$$|\beta_j - \beta_n| = \frac{\beta_j^{-j} - \beta_n^{-n}}{\beta_n + \beta_j - 1} \leq \beta_j^{-j}$$

  Thus, we have $|\phi_p(j) - \beta_n| = |\phi_{p'}(2j+2) - \beta_n| \leq |\phi_{p'}(2j+2) - \beta_{2j+2}| + |\beta_{2j+2} - \beta_n| \leq 2^{-(2j+2)} + \beta_{2j+2}^{-2j-2} \leq 2^{-(2j+2)} + \Phi^{-(2j+2)} < 2^{-(2j+2)} + 2^{-(j+1)} \leq 2^{-j}$. Hence, p is a computable name of β_n.

Thus, p is a computable name of a computable real number.

Now, for all $i \in \mathbb{N}$ denote by α_i the computable real whose name is p (as obtained from i above), and assume, for the purpose of contradiction, that the function ζ of the theorem existed. Given i, we could then effectively construct p, then use $\zeta(p)$ to ascertain whether $d^*_{\alpha_i}(1)_2 = 0$.

But $d^*_\Phi(1) = (10)^\omega$ and $d_{\beta_n}(1) = 11c_n$ for all n. Hence, $d_{\alpha_i}(1)_2 = 0$ implies $\alpha_i = \Phi$ which implies $\phi_i(j) = 0$ for all $j \in \mathbb{N}$. And $d^*_{\alpha_i}(1)_2 = 1$ implies $\alpha_i = \beta_n$ which implies existence of some $n \in \mathbb{N}$ with $\phi_i(n) = 1$.

We could thus, for an arbitrary total function ϕ_i decide whether there exists a $n \in \mathbb{N}$ such that $\phi_i(n) = 1$. This contradicts Proposition 1. □

Corollary 9: *There is no program that, on input a computable name of a computable real* $1 < \beta < 2$, *produces a decision procedure for* $\mathcal{L}(X_\beta)$.

Proof: Assume otherwise. For each natural number n, the assumption implies that we can obtain, by a computable procedure, the finite set $\mathcal{L}(X_\beta) \cap \{0,1\}^n$. The lexicographically greatest element of this finite set is the prefix of $d^*_\beta(1)$ of length n and can obviously be found in finite time. But then we have an algorithm that produces the sequence of bits of $d^*_\beta(1)$, contradicting Theorem 6. □

Thus, while the fact that β is a computable real allows us to conclude that $\mathcal{L}(X_\beta)$ is a recursive set, we *cannot* on the basis of a Turing machine computing β, *construct* a Turing machine that lets us decide $\mathcal{L}(X_\beta)$, even though we know that such a machine exists.

7 Unresolved questions

While we have established correspondences between computability of real numbers β and decidability of the languages of their associated β-shifts, some pertinent questions remain:

1. In analogy to the extension of the notion of recursive set to the full arithmetical hierarchy [27], it is possible to extend the notion of computable real to an *arithmetical hierarchy of reals* [42]. The class of β-shifts with languages at level n of the arithmetical hierarchy comprise precisely those shifts for which β is at level n in the arithmetical hierarchy of reals. The proofs of our results are so generic that they can be extended without much effort to show that the β-shifts having languages at the nth level of the arithmetical hierarchy are precisely those where β is a real number at the

nth level of the arithmetical hierarchy of reals. For purposes of exposition, this paper only concerns results at the very first level of both hierarchies: Recursive sets and computable reals. Full details concerning sets and reals higher in the hierarchies may be found in the companion technical report [33].

Apart from the arithmetical hierarchy of reals, there is also an analytical such hierarchy and a correspondingly-defined classical notion of analytical sets [27]. We conjecture that our correspondence theorems can be lifted to this settings.

2. For subrecursive classes of sets and their realizability as the languages of β-shifts, the attention has so far focused on the Chomsky hierarchy (hence on regular, context-free and context-sensitive languages), and a number of interesting results for these are known [5, 17]. For other subrecursive classes that have been studied in computer science, not much is known. The question of β-shifts having languages complete for complexity classes like P, NP, PSPACE and so forth has been briefly investigated [34], but there is still a plethora of sets investigated within computational complexity theory that might conceivably be realizable as the languages of interesting classes of reals.

   The same observation goes for other classes of shifts. For instance, Hertling and Spandl have investigated computability theoretical properties of the languages of gap shifts [13]. Investigating subrecursive classes in that setting could yield interesting results as well.

8 Acknowledgments

The author extends his thanks to Morten Fjord-Larsen for enlightening discussions and to Søren Debois, Tom Hvitved, Ken Friis Larsen, and anonymous referees of both the current and of earlier versions of the paper for many style-improving comments. In addition, the author expresses his gratitude to the participants of the 3gERP 2007 and 2008 seminars; bizarrely, this paper would never have existed without them.

References

[1] B. Adamczewski and Y. Bugeaud. Dynamics for β-shifts and diophantine approximation. *Ergodic Theory and Dynamical Systems*, 27:1–17, 2007.

[2] M. Barnsley. *Fractals Everywhere*. Morgan Kaufmann, 1993.

[3] V. Berthé and M. Rigo. Abstract numeration systems and tilings. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS '05)*, volume 3618 of *Lecture Notes in Computer Science*, pages 131–143. Springer-Verlag, 2005.

[4] E. Bishop and D. Bridges. *Constructive Analysis*, volume 279 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1985.

[5] F. Blanchard. β-expansions and symbolic dynamics. *Theoretical Computer Science*, 65:131–141, 1989.

[6] D. Bridges and F. Richman. *Varieties of Constructive Mathematics*, volume 97 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, Cambridge, UK, 1987.

[7] K. Dajani and M. de Vries. Measures of maximal entropy for random β-expansions. *Journal of the European Mathematical Society*, 1:51–68, 2005.

[8] K. Dajani and C. Kraaikamp. *Ergodic Theory of Numbers*, volume 29 of *The Carus Mathematical Monographs*. The Mathematical Association of America, 2002.

[9] C. Frougny, J.-P. Gazeau, and R. Krejcar. Additive and multiplicative properties of point sets based on beta-integers. *Theoretical Computer Science*, 303(2-3):491–516, 2003.

[10] C. Frougny and B. Solomyak. Finite beta-expansions. *Ergodic Theory and Dynamical Systems*, 12:713–723, 1992.

[11] C. Frougny and A. Surarerks. On-line multiplication in real and complex base. In *Proceedings of the 16th IEEE Symposium on Computer Arithmetic (ARITH 16)*, pages 212–219. IEEE Computer Society Press, 2003.

[12] G. Hansel, D. Perrin, and I. Simon. Compression and entropy. In *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science (STACS '92)*, volume 577 of *Lecture Notes in Computer Science*, pages 515–528. Springer-Verlag, 1992.

[13] P. Hertling and C. Spandl. Computability theoretic properties of the entropy of gap shifts. *Fundamenta Informaticae*, 83(1–2):141–157, 2008.

[14] M. Hochman and T. Meyerovitch. A characterization of the entropies of multidimensional shifts of finite type. *Annals of Mathematics*, 2009. To appear.

[15] F. Hofbauer. β-shifts have unique maximal measure. *Monatshefte für Mathematik*, 85:189–198, 1978.

[16] S. Ito and Y. Takashi. Markov subshifts and realization of β-expansions. *Journal of the Mathematical Society of Japan*, 26:33–55, 1974.

[17] K. Johnson. *Beta-Shift Dynamical Systems and Their Associated Languages*. PhD thesis, University of North Carolina, 1999.

[18] N. Jones. *Computability and Complexity from a Programming Perspective*. The MIT Press, 1997.

[19] K.-I. Ko and D.-Z. Du. *Theory of Computational Complexity*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., New York, 2000.

[20] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Graduate Texts in Computer Science. Springer-Verlag, 1997.

[21] D. Lind. The entropies of topological Markov shifts and a related class of algebraic integers. *Ergodic Theory and Dynamical Systems*, 4:283–300, 1984.

[22] D. Lind and B. Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1995.

[23] M. Morse. Recurrent geodesics on a surface of negative curvature. *Transactions of the American Mathematical Society*, 22:84–110, 1921.

[24] P. Odifreddi. *Classical Recursion Theory, volume I*, volume 129 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1989.

[25] W. Parry. On the β-expansion of real numbers. *Acta Math. Acad. Sci. Hung.*, 11:401–416, 1960.

[26] A. Rényi. Representations for real numbers and their ergodic properties. *Acta Math. Acad. Sci. Hung.*, 8:477–493, 1957.

[27] H. Rogers Jr. *Theory of Recursive Functions and Effective Computability*. The MIT Press, paperback edition, 1987.

[28] K. Schmidt. On periodic expansions of Pisot numbers and Salem numbers. *Bulletin of the London Mathematical Society*, 12:269–278, 1980.

[29] N. Sidorov. Almost every number has a continuum of beta-expansions. *American Mathematic Monthly*, 110:838–842, 2003.

[30] N. Sidorov. Arithmetic dynamics. In *Topics in Dynamics and Ergodic Theory*, volume 310 of *London Mathematical Society Lecture Notes Series*, pages 145–189. London Mathematical Society, 2003.

[31] J. Simonsen. On beta-shifts having arithmetical languages. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS 2005)*, volume 3618 of *Lecture Notes in Computer Science*, pages 757–768. Springer-Verlag, 2005.

[32] J. Simonsen. On the computability of the topological entropy of subshifts. *Discrete Mathematics and Theoretical Computer Science*, 8:83–96, 2006.

[33] J. Simonsen. Beta-shifts, their languages and computability. Technical report, Department of Computer Science, University of Copenhagen (DIKU), 2008.

[34] J. Simonsen. On the computational complexity of languages of general symbolic dynamical systems and beta-shifts. *Theoretical Computer Science*, 2009. To appear.

[35] C. Spandl. Computing the topological entropy of shifts. *Electronic Notes in Theoretical Computer Science*, 167:131–155, 2007.

[36] E. Specker. Nicht konstruktiv beweisbare Sätze der Analysis. *Journal of Symbolic Logic*, 14:145–158, 1949.

[37] Y. Takahashi. Shift with free orbit basis and realization of one-dimensional maps. *Osaka Journal of Mathematics*, 20:599–629, 1983.

[38] A. Turing. On computable numbers with an application to the "entscheidungsproblem". *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.

[39] A. Turing. On computable numbers with an application to the "entscheidungsproblem". a correction. *Proceedings of the London Mathematical Society*, 43(2):544–546, 1937.

[40] P. Walters. *An Introduction to Ergodic Theory*, volume 79 of *Graduate Texts in Mathematics*. Springer-Verlag, 1981.

[41] K. Weihrauch. *Computable Analysis: An Introduction*. Springer-Verlag, 1998.

[42] X. Zheng and K. Weihrauch. The arithmetical hierarchy of real numbers. *Mathematical Logic Quarterly*, 47(1):51–65, 2001.

[43] M. Ziegler and V. Brattka. Computability in linear algebra. *Theoretical Computer Science*, 326:187–211, 2004.