On the Computability of the Topological Entropy of Subshifts

Jakob Grue Simonsen

Department of Computer Science, University of Copenhagen (DIKU)
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

We prove that the topological entropy of subshifts having decidable language is uncomputable in the following sense: For no error bound less than 1/4 does there exists a program that, given a decision procedure for the language of a subshift as input, will approximate the entropy of the subshift within the error bound. In addition, we prove that not only is the topological entropy of sofic shifts computable to arbitary precision (a well-known fact), but all standard comparisons of the topological entropy with rational numbers are decidable.

Keywords: symbolic dynamics, shift spaces, subshifts, computability, recursion theory

1 Introduction

Dynamical systems are among the most studied objects in branches of mathematics, computer science, and physics. To each dynamical system can be attributed several quantities that gauge the “complexity” of the system, one of the simplest and most important of these being the socalled *topological entropy*.

For two well-known species of dynamical systems, namely cellular automata [9] and iterated piecewise affine functions [12], the topological entropy is known to be computably unapproximable in the sense that for sufficiently low $\epsilon \in \mathbb{R}_+$, no program can yield a rational number p/q in finite time such that the topological entropy differs from p/q by at most ϵ.

For other dynamical systems, e.g. Turing machines and Wang tilings of the plane, the problem of whether the topological entropy of a given system is zero (or, in general, equal to some fixed real number) is undecidable [7], but it is as yet unknown whether the topological entropy of these systems is computable in the “approximation” sense above.

In this paper, we study computability and decidability issues related to another standard class of dynamical systems, the *subshifts* (aka. *symbolic dynamical systems*, *shift spaces*, or just *shifts*). These systems are a staple of introductory courses in dynamical systems [19, 13], and have come under scrutiny in a variety of settings, particularly in fractal geometry [2], and in data compression [8]. The *shift operation* either moves a bi-infinite string one place to the left (two-sided shift), or “chops off” the leftmost character of a right-infinite string (one-sided shift). Subshifts are sets of infinite strings that are stable under the shift operation. In order to restrict ourselves to well-behaved systems, we consider subshifts with decidable *language*, i.e. where there is a program that decides, for each *finite* string, whether it occurs as a substring of an infinite string in the subshift.

Our results are:

1. The topological entropy of subshifts having decidable language is computably unapproximable: For no $\epsilon \in \mathbb{Q} \cap (0, \log(2)/4)$ does there exist a program that, given a decision procedure known to decide the language of a subshift over the alphabet $\{0, 1\}$ will compute the topological entropy with error $< \epsilon$. By considering larger alphabets than $\{0, 1\}$, this result may be extended to any $\epsilon \in \mathbb{Q} \cap [0; \infty)$, and the result holds for several restricted kinds of subshifts. In the lingo of formal language theory, the result states that the entropy rate of elements of the class of languages that are *decidable, irreducible, factorial, and extensible* is computably unapproximable.

2. Not only is the topological entropy of sofic subshifts computable in the sense that there is a program that, when given a right-resolving graph representation of the shift as input, the program will approximate the topological entropy to any desired precision (a well-known fact), but comparison under $<, \leq, =, \geq, >$ of the topological entropy with any rational number is also *decidable*.

Thus, two notions of computability of topological entropy are considered in this paper:

(1) approximation within some rational error ϵ, and

(2) comparison of the topological entropy with rational numbers.

We note that the latter notion is stronger in the sense that if there were an effective procedure P allowing us to compare the entropy with *any* rational number, we could obtain arbitrarily good ϵ-approximations to the entropy by bounding the entropy from above and below with successively good approximations by quering P (“is the entropy ≥ 0”, “if so is the entropy $\geq 1/2$”, “if so, is it $\geq 3/4$”), and so on. Thus, the positive result 2 entails as a corollary the well-known result that the topological entropy of sofic shifts is effectively approximable to arbitrary precision by rational numbers.

Conversely, having effective approximability, even to any specified error bound ϵ would not necessarily yield decision procedures allowing comparison of the entropy with arbitrary rationals (see the remarks after Definition 14). In addition, our negative result 1 above renders this point void in any case.

Considering shifts in terms of decision procedures for their languages may seem like “cheating” since the “finite description” of a shift with decidable language that we use as input to a potential procedure for approximating the entropy of the shift will be a full-fledged program. As the input may thus be any program in a (Turing-complete) programming language, the class of “allowed” inputs is clearly undecidable: Namely the class of *total* programs that decide languages of shift spaces.

Contrast this with certain other classes of dynamical systems for which the entropy is known to be uncomputable, e.g. cellular automata, where it is decidable whether the input (a finite description of a cellular automaton) is of the correct form.

The specification of shift spaces with decidable languages used in this paper is akin to that in (Type 2) computable analysis [20] where the input to computable functions are (programs computing approximations to) computable reals and it is undecidable whether a given program provides such an approximation. There, the computable functions are not required to “check” the validity of their input, but merely to act correctly on valid input and behave in an unspecified manner on invalid input.

As there is no canonical way (in fact, at the time of writing no way at all) to represent shift spaces with decidable languages in such a way that it is decidable whether the input is correct, we regard the inapproximability result given in this paper as relevant. We further believe that future efforts directed at demarcating the limits of computability for shift spaces should accordingly focus on suitable subrecursive classes, e.g. shifts with primitive recursive languages.

2 Preliminaries

The reader is assumed to have a basic background in computability theory and basic familiarity with algebra, linear and otherwise. Assuming the Church-Turing thesis, we use the generic word "program" which the reader is assumed to replace with "(index of a) partial (or total) recursive function" or "Turing machine" at her leisure, *mutatis mutandis*. The Greek letters ϕ, ψ and ξ denote partial recursive functions; we overload ϕ_i to denote the ith partial recursive function on C to D where C and D are any of the sets $\mathbf{2}$ $(= \{0,1\})$, $\mathbb{N} = \{1,2,\ldots\}$, $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$, $\mathbb{Z} = \{\ldots,-1,0,1,\ldots\}$, or $\mathbb{Q} = \{p/q \mid p,q \in \mathbb{Z}, q \neq 0\}$. Note for later use that comparisons $<, =, >$ in any of these sets are decidable. Throughout the text, $U(x)$ is a universal Turing machine such that $U(i)$ simulates the ith inputless Turing Machine. Σ denotes a finite alphabet of symbols, and λ denotes the empty string unless otherwise stated.

Definition 1 *The set of all bi-infinite binary sequences is denoted by $\Sigma^{\mathbb{Z}}$, the set of all (right-)infinite binary sequences by $\Sigma^{\mathbb{N}}$. α, β and γ range over (bi- or right-)infinite binary sequences. If $\alpha \in \Sigma^{\mathbb{Z}}$ and $k, m \in \mathbb{Z}$ with $k \leq m$, then $\alpha_{[k,m]}$ is the binary string $\alpha(k) \cdots \alpha(m)$ of length $m - k + 1$, while $\alpha_{\leq k} = \cdots \alpha(k-1) \cdot \alpha(k)$, and $\alpha_{\geq k} = \alpha(k) \cdots \alpha(k+1) \cdots$. If $\alpha \in \Sigma^{\mathbb{N}}$, the concepts are carried over* mutatis mutandis *(if $\alpha \in \Sigma^{\mathbb{N}}$, then $\alpha_{\leq k} = \alpha(1) \cdots \alpha(k)$).*

Definition 2 *The* shift map *on $\Sigma^{\mathbb{Z}}$ (and $\Sigma^{\mathbb{N}}$) is defined by $(\sigma(\alpha))(i) \triangleq \alpha(i+1)$ for all $i \in \mathbb{Z}$ (resp. $i \in \mathbb{N}$). A set $X \subseteq \Sigma^{\mathbb{Z}}$ (or $\Sigma^{\mathbb{N}}$) is said to be* shift-closed *if $\sigma(X) \subseteq X$, and* shift-invariant *if $\sigma(X) = X$.*

Definition 3 *The sequence metric on $\Sigma^{\mathbb{Z}}$ is defined by $d(\alpha, \alpha) = 0$, and, if $\alpha \neq \beta$, $d(\alpha, \beta) = 2^{-n}$ where $n \in \mathbb{N}_0$ is the least number such that $\alpha_{-n} \neq \beta_{-n}$ or $\alpha_n \neq \beta_n$.*

Definition 4 *A* two-sided subshift *(aka. a* symbolic dynamical system*) is a topologically closed (wrt. the sequence metric on $\Sigma^{\mathbb{Z}}$), shift-invariant subset $X \subseteq \Sigma^{\mathbb{Z}}$. A* one-sided *subshift is a topologically closed (wrt. the standard sequence metric on $\Sigma^{\mathbb{N}}$), shift-closed subset $X \subseteq \Sigma^{\mathbb{N}}$.*

The term "Topologically closed" refers to the fact that every sequence of elements of the subset X that converges in the standard sequence metric on $\Sigma^{\mathbb{Z}}$ or $\Sigma^{\mathbb{N}}$ must have its limit in X; in the greater context of symbolic dynamical systems, this ensures that the shift is a *compact* subset of $\Sigma^{\mathbb{Z}}$ or $\Sigma^{\mathbb{N}}$.

An alternative characterization of subshifts can be given in terms of sets of "forbidden" words:

Definition 5 *Let $F \subseteq \Sigma^*$. The* two-sided shift generated by *F, denoted X_F, is the subset of $\Sigma^{\mathbb{Z}}$ containing exactly those α such that, for all $i, j \in \mathbb{Z}$ with $i \leq j$, it is the case that $\alpha_{[i,j]} \notin F$. The one-sided language generated by F is defined with the obvious changes. A* shift of finite type *is a subshift X such that there is a* finite *F with $X = X_F$.*

Lemma 6 *X is a subshift iff there exists an $F \subseteq \Sigma^*$ such that $X = X_F$.*

Proof: Standard. See e.g. [13, Ch. 6]. □

For characterization of subshifts in terms of *finite* strings, the concept of *language of a subshift* is needed:

Definition 7 *The* language *of a two-sided subshift X is the set of finite binary strings $\alpha_{[i,j]}$ where $i, j \in \mathbb{Z}$, $i \leq j$, and $\alpha \in X$. The language of a one-sided subshift is defined as above, but with $i \geq 1$. In both cases, we denote the language by $\mathcal{L}(X)$.*

One can prove that $L \subseteq \Sigma^*$ is the language of a subshift iff L has the well-known properties of being *factorial* and *extensible* [5, 4].

Definition 8 *A langage $L \subseteq \Sigma^*$ is said to be* factorial *if all substrings of all elements of L are also in L, i.e. if $w \cdot x \cdot v \in L$ with $w, x, v \in \Sigma^*$, then $x \in L$. Furthermore, L is said to be* extensible *if, for any $x \in L$, there are $v, w \in \Sigma$ such that $v \cdot x \in L$ and $x \cdot w \in L$.*

There is a canonical set of forbidden words for each subshift X, defined in terms of $\mathcal{L}(X)$.

Definition 9 *The set of* minimal forbidden words *of a subshift X, denoted $\mathcal{MF}(X)$, is the set of words $w \in \Sigma^*$ such that $w = w_1 \cdots w_n$ is forbidden, but both $w_1 \cdots w_{n-1} \in \mathcal{L}(X)$ and $w_2 \cdots w_n \in \mathcal{L}(X)$.*

An important class of subshifts consists of the irreducible shifts:

Definition 10 *A subshift X is* irreducible *(aka.* transitive*) if, for all $a, b \in \mathcal{L}(X)$, there exists $w \in \mathcal{L}(X)$ such that $a \cdot w \cdot b \in \mathcal{L}(X)$.*

2.1 Topological Entropy

We now define our central object of study:

Definition 11 *Let X be a subshift. The* topological entropy *of X, denoted $h_{top}(X)$ is the quantity*

$$h_{top}(X) \triangleq \lim_{n \to \infty} \left(\frac{\log(|\mathcal{L}(X) \cap \Sigma^n|)}{n} \right)$$

where the logarithm is usually taken to base 2.

For subshifts, the limit above will always be well-defined (see e.g. [13, Example 6.3.4]). For arbitrary languages $L \subseteq \Sigma^*$, the quantity $\limsup_{n \to \infty} \frac{\log(|L \cap \Sigma^n|)}{n}$ (which coincides with the topological entropy in case $L = \mathcal{L}(X)$ for a subshift X) is variously known as the *entropy rate*, the *Kolmogorov entropy*, or the (Shannon) *channel capacity*, but in general the limsup will not be a limit.

2.2 Sofic Shifts

An important class of subshifts are the *sofic shifts*, i.e. those subshifts whose languages are generated by (bi-)infinite paths in finite digraphs (also called *cover automata*):

Definition 12 *A* labelled graph *G is a pair $((V,E),L)$ where (V,E) is a digraph and $L : E \longrightarrow \Sigma$ is a function mapping edges to symbols. The* edge shift *of G is the subset X_G of $\Sigma^{\mathbb{Z}}$ containing all bi-infinite strings obtained by starting from nodes in V, tracing a bi-infinite path in the graph and noting the edge label in each step of the path. A (two-sided) subshift is called a (two-sided)* sofic shift *if it is the edge shift of some finite graph G. One-sided sofic shifts contain the right-infinite strings obtained by starting from nodes and following directed paths starting from such nodes.*

It is easy to see that a sofic shift is indeed a subshift, and that every shift of finite type is sofic. Also, using standard methods from automata theory, we may wlog. assume that the underlying (undirected) graph of a digraph is connected. Recall that the *spectral radius* of an $n \times n$ matrix A with coefficients in $\mathbb{Z}$ is $\max_{1 \leq i \leq n}\{|\lambda_i|\}$ where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A counted with (algebraic) multiplicities. For a digraph with n nodes, the *adjacency matrix* is the $n \times n$ matrix A such that a_{ij} is the number of directed arrows from node i to node j. Finally, recall that a labelled graph is said to be *right-resolving* if, for each node N, the edges starting at N carry different labels.

The following theorem is standard (though often proved only for the special case of irreducible shifts). See e.g. [13, Ch. 4].

Theorem 13 *If G is a finite, right-resolving labelled digraph with n nodes, then $h_{top}(X_G)$ is $\log(\lambda_{max})$ where λ_{max} is the spectral radius of the adjacency matrix, A, of G, i.e. is the maximal modulus of the solutions of the polynomial $det(xI - A)$ where I is the identity matrix of dimension $n \times n$.*

It is a standard result that the largest eigenvalue of a square matrix with non-negative real entries (the *Perron eigenvalue*) is a non-negative real number, hence equals its modulus [13, Ch. 4].

2.3 Computable Real and Complex Numbers

The topological entropy will not in general be a rational number. To properly treat computable approximation of such numbers, we shall thus need the concept of *computable reals* [18, 10, 20].

Definition 14 *A* computable name *is a total recursive function $\phi : \mathbb{N} \longrightarrow \mathbb{Q}$ such that $|\phi(m) - \phi(n)| < 2^{-m}$ for all $m, n \in \mathbb{N}$ with $n > m$. A real number x is said to be* computable *if there is a computable name that converges to x. The set of computable reals is denoted by $\mathbb{R}_c$.*

Any program working on computable reals takes computable *names* as input and outputs computable names. A straightforward consequence of this is that the comparison operators $<$, $\leq$, $>$, $\geq$ and $=$ are undecidable on $\mathbb{R}_c \times \mathbb{R}_c$, and stay undecidable even when one of the arguments is fixed [20]. However, for a fixed $a \in \mathbb{R}_c$, it follows straightforwardly from the definition of computable real that there is a program that halts with output 'yes' if given input $x \in \mathbb{R}_c$ with $x > a$, and never halts with output 'yes' if $x \leq a$ (but may either loop infinitely or halt with 'no' in this case).

Computable complex numbers are defined in the obvious way:

Definition 15 *A* computable complex number *is a pair (a, b) of computable reals (representing the real and imaginary parts of a complex number, resp.). The set of computable complex numbers are denoted by $\mathbb{C}_c$.*

2.4 Conversion between $\mathcal{L}(X_F)$, F, and $\mathcal{MF}(X_F)$

Since we will show that the topological entropy of subshifts with decidable language is uncomputable, we are under obligation to show that other standard characterizations of such languages do not yield computable entropy. We accomplish this by establishing effective conversions between decidable $\mathcal{L}(X)$ and $\mathcal{MF}(X)$:

Lemma 16 *There is a program which, given an oracle to $\mathcal{L}(X)$, outputs a decision procedure for $\mathcal{MF}(X)$. Conversely, there is a program which, given an oracle to a set of minimal words $\mathcal{MF}(X)$ for some subshift X outputs a decision procedure for $\mathcal{L}(X)$.*

Proof: It is straightforward to prove that $\mathcal{MF}(X) = (\Sigma \cdot \mathcal{L}(X)) \cap (\mathcal{L}(X) \cdot \Sigma) \cap (\Sigma^* \setminus \mathcal{L}(X))$, from which the result follows. □

Thus, we can effectively transform a decision procedure for $\mathcal{L}(X)$ into a decision procedure for $\mathcal{MF}(X)$ and vice versa.

Remark 17 *On the other hand, consider the class of languages F_i (where $i \in \mathbb{N}$) such that each F_i is inductively defined by: $0 \in F_i$ and for $k \in \mathbb{N}$, $1^k \in F_i$ iff $U(i)$ is in the halting state after k steps. Then, F_i is decidable for all $i \in \mathbb{N}$, but $1 \in \mathcal{L}(X_{F_i})$ iff U does not halt on i (and it is thus undecidable whether $1 \in \mathcal{L}(X_{F_i})$). An immediate corollary is that there is no recursive way of transforming a decision procedure for an arbitrary decidable set of forbidden words F to a decision procedure for $\mathcal{MF}(X_F)$, hence, in light of Lemma 16, neither to a decision procedure for $\mathcal{L}(X_F)$.*

3 A Simple Class of Sofic Shifts

We shall define a class of right-resolving graphs that represent irreducible sofic shifts with topological entropy $\log(2)/2$. When adding the full shift (with entropy $\log(2)$) to this class, we can exhibit the "discontinuous" behaviour of the topological entropy that we need to show it uncomputable.

Definition 18 *For each $k \in \mathbb{N}$ let T_k be the full directed binary tree of depth $k - 1$ such that the two outgoing edges from each non-leaf node are labelled with '0' and '1', respectively. Label each node with its address in the tree (i.e. the unique sequence of labels seen on edges in the unique path from the root to the node; the root is labelled λ). Let $S_0, \ldots, S_{k-1}$ be fresh nodes not in T_k and add edges with labels '0' thus: $S_0 \xrightarrow{0} S_1 \xrightarrow{0} \cdots \xrightarrow{0} S_{k-1}$. For each node N labelled with an element of $\mathbf{2}^{k-1}$, add two directed edges with labels '0' and '1' to S_0. Finally, add an edge with label '0' from S_{k-1} to the node labelled λ. Call the resulting graph A_k (see Figure 1).*

Clearly, A_k is right-resolving for all $k \in \mathbb{N}$, and since every node is reachable from every other node by a directed path (i.e. the graph is irreducible), the associated two-sided sofic shift is irreducible. Using irreducibility, it is straightforward to prove that the associated one-sided sofic shift is also shift-invariant.

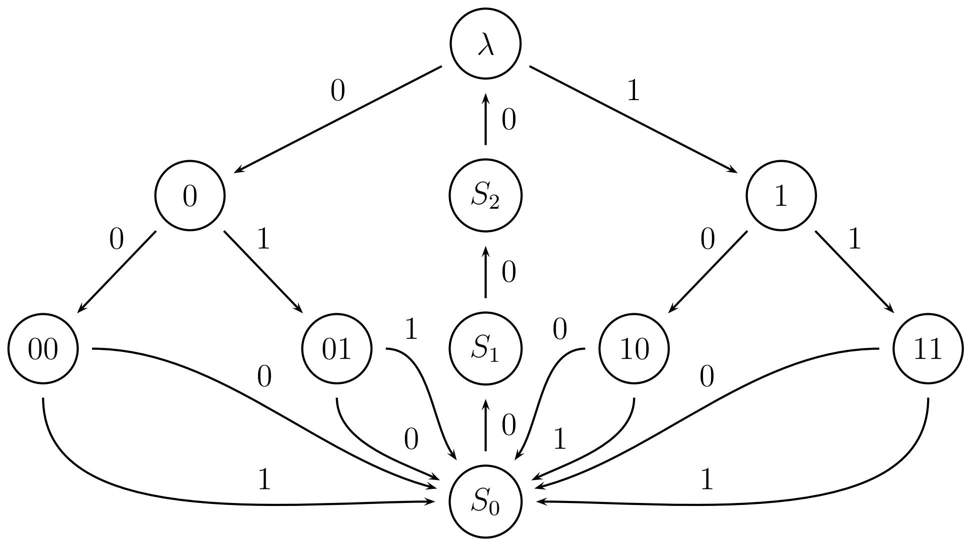

Fig. 1: The irreducible, right-resolving graph A_3

Definition 19 *Let $k \in \mathbb{N}$. The two-sided sofic shift consisting of the set of bi-infinite strings generated by bi-infinite paths in the A_k is called $X^{k,2}$. The corresponding one-sided sofic shift consisting of set of right-infinite strings generated by all infinite paths in A_k is called $X^{k,1}$.*

A simple auxiliary lemma is:

Lemma 20 *For any $n, k \in \mathbb{N}$, we have $2^{n \cdot k} \leq |\mathcal{L}(X^{k,1}) \cap \mathbf{2}^{n \cdot (2 \cdot k)}| = |\mathcal{L}(X^{k,2}) \cap \mathbf{2}^{n \cdot (2 \cdot k)}| \leq 2^{k+1} \cdot 2^{n \cdot k}$*

Proof: By construction, $|\mathcal{L}(X^{k,1}) \cap \mathbf{2}^{n \cdot (2 \cdot k)}| = |\mathcal{L}(X^{k,2}) \cap \mathbf{2}^{n \cdot (2 \cdot k)}|$. Also by construction, all strings of length $2 \cdot k$ generated by paths starting at the node labelled λ are distinct, and there are 2^k such strings. Note that each such path also ends at node λ. By extension, all strings of length $n \cdot (2 \cdot k)$ generated by paths starting and ending at node λ are distinct for any $n \in \mathbb{N}$, whence there are at least $(2^k)^n$ such strings, i.e. $2^{n \cdot k} \leq |\mathcal{L}(X^{k,1}) \cap \mathbf{2}^{n \cdot (2 \cdot k)}|$. Since any path of length $n \cdot (2 \cdot k)$ starting from *any* node N must pass through node λ after at most $2 \cdot k$ steps, we see that $(2^k)^n$ is an upper bound on the number of distinct strings of length $n \cdot (2 \cdot k)$ generated by such paths from node N. There are $2^k - 1 + k$ nodes, and thus: $|\mathcal{L}(X^{k,1}) \cap \mathbf{2}^{n \cdot (2 \cdot k)}| \leq (2^k - 1 + k) \cdot 2^{n \cdot k} \leq 2^{k+1} \cdot 2^{n \cdot k}$. □

Theorem 21 $h_{top}(X^{1,k}) = h_{top}(X^{2,k}) = \log(2)/2$

Proof: We have, using Lemma 20:

$$\begin{array}{rcccl} h_{\text{top}}(X^{1,k}) & = & \lim_{n\to\infty} \frac{\log(|\mathcal{L}(X^{1,k}) \cap \mathbf{2}^n|)}{n} & = & \lim_{n\to\infty} \frac{\log(|\mathcal{L}(X^{1,k}) \cap \mathbf{2}^{n \cdot (2 \cdot k)}|)}{n \cdot (2 \cdot k)} \\ & \leq & \lim_{n\to\infty} \frac{\log(2^{k+1} \cdot 2^{n \cdot k})}{n \cdot (2 \cdot k)} & = & \lim_{n\to\infty} \left(\frac{(k+1) \cdot \log(2)}{n \cdot (2 \cdot k)} + \frac{(n \cdot k) \cdot \log(2)}{n \cdot (2 \cdot k)} \right) \\ & & & = & \frac{\log(2)}{2} \end{array}$$

A completely analogous derivation using the other inequality of Lemma 20 shows that $(\log(2))/2 \leq h_{\text{top}}(X^{1,k})$, concluding the proof. □

4 The Topological Entropy of Subshifts with Decidable Languages is Computably Unapproximable

We now show our first main result.

Theorem 22 *The topological entropy of subshifts with decidable language is computably unapproximable, i.e. for no real number $\gamma \in (0, \log(2)/4)$ does there exist a program that, on input a decision procedure for the language of a subshift X will output a rational number p_o/q_o such that $|h_{top}(X) - p_o/q_o| < \gamma$. The result holds even when attention is restricted to the set of irreducible shifts, or to the set of irreducible, shift-invariant, one-sided shifts.*

Proof: For each $i \in \mathbb{N}$, construct a subset V_i of $\mathbf{2}^*$ monotonically as follows: Start with the empty set and initiate a run of $U(i)$. If $U(i)$ is not in the halting state after k steps, add all of the 2^k strings in $\mathbf{2}^k$ to V_i. If $U(i)$ *is* in the halting state after k steps, set $V_i = \mathcal{L}(X^{k,1})$ and stop the construction. Observe that if $U(i)$ does *not* halt, then $V_i = \bigcup_{k \in \mathbb{N}} \mathbf{2}^k = \mathbf{2}^*$, i.e. V_i is the language of the full shift $\mathbf{2}^{\mathbb{Z}}$ (or $\mathbf{2}^{\mathbb{N}}$) with topological entropy $\log(2)$. Thus, all V_i are the languages of (sofic) subshifts X_i. For each $i \in \mathbb{N}$, we have a naïve decision procedure for $\mathcal{L}(V_i)$: On input $a \in \mathbf{2}^*$ simply run $U(i)$ for $|a|$ steps. If $U(i)$ halts after $k < |a|$ steps, then $\mathcal{L}(V_i) = \mathcal{L}(X^{k,1})$, and it is clearly decidable, by tracing through all possible paths of length $|a|$ in the graph A_k, whether $a \in \mathcal{L}(X^{k,1}) \cap \mathbf{2}^{|a|}$. If $U(i)$ runs for $|a|$ or more steps, we know that $\mathbf{2}^{|a|} \subseteq V_i$, and thus $a \in V_i$.

Using Theorem 21, we see that $h_{\text{top}}(X_i) = \log(2)/2$ iff $U(i)$ halts, and $h_{\text{top}}(X_i) = \log(2)$ otherwise. If the entropy were approximable for some $\gamma \in (0, \log(2)/4)$, we could, for each $i \in \mathbb{N}$, effectively find $p_o/q_o \in \mathbb{Q}$ such that $|h_{\text{top}}(X_i) - p_o/q_o| < \gamma < \log(2)/4$. By construction of the V_i, for such a p_o/q_o we either have $p_o/q_o < 3/4 \cdot \log(2)/2$, or $p_o/q_o > 3/4 \cdot \log(2)/2$, and can effectively decide which one of these holds. But $\gamma < 3/4 \cdot \log(2)/2$ implies that $h_{\text{top}}(X_i) = \log(2)/2$ and $\gamma > 3/4 \cdot \log(2)/2$ implies that $h_{\text{top}}(X_i) = \log(2)$, i.e. we can decide whether $U(i)$ halts, an impossibility. □

The theorem can be extended to alphabets Σ arbitrarily larger than $\mathbf{2}$ in a straightforward way by letting the fan-out of the graphs A_k be equal to $|\Sigma|$, and can thus, if the base of the logarithm in the definition of $h_{\text{top}}(X)$ is fixed, be made to hold for arbitrary $\gamma \in \mathbb{R}^{>0}$.

Note that the real number γ of the theorem may be specified however we like; for instance, if γ is rational, we may choose the canonical representation of γ as an irreducible fraction p/q. Thus, the approximability result above is not contingent on any computational artifacts due to representations of real numbers. The same observation holds for the following corollary where $\gamma \in (0, \log(2)/4)$ may be replaced by $p/q \in \mathbb{Q} \cap (0, \log(2)/4)$ at the reader's leisure.

Corollary 23 *For any real number $\gamma \in (\log(2)/2, \log(2))$, the following questions are all undecidable.* Given*: A decision procedure for $\mathcal{L}(X)$ where X is a (one- or two-sided) subshift.* To decide*:*

1. *Is $h_{top}(X) = \log(2)/2$? (Alternatively, is $h_{top}(X) = \log(2)$?)*

2. *Is $h_{top}(X) \leq \gamma$? ($\geq \gamma$?)*

3. *Is $h_{top}(X) < \gamma$? ($> \gamma$?)*

Proof: Consider the languages V_i in the proof of Theorem 22. Each of these is the language of a subshift X_i. If *any* of the three questions were decidable, we could decide both whether $h_{\text{top}}(X_i) = \log(2)/2$ and whether $h_{\text{top}}(X_i) = \log(2)$ and thus whether $U(i)$ halts. □

Corollary 24 *The topological entropy of subshifts with decidable set F of forbidden words is uncomputable (in the sense of Theorem 22). The result holds even when restricted to sets of minimal forbidden words.*

Proof: Otherwise, we could, for each subshift X with decidable language, first use the effective procedure of Lemma 16 to obtain a decision procedure for $\mathcal{MF}(X)$ and subsequently compute $h_{\text{top}}(X) = h_{\text{top}}(X_{\mathcal{MF}(X)})$, contradicting Theorem 22. □

Though we have used sofic shifts to prove the topological entropy computably unapproximable, this says nothing about the "right" way to compute the entropy for sofic shifts. The result of this section essentially showed that computing the entropy given decision procedures P as input is impossible, even when P is accompanied by a (non-constructive) proof that it decides the language of a sofic shift. This is a completely valid result, since the "natural" input to a program in that case is a decision procedure for the language, however it is almost certainly the wrong way to reason about the class of sofic shifts, since their "natural" representation is right-resolving graphs. We shall see in the next section that this fact makes a world of difference.

5 The Topological Entropy of Sofic Shifts is Computable *and* has Decidable Comparisons

We now restrict our attention to the computation of the topological entropy of sofic shifts *when these are represented by right-resolving graphs*. There are algorithms for finding (computable names of) the topological entropy of sofic shifts, given a finite graph generating their language. However, the question of *decidability* remains. Since it is not in general decidable for a $p/q \in \mathbb{Q}$ and a (computable name of a) computable real x whether $p/q < x$, computability of $h_{\text{top}}(S)$ for a sofic shift S does not immediately entail decidability of the questions of Corollary 23. However, it turns out that they *are* decidable if the logarithm in the definition of topological entropy is taken to integer base. In the remainder of the section we assume the logarithm to be taken to base 2; the reader may substitute any other integer > 2 at her leisure.

We begin with a staple result from computable analysis:

Theorem 25 (Computable Fundamental Theorem of Algebra (Lite)) *There is a program P taking as input $z_1, \ldots, z_n \in \mathbb{Z}$ (where $n \in \mathbb{N}$ may vary) and outputting a list of n computable names of elements $c_1, \ldots, c_n \in \mathbb{C}_c$ such that $c_1, \ldots, c_n$ are the n roots (counted with multiplicities) of the polynomial $x^n + z_{n-1}x^{n-1} \cdots + z_1 x + z_0$.*

Proof: See [20]. The "lite" tag refers to the fact that $z_0, \ldots, z_{n-1} \in \mathbb{Z}$, and not $z_0, \ldots, z_{n-1} \in \mathbb{C}_c$. $\square$

Note that the theorem does not tell us how many distinct roots there are. For a given computable complex number, its modulus is computable, i.e. there is a program that given a computable name of $c \in \mathbb{C}_c$ outputs a computable name of $|c|$ [20]. Thus, we can effectively list the n moduli of the roots of an n-degree polynomial over $\mathbb{Z}$. But we cannot, in general, decide, for a given $1 \leq i \leq n$, whether c_i is an eigenvalue of maximal modulus (since, in finite time, the names only yield *approximations* to the roots). However, we *can* compute the spectral radius, as shown in the following folklore lemma (see also [21]):

Lemma 26 *The spectral radius of a matrix with non-negative integer coefficients is a computable real. The proof can be effectivized, i.e. there is a program that, given a suitable representation of any such matrix as input, outputs a computable name of the spectral radius.*

Proof: We use the following effective procedure: Given a representation of a matrix of dimension $n \times n$ with integer coefficients, we can effectively find a representation for $det(xI - A)$. By Theorem 25 and the comments thereafter, we effectively obtain a list of computable names of the n moduli $y_1, \ldots, y_n$ of the n roots of $det(xI - A)$. Let $\psi : \mathbb{N} \longrightarrow \mathbb{Q}$ be the total recursive function that does the following: On input $m \in \mathbb{N}$, use the list $y_1, \ldots, y_n$ to find $p_1/q_1, \ldots, p_n/q_n \in \mathbb{Q}$ satisfying $|y_i - p_i/q_i| < 2^{-m}$ for all $1 \leq i \leq n$ (possible, since the y_i are computable names). Output a maximal p_i/q_i.

We claim that ψ is a computable name of the spectral radius λ_{max}. To see this, observe that, for each $m \in \mathbb{N}$, one of the rationals $p_1/q_1, \ldots, p_n/q_n$ is at most 2^{-m} from the Perron eigenvalue (that equals the spectral radius, in this case). If p_i/q_i is maximal among $p_1/q_1, \ldots, p_n/q_n$, it is at most 2^{-m} from an eigenvalue that is $\leq \lambda_{\text{max}}$. We must thus have $|\lambda_{\text{max}} - p_i/q_i| < 2^{-m}$. Hence, ψ is a computable name of λ_{max}. $\square$

Corollary 27 *There is a program that, on input a right-resolving graph G outputs a computable name of* $h_{top}(X_G)$.

Proof: By the lemma and the fact that there is a program that as input takes a computable name of $x \in \mathbb{R}_c^{\geq 0}$ as input and outputs a computable name of $\log(x)$ [20]. $\square$

We reiterate that Lemma 26 in itself is not sufficient for decidability due to the inherent trouble with effectively comparing (even computable) reals to rational numbers.

We now prove our second main result:

Theorem 28 *The following problems are all decidable in polynomial time in* $|G|$. Given: *(1)* $p/q \in \mathbb{Q}^{\geq 0}$ *and (2) a right-resolving finite graph G over some alphabet* Σ. To decide:

1. *Is* $h_{top}(X_G) = p/q$*?*
2. *Is* $h_{top}(X_G) \geq p/q$ *(*$\leq p/q$*)?*
3. *Is* $h_{top}(X_G) > p/q$ *(*$< p/q$*)?*

Proof: To decide whether $h_{\text{top}}(X_G) = p/q$, we use exactly the same method as the polynomial-time algorithm for deciding whether two sofic shifts have identical topological entropies in [3]. The only difference is that instead of comparing two sofic shifts (via their associated polynomials), we have only one sofic shift and simply replace the polynomial of the other shift by the characteristic polynomial, $X^q - 2^p$, of the algebraic number $2^{p/q}$.

For completeness, we sketch the method: We can effectively construct (a representation of) the adjacency matrix A_G in polynomial time in $|G|$. An algorithm due to Davenport [6] provides, for each polynomial Q in $\mathbb{Z}[X]$, a finite sequence of pairwise disjoint intervals with endpoints in $\mathbb{Q}$ such that each interval contains exactly one real root of Q. Apply this algorithm to $det(xI - A_G)$ and to $X^q - 2^p$ and observe that the spectral radius of A_G is the largest real root of $det(xI - A_G)$, and that $2^{p/q}$ is the largest real root of $X^q - 2^p$. Use standard polynomial division to effectively obtain the polynomial Q in $\mathbb{Z}[X]$ that is the gcd of $det(x_I - A_G)$ and $X^q - 2^p$. Using Sturm's Theorem, we can in polynomial time in the size of Q decide whether Q has a root in the rightmost interval of each of the two finite sequences (see e.g. [6, 14]). The largest real roots of $det(xI - A_G)$ and $X^q - 2^p$ are identical iff Q has a root in each of the two rightmost intervals, i.e. iff $\lambda_{\text{max}} = 2^{p/q}$, showing that $h_{\text{top}}(X) = p/q$ is decidable in polynomial time in $|G|$.

To decide whether $h_{\text{top}}(X_G) > p/q$, consider the rightmost interval $[p_1/q_1, p_2/q_2]$ output by Davenport's algorithm on $det(xI - A_G)$ (which we can obtain in polynomial time). If $p/q < p_1/q_1$ or $p/q > p_2/q_2$, we get the answer immediately.

If $p/q = p_1/q_1$ or $p/q = p_2/q_2$, we may, by the first part of the proof, check whether $h_{\text{top}}(X_G) = p/q$ in polynomial time. If $p/q = p_1/q_1$, we have $h_{\text{top}}(X_G) > p/q$, and we may halt with 'yes'. If $p/q = p_2/q_2$, we have $h_{\text{top}}(X_G) < p/q$ and we may halt with 'no'.

Otherwise, $p_1/q_1 < p/q < p_2/q_2$, and we may consider the two non-degenerate intervals $[p_1/q_1, p/q]$ and $[p/q, p_2/q_2]$. Since $h_{\text{top}}(X_G) \neq p/q$, exactly one of these two intervals contains $h_{\text{top}}(X_G)$. Also, $h_{\text{top}}(X_G)$ is the only real zero of $det(xI - A_G)$ in $[p_1/q_1, p_2/q_2]$. We may then use Sturm's Theorem applied to both intervals to ascertain in polynomial time the number of zeroes (which is either 0 or 1) in each. If the zero is in $[p_1/q_1, p/q]$, we may halt with output 'no'. Otherwise, the zero is in $[p/q, p_2/q_2]$, and we may halt with output 'yes'. Thus, $h_{\text{top}}(X_G) > p/q$ is decidable in polynomial time.

Polynomial-time decidability of the remaining questions follow from polynomial-time decidability of $h_{\text{top}}(X_G) = p/q$ and $h_{\text{top}}(X_G) > p/q$. $\square$

The polynomial time bound of Davenport's algorithm used in the proof above assumes that comparison of two arbitrary rational numbers can be done in unit time, as is usually assumed in computational algebra [14]. However, Davenport's algorithm only increases the number of bits in any two rational numbers by a polynomial factor. Thus, even if comparison between rational numbers takes linear time in the number of bits used to represent the numbers, the above algorithm still runs in polynomial time in the size of G and p/q.

6 Related and Future Work

This paper adds a new class of dynamical system to the slowly growing list of classes for which the topological entropy is computably unapproximable [9, 12]; as with the other classes, we essentially use a discontinuity property of the entropy, but our concrete construction has a flavour quite distinct from these, and from the results in [7] where the apparently weaker result is shown that, for some classes of systems, the question $h_{\text{top}}(X) = \alpha$ (where α is any real number) is undecidable.

Considerable effort has been devoted to construct subsets of $\mathbf{2}^*$ having entropy rates equalling computable reals, right- but not left-computable reals, and other supersets of the computable reals. These methods are almost invariably based on sets with (Hausdorff) dimension equalling the entropy rate of the language [16, 17, 1] (also [11] for a fundamental construction that can be translated to entropy rates). The entropy rate of a subset of $\mathbf{2}^*$ is defined analogously to the topological entropy of a subshift, but even though the languages constructed in the above papers are often decidable, they don't appear to be *factorial* in general, hence cannot be the languages of subshifts.

We note that if there exists a subshift X such that $\mathcal{L}(X)$ is decidable and $h_{\text{top}}(X)$ is not a computable real, then the results in this paper related to uncomputability would follow immediately. In the lingo of language theory, this becomes the interesting question:

Does there exist a decidable, extensible, factorial language the entropy rate of which is not a computable real?

Finally, there is a remarkable gap from Theorem 28 to Theorem 22. This is perhaps not surprising, since it essentially is the gap from the class of (factorial) regular languages to the full class of (factorial) decidable ones. There has been little work in finding subrecursive classes of *factorial* languages larger than the regular ones. Characterization of such classes could be a first step in pinning down the computational properties of the topological entropy. One possible class (in fact, the largest "natural" class) to consider is the set of *primitive recursive* factorial languages [15]; we urge the reader to pursue this possibility.

References

[1] K.B. Athreya, J.M. Hitchcock, J.H. Lutz, and E. Mayordomo. Effective strong dimension in algorithmic information and computational complexity. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS '04)*, volume 2996 of *Lecture Notes in Computer Science*, pages 632–643. Springer-Verlag, 2004.

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

[3] M.-P. Béal, M. Crochemore, F. Mignosi, A. Restivo, and M. Sciortino. Computing forbidden words of regular languages. *Fundamenta Informaticae*, 56(1–2):121–135, 2003.

[4] M.-P. Béal, F. Mignosi, A. Restivo, and M. Sciortino. Forbidden words in symbolic dynamics. *Advances in Applied Mathematics*, 25:163–193, 2000.

[5] M.-P. Béal and D. Perrin. Symbolic dynamics and finite automata. In G. Rosenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 2, chapter 10. Springer-Verlag, 1997.

[6] J.H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra*. Academic Press, Ltd., 2nd edition, 1993.

[7] J.-C. Delvenne and V.D. Blondel. Quasi-periodic configurations and undecidable dynamics for tilings, infinite words and turing machines. *Theoretical Computer Science*, 319:127–143, 2004.

[8] 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.

[9] L. Hurd, J. Kari, and K. Culik. The topological entropy of cellular automata is uncomputable. *Ergodic Theory and Dynamical Systems*, 12:255–265, 1992.

[10] K.-I. Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.

[11] K.-I. Ko. On the computability of fractal dimensions and hausdorff measure. *Annals of Pure and Applied Logic*, 91(1–3):195–216, 1998.

[12] P. Koiran. The topological entropy of iterated piecewise affine maps is uncomputable. *Discrete Mathematics and Theoretical Computer Science*, 4(2):351–356, 2001.

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

[14] B. Mishra. *Algorithmic Algebra*. Texts and Monographs in Computer Science. Springer-Verlag, 1993.

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

[16] L. Staiger. Kolmogorov complexity and hausdorff dimension. *Information and Computation*, 103:159–194, 1993.

[17] L. Staiger. A tight upper bound on kolmogorov complexity and uniformly optimal prediction. *Theory of Computing Systems*, 31:215–229, 1998.

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

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

[20] K. Weihrauch. *Computable Analysis: An Introduction*. Springer, 1998.

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