

On the Computational Complexity of the Languages of General Symbolic Dynamical Systems and Beta-Shifts

Jakob Grue Simonsen

Department of Computer Science, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark
simonsen@diku.dk

May 11, 2009

Abstract

We consider the computational complexity of languages of symbolic dynamical systems. In particular, we study complexity hierarchies and membership of the non-uniform class P/poly. We prove:

1. For every time-constructible, non-decreasing function $t(n) = \omega(n)$, there is a symbolic dynamical system with language decidable in deterministic time $\mathcal{O}(n^2t(n))$, but not in deterministic time $o(t(n))$.
2. For every space-constructible, non-decreasing function $s(n) = \omega(n)$, there is a symbolic dynamical system with language decidable in deterministic space $\mathcal{O}(s(n))$, but not in deterministic space $o(s(n))$.
3. There are symbolic dynamical systems having hard and complete languages under \leq_m^{logs} - and \leq_m^p -reduction for every complexity class above LOGSPACE in the backbone hierarchy (hence, P-complete, NP-complete, coNP-complete, PSPACE-complete, and EXPTIME-complete sets).
4. There are decidable languages of symbolic dynamical systems in P/poly for every alphabet of size $|\Sigma| \geq 1$.
5. There are decidable languages of symbolic dynamical systems not in P/poly iff the alphabet size is > 1 .

For the particular class of symbolic dynamical systems known as β -shifts, we prove that:

1. For all real numbers $\beta > 1$, the language of the β -shift is in P/poly.
2. If there exists a real number $\beta > 1$ such that the language of the β -shift is NP-hard under \leq_T^p -reduction, then the polynomial hierarchy collapses to the second level. As NP-hardness under \leq_m^p -reduction implies hardness under \leq_T^p -reduction, this result implies that it is unlikely that a proof of existence of an NP-hard language of a β -shift will be forthcoming.
3. For every time-constructible, non-decreasing function $t(n) \geq n$, there is a real number $1 < \beta < 2$ such that the language of the β -shift is decidable in time $\mathcal{O}(n^2t(\log n + 1))$, but not in any proper time bound $g(n)$ satisfying $g(4^n) = o(t(n)/16^n)$.
4. For every space-constructible, non-decreasing function $s(n) = \omega(n^2)$, there is a real number $1 < \beta < 2$ such that the language of the β -shift is decidable in space $\mathcal{O}(s(n))$, but not in space $g(n)$ where g is any function satisfying $g(n^2) = o(s(n))$.
5. There exists a real number $1 < \beta < 2$ such that the language of the β -shift is recursive, but not context-sensitive.

1 Symbolic dynamical systems and (the complexity of) their languages

Symbolic dynamics [LM95, BP97, BMRS00] is the discipline of studying spaces of infinite sequences over some alphabet and an associated shift operator left-shifting the infinite words.

To each symbolic dynamical system is associated a *language* consisting of finite sequences; this paper is concerned with finding dynamical systems that have languages *hard* and *complete* for well-known complexity classes such as P, NP, EXPTIME [Jon97, KD00, Sip06], and to establish hierarchies of dynamical systems ranked by the hardness of their languages.

To this end, we use the well-known characterization of languages of dynamical systems as being exactly the so-called *factorial* and *extensible* languages; as an important stepping stone, we also consider the class of *anti-factorial* languages consisting of languages whose elements do not contain certain “forbidden” words.

While the dynamical systems considered for the above results are tailor-made for this paper, we also consider the concrete case of the class of β -shifts [Par60, Sch97, DK02, FS92, DdV05], one of the most well-studied classes of symbolic dynamical systems. As it turns out, β -shifts are much less likely to exhibit languages that are hard for certain complexity classes, indeed due to the languages of β -shifts being contained in the complexity class P/poly, existence of, β -shifts with languages hard for, say, P or NP only holds if the dubious results $P = NP$, respectively $P = LOGSPACE$, hold.

While we briefly consider the possibility of a good complexity hierarchy result for β -shifts, we are unable to establish existence of a hierarchy sufficiently “tight” to be truly interesting in itself. However, one consequence of the hierarchy result is that we can settle the open question of the existence of a β -shift with recursive, but not context-sensitive language.

Previous work on the computational properties of languages of dynamical systems has focused on the decidability of the languages [Sim05, Sim06, HS08b, HS08a]. We hope that the present paper may aid in establishing a more fine-grained analysis.

The paper is organized as follows:

- Sections 2 and 3 review basic facts about formal languages, symbolic dynamics and complexity theory, respectively. Readers with background in one or more of these may skip the relevant sections at their leisure.
- Section 4 establishes time and space overhead for converting between factorial and anti-factorial languages and various languages associated with β -shifts.
- Section 5 establishes hardness- and completeness results for general symbolic dynamical systems and β -shifts.
- Section 6 sets up complexity hierarchies for general symbolic dynamical systems and β -shifts.
- Section 7 concerns construction of symbolic dynamical systems with languages not in the non-uniform complexity class P/poly.
- Section 8 Gives a list of open problems suitable for future research.

2 Languages of Symbolic Dynamical Systems: Preliminaries

In this and the following section, we give only the briefest of introductions and mainly list the relevant definitions without further comment.

A *language* over a non-empty alphabet Σ is a subset of Σ^* , the set of all finite words¹ over Σ . The empty word over Σ is denoted by λ .

Definition 1: Given a language $L \subseteq \Sigma^*$, the census function $c_L : \mathbb{N} \rightarrow \mathbb{N}_0$ for L is defined by $c_L(n) = |L \cap \Sigma^n|$. The language L is said to be *sparse* if there exists a polynomial p with integer coefficients s.t. c_L is bounded above by p , i.e. for all $n \in \mathbb{N}$, we have $c_L(n) \leq p(n)$. \square

Definition 2: The language L is said to be *factorial* if $x \cdot y \in L$ implies $x \in L$ and $y \in L$. L is said to be *extensible* if $x \in L$ implies existence of $a, b \in \Sigma$ such that $axb \in L$. The language L is said to be *anti-factorial* if, for all $x, y, z \in \Sigma^*$, we have: $x \cdot z \cdot y \in L$ and either $x \neq \lambda$ or $y \neq \lambda$, implies $z \notin L$; that is, no proper subword of $x \cdot z \cdot y$ is an element of L . \square

Definition 3: Let Σ be a non-empty, finite alphabet. Σ^* , $\Sigma^{\mathbb{N}}$ and $\Sigma^{\mathbb{Z}}$ denote the sets of finite words, right-infinite words and bi-infinite words over Σ , respectively. A subword of a finite or infinite word x is a finite, contiguous set of elements of Σ occurring in x .

The shift operation on $\Sigma^{\mathbb{Z}}$ is the map $\sigma : \Sigma^{\mathbb{Z}} \rightarrow \Sigma^{\mathbb{Z}}$ such that if $x \in \Sigma^{\mathbb{Z}}$ with $x = (x_i)_{i \in \mathbb{Z}}$ (where x_i is the i th “coordinate” of x), then $\sigma(x)$ is the element of $\Sigma^{\mathbb{Z}}$ whose i th coordinate is the $i + 1$ th coordinate of x . The shift map on $\Sigma^{\mathbb{N}}$ is defined analogously.

A symbolic dynamical system (aka. shift space, subshift, or just shift), abbreviated symbolic dynamical system, is a subset, X_F of $\Sigma^{\mathbb{Z}}$ (two-sided shift) or $\Sigma^{\mathbb{N}}$ (one-sided shift) such that there is a set $F \subseteq \Sigma^*$ where the elements of X_F are exactly those $x \in \Sigma^{\mathbb{Z}}$ ($\Sigma^{\mathbb{N}}$ in the one-sided case) that contain no element of F as a subword. \square

It is easy to see that that any symbolic dynamical system X is invariant under the shift operation, i.e. $\sigma(X) = X$.

Definition 4: The language of a symbolic dynamical system X over alphabet Σ , denoted $\mathcal{L}(X)$, is the subset of Σ^* consisting of those $x \in \Sigma^*$ occurring as subwords of elements of X . \square

The following is straightforward to prove:

Proposition 1: Let $L \subseteq \Sigma^*$ be a language. Then there is a symbolic dynamical system X such that $L = \mathcal{L}(X)$ iff L is factorial and extensible. \square

In addition, the following holds:

Proposition 2: [[BMRS00, BCM⁺03]] Let F be anti-factorial. We have:

- $\mathcal{L}(X_F) = \Sigma^* \setminus (\Sigma^* F \Sigma^*)$
- $F = (\Sigma \mathcal{L}(X_F)) \cap (\mathcal{L}(X_F) \Sigma) \cap (\Sigma^* \setminus \mathcal{L}(X_F))$, i.e. $y = y_1 \cdots y_n \in F$ iff $y \notin \mathcal{L}(X_F)$, and $y_1 \cdots y_{n-1} \in \mathcal{L}(X_F)$ and $y_2 \cdots y_n \in \mathcal{L}(X_F)$

\square

We refer the reader to [LM95] for ample background material concerning general symbolic dynamical systems.

¹In keeping with the nomenclature of dynamical systems, we shall consistently use the term “word” instead of the term “string” more common in other parts of computer science.

2.1 β -Shifts

One of the most studied classes of symbolic dynamical systems is the class of β -shifts [Wal81, DK02, Bla89, Sch97], introduced below.

Definition 5: Let \leq_{lex} be the usual lexicographic order on finite and infinite words over an ordered alphabet Σ . If words a, b are of different length, we assume the shortest of them to be padded with the least element of Σ before comparison with \leq_{lex} is performed.

Let furthermore $\beta > 1$ be a non-integral real number. The β -compactum, X_β , is the set of infinite sequences in $\{0, \dots, \lfloor \beta \rfloor\}^{\mathbb{N}}$ that are expansions of real numbers in $[0, 1)$ to base $\lfloor \beta \rfloor$. Equivalently, define the operation T_β on reals x by $T_\beta : x \mapsto \beta x \pmod{1}$ and define for $i \geq 1$: $(d_\beta(x))_i \triangleq \lfloor \beta T_\beta^{i-1}(x) \rfloor$ (where we put $T^0(x) = 1$). We denote the sequence $(d_\beta(1))_{i \in \mathbb{N}}$ by $d_\beta(1)$ for short.

If $d_\beta(1) = u_1 \cdots u_m \cdot 0^\omega$ where $u_m \neq 0$, define $d_\beta^*(1)$ to be the sequence $(u_1 u_2 \cdots (u_m - 1))^\omega$. Otherwise define $d_\beta^*(1)$ to be $(u_i)_{i \in \mathbb{N}}$. Then, X_β consists of the subset of those $x \in \{0, \dots, \lfloor \beta \rfloor\}^\omega$ such that, for all $i \in \mathbb{N}_0$: $\sigma^i(x) \leq_{\text{lex}} d_\beta^*(1)$.

The (one-sided) β -shift is the set X_β . The two-sided β -shift is the subset \hat{X}_β of $\Sigma^{\mathbb{Z}}$ such that $(x_z)_{z \in \mathbb{Z}} \in \hat{X}_\beta$ iff for all $z \in \mathbb{Z}$, the right-infinite sequence $x_z x_{z+1} x_{z+2} \cdots$ is an element of X_β . \square

It is straightforward to see that, for any non-integral real number $\beta > 1$, both the one- and the two-sided β -shifts are indeed symbolic dynamical systems.

3 Computational Complexity: Preliminaries

The reader is assumed to be familiar with the usual notion of Turing machines and the backbone hierarchy of complexity classes: P, NP, PSPACE, EXPTIME and so on; ample introductions can be found in [Jon97, KD00, Sip06]. All algorithms in this paper are supposed to be implemented on Turing machines with one input tape and at least one auxiliary tape.

We remind the reader that an oracle machine is a Turing machine with access to a set A , membership of which can be queried for in constant time. For any set A , we let NP^A denote the class of sets accepted by polynomial-time non-deterministic oracle Turing machines with an oracle to A and if \mathcal{C} is some class of sets, we let $\text{NP}^{\mathcal{C}}$ be $\bigcup_{A \in \mathcal{C}} \text{NP}^A$. The definition extends naturally to P^A , $P^{\mathcal{C}}$ and coNP^A and $\text{coNP}^{\mathcal{C}}$.

Definition 6: Set $A \subseteq \Sigma^*$ many-one reduces to set $B \subseteq \Sigma^*$, written $A \leq_m B$ if there is a Turing machine M that takes any $x \in \Sigma^*$ to element $M(x) \in \Sigma^*$ and $x \in A$ iff $M(x) \in B$. If M can be taken to run in LOGSPACE, we write $A \leq_m^{\text{logs}} B$, and if M can be taken to run in polynomial time, we write $A \leq_m^p B$.

Set $A \subseteq \Sigma^*$ Turing-reduces to set $B \subseteq \Sigma^*$, written $A \leq_T B$ if there is a Turing-machine M with an oracle to B that decides A . If M can be taken to run in polynomial time—assuming that queries to the oracle take unit time—we write $A \leq_T^p B$. \square

It is straightforward to show that $A \leq_m^p B$ implies that $A \leq_T^p B$ (see also [KD00, Prop. 2.23]).

A straightforward connection with factorial and anti-factorial languages is the following.

Proposition 3: Let F be anti-factorial. We have:

- $\mathcal{L}(X_F) \leq_T^p F$, in particular, if F is decidable, so is $\mathcal{L}(X_F)$.

- $F \leq_T^p \mathcal{L}(X_F)$, in particular, if $\mathcal{L}(X_F)$ is decidable, then so is F .

□

Proof: By Proposition 2, given $x \in \Sigma^*$ to decide whether $x \in \mathcal{L}(X_F)$, we need only enumerate that $\mathcal{O}(n^2)$ possible subwords y of x and query whether $y \in F$ for each subword. We have $x \in \mathcal{L}(X_F)$ iff no such subword is in F . At worst, we only use a polynomial number of queries, whence $\mathcal{L}(X_F) \leq_T^p F$. Conversely, By Proposition 2, given $x \in \Sigma^*$ to decide whether $x = x_1 \cdots x_n \in F$, we need perform only 3 queries to $\mathcal{L}(X_F)$, namely $x \in \mathcal{L}(X_F)$, $x_1 \cdots x_{n-1} \in \mathcal{L}(X_F)$ and $x_2 \cdots x_n \in \mathcal{L}(X_F)$. □

The polynomial hierarchy is a complexity-theoretic analogue of the so-called arithmetical hierarchy from logic; formally:

Definition 7: Define $\Delta_0^p \triangleq \Sigma_0^p \triangleq \Pi_0^p \triangleq P$.

Now, for $i \geq 0$, define:

- $\Delta_{i+1}^p \triangleq P^{\Sigma_i^p}$.
- $\Sigma_{i+1}^p \triangleq NP^{\Sigma_i^p}$.
- $\Pi_{i+1}^p \triangleq coNP^{\Sigma_i^p}$.

And, finally, $PH \triangleq \bigcup_{i \in \mathbb{N}_0} \Delta_i^p$.

□

Straightforward consequences of the definition are: $NP = \Sigma_1^p$, $coNP = \Pi_1^p$ and $PH \subseteq PSPACE$. It is unknown whether the polynomial hierarchy collapses, that is, whether there is an $n \in \mathbb{N}$ such that $\Delta_m^p = \Delta_n^p$ for all $m \geq n$.

Definition 8: The class *P/poly* comprises the sets $A \subseteq \Sigma^*$ such that there exists a sparse set S with $A \leq_T^p S$. □

The above definition hides several interesting facts about P/poly: it is in addition the class of sets decidable by boolean circuits of polynomial size in the input length, and also the class of sets decidable by Turing machines in polynomial time when the machine has access to an *advice* word s for each input length n such that the length of s is bounded above by a polynomial in n [KD00].

Lemma 1: [From [KD00, Lem. 6.5]] If $A \leq_T^p B$ and $B \in P/poly$, then $A \in P/poly$. □

3.1 Hierarchy Theorems

Intuitively, giving the class of Turing machines access to (asymptotically) greater resources should enable them to solve a properly larger class of problems. Formal results of this type are known as *hierarchy theorems* and have been particularly studied for the *time* and *space* measures of Turing machines which we assume known to the reader.

Definition 9: Let $f : \mathbb{N} \rightarrow \mathbb{N}$. Then f is said to be time-constructible if there is a Turing machine M which, given a word 1^n of n ones, stops after exactly $f(n)$ steps. The function f is said to be space-constructible if there is a Turing machine, on input 1^n halts after using exactly $f(n)$ cells of storage. □

The currently strongest known time hierarchy theorem for multi-tape Turing machines is:

Theorem 1: [Fürer [Für82]] For every alphabet Σ with $|\Sigma| \geq 2$ and every time-constructible $t(n) > n$, there is a language over Σ that is decidable in time $t(n)$, but not decidable in time $o(t(n))$. \square

The currently strongest known hierarchy theorem for space classes is:

Theorem 2: [Geffert [Gef03]] For every non-empty alphabet Σ , every space-constructible $s(n) \geq \log(n)$, every function $f(n) \in \mathcal{O}(s(n))$ and computable function $g(n) \in o(s(n))$, there is a language over Σ decidable in space $f(n)$, but not decidable in space $g(n)$. \square

4 Conversion results for symbolic dynamical systems

Our goal is to establish complexity results for symbolic dynamical systems whose languages are all factorial and extensible. However, languages constructed explicitly for general results in complexity theory are not in general factorial or extensible, whence we shall need to convert back-and-forth between arbitrary languages and factorial and extensible ones. This section establishes such conversions and the associated time and space overheads incurred by such conversions.

4.1 Anti-Factorial Languages

To obtain an anti-factorial language from any language L over alphabet Σ , it suffices to add an extra symbol $\# \notin \Sigma$ and use it as a marker at the beginning and end of each element of L .

Definition 10: Given $L \subseteq \Sigma^*$ and $\# \notin \Sigma$, define $L^\# \triangleq \{\#x\# : x \in L\}$ \square

Proposition 4: For any $L \subseteq \Sigma^*$, $L^\#$ is anti-factorial. \square

Proof: If $a, b \in L^\#$ and a occurs as a subword of b , then by construction of $L^\#$ and the fact that $\# \notin \Sigma$, we must have $a = b$. \square

Proposition 5: If L is decidable in deterministic time $t(n)$ and deterministic space $s(n)$, then $L^\#$ is decidable in deterministic time $\mathcal{O}(t(n) + n)$ and deterministic space $\mathcal{O}(s(n) + n)$. \square

Proof: On input $y \in (\Sigma \cup \{\#\})^*$, we may check using time $\mathcal{O}(n)$ and constant space whether $y = \#x\#$ where $x \in \Sigma^*$. Subsequently, we may simply query a decision procedure for L asking whether $x \in L$. This incurs a total resource use of $\mathcal{O}(t(n) + n)$ time and $\mathcal{O}(s(n) + n)$ space. \square

The obvious converse result holds.

Proposition 6: If $L^\#$ is decidable in deterministic time $T(n)$ (deterministic space $S(n)$), then L is decidable in deterministic time $\mathcal{O}(T(n) + n)$ (deterministic space $\mathcal{O}(S(n) + n)$). \square

Proof: On input $x \in \Sigma^*$, we may construct the word $\#x\#$ using time $\mathcal{O}(n)$ and space $\mathcal{O}(n)$, and subsequently query a decision procedure for $L^\#$, asking whether $\#x\# \in L^\#$, for a total time usage of $\mathcal{O}(T(n) + n)$ and space usage of $\mathcal{O}(S(n) + n)$. \square

The proofs of the two propositions above can obviously be made to work for non-deterministic computation as well, but we shall not need those results in the present paper.

4.2 Factorial, Extensible Languages

We know that if F is anti-factorial, then $\mathcal{L}(X_F)$ is factorial and extensible. We now establish suitable conversion results for converting between (anti-factorial) languages F and $\mathcal{L}(X_F)$.

Proposition 7: *Let $t(n)$ and $s(n)$ be non-decreasing functions such that $t(n) \geq n$ and $s(n) \geq n$. If F is decidable in deterministic time $t(n)$ and deterministic space $s(n)$, then $\mathcal{L}(X_F)$ is decidable in time $\mathcal{O}(n^2t(n))$ and space $\mathcal{O}(s(n))$.* \square

Proof: By Proposition 2, on input $y \in \Sigma^*$, it suffices to check whether, for each subword z of y , $z \notin F$. By examining y , each of the $\mathcal{O}(n^2)$ possible subwords of y can be constructed in time $\mathcal{O}(n)$ and space $\mathcal{O}(n)$. By monotonicity of $t(n)$ and $s(n)$, checking all of the $\mathcal{O}(n^2)$ subwords of z takes time bounded above by $\mathcal{O}(n^2t(n))$ and space bounded above by $\mathcal{O}(s(n))$ (as the space used to process each subword can be recovered). \square

For later use, we shall need a variant of the previous proposition for non-deterministic time and space.

Proposition 8: *Let $t(n)$ and $s(n)$ be non-decreasing functions such that $t(n) \geq n$ and $s(n) \geq n$. If $\Sigma^* \setminus F$ is decidable in non-deterministic time $t(n)$ and non-deterministic space $s(n)$, then $\mathcal{L}(X_F)$ is decidable in non-deterministic time $\mathcal{O}(n^2t(n))$ and non-deterministic space $\mathcal{O}(s(n))$.* \square

Proof: By Proposition 2, on input $y \in \Sigma^*$, it suffices to check, for each subword z of y , whether z is *not* in F . Thus, it suffices to check whether all $\mathcal{O}(n^2)$ subwords of y are in $\Sigma^* \setminus F$ which can be done in non-deterministic time $\mathcal{O}(n^2t(n))$ and non-deterministic space $\mathcal{O}(s(n))$ (as space for each query can be recovered). \square

Observe that we cannot necessarily replace $\Sigma^* \setminus F$ in the statement of the proposition by F as non-deterministic computation is involved, since we might then, in the worst case, have to brute-force search through all runs of length $t(n)$ of a non-deterministic machine for deciding F to find whether $z \notin F$. This would incur a time usage of $\mathcal{O}(2^n t(n))$ and—by Savitch’s Theorem [Sav70]—space usage of $\mathcal{O}((s(n))^2)$.

Given a decision procedure for $\mathcal{L}(X_F)$ with F anti-factorial, there exists a decision procedure for F using almost as few resources as that for $\mathcal{L}(X_F)$.

Proposition 9: *Let $T(n)$ and $S(n)$ be non-decreasing functions. Let F be anti-factorial. If $\mathcal{L}(X_F)$ is decidable in deterministic time $T(n)$ and deterministic space $S(n)$, then F is decidable in deterministic time $\mathcal{O}(T(n))$ and deterministic space $\mathcal{O}(S(n))$.* \square

Proof: By Proposition 2, it suffices, on input $y = y_1 \cdots y_n \in \Sigma^*$, to check whether $y \notin \mathcal{L}(X_F)$ and $y_1 \cdots y_{n-1}, y_2 \cdots, y_n \in \mathcal{L}(X_F)$. Thus, a total of three queries to a decision procedure for $\mathcal{L}(X_F)$ with words of length $\leq |y|$ which can obviously be done in time $\mathcal{O}(T(n))$ and space $\mathcal{O}(S(n))$. \square

4.3 Conversion results for β -Shifts

For β -shifts, it is unknown whether we can massage an arbitrary language L sufficiently to yield a language F coinciding with the set of forbidden words for a β -shift *and* retaining the computational hardness of L . Thus, the methods of the last two subsections would not yield much information, and we are forced to construct correspondences between the language, $\mathcal{L}(X_\beta)$ of the β -shift and some other construct having the twin advantages of (1) characterizing $\mathcal{L}(X_\beta)$ and (2) being amenable to construction using other languages whose computational complexity is known. As it turns out, a well-suited candidate for such a construct is the greedy expansion $d_\beta(1)$. As we shall only consider real numbers β such that $d_\beta(1)$ is not finite, we will always have $d_\beta(1) = d_\beta^*(1)$ in the following.

Lemma 2: *Let $\beta > 1$ be a real number and let $t(n)$ and $s(n)$ be non-decreasing functions. Assume that the following problem is decidable in deterministic time $t(n)$ (respectively, deterministic space $s(n)$):*

- Given: $x \in \{0, \dots, \lfloor \beta \rfloor\}^*$.
- To decide: whether $x = d_\beta^*(1)_1 \cdots d_\beta^*(1)_n$.

Then the following problem is decidable in deterministic time $\mathcal{O}(nt(n) + n^2)$ (respectively, deterministic space $\mathcal{O}(s(n) + n)$):

- Given: $x \in \{0, \dots, \lfloor \beta \rfloor\}^*$.
- To decide: whether $x \in \mathcal{L}(X_\beta)$.

□

Proof: For each $1 \leq k \leq n$, we build the word $d_\beta^*(1)_1 \cdots d_\beta^*(1)_k$ inductively: Set $d_\beta^*(1)_0 = \lambda$ and assume that $d_\beta^*(1)_1 \cdots d_\beta^*(1)_{k-1}$ has been constructed. In *descending* order, ask for each $b \in \{0, \dots, \lfloor \beta \rfloor\}$ if $d_\beta^*(1)_1 \cdots d_\beta^*(1)_{k-1} \cdot b \in \mathcal{L}(X_\beta)$. The first such b encountered satisfies $b = d_\beta^*(1)_k$.

Thus, we can construct $d_\beta^*(1)_1 \cdots d_\beta^*(1)_n$ in time $\mathcal{O}((\lfloor \beta \rfloor + 1)(t(1) + \cdots + t(n)) = \mathcal{O}(n \cdot t(n))$ (where the equality follows from non-decreasingness of $t(n)$). As space can be reused for each k , except for the space needed to hold $d_\beta^*(1)_1 \cdots d_\beta^*(1)_{k-1}$, the construction uses total space $\mathcal{O}(s(n) + n)$.

On input x , compare lexicographically, for all $0 \leq j \leq n$ the two words $\sigma^j(x)$ and $d_\beta^*(1)_1 \cdots d_\beta^*(1)_n$. A shift operation and subsequent lexicographical comparison takes time $\mathcal{O}(n)$ and space $\mathcal{O}(\log(n))$ to maintain counters (in addition to the $\mathcal{O}(n)$ space needed to hold the words in memory). The total resource usage of the comparisons is hence bounded above by time $\mathcal{O}(n^2)$ and space $\mathcal{O}(n)$.

Now, for $x \in \{0, \dots, \lfloor \beta \rfloor\}^n$, we have $x \in \mathcal{L}(X_\beta)$ iff for all $0 \leq j \leq n$, we have $\sigma^j(x) \leq_{\text{lex}} d_\beta^*(1)_1 \cdots d_\beta^*(1)_n$. We can perform each of the $n + 1$ lexicographic comparisons in time $\mathcal{O}(n)$ and space $\mathcal{O}(n)$ for a total resource usage for the comparisons of time $\mathcal{O}(n^2)$ and space $\mathcal{O}(n)$ (as space can be recovered).

We can hence decide whether $x \in \mathcal{L}(X_\beta)$ using total time $\mathcal{O}(nt(n) + n^2)$ and space $\mathcal{O}(s(n) + n)$. □

Examination of the proof of the above lemma reveals that it is in fact the *queries* to a decision procedure for the language $\{d_\beta^*(1)_1 \cdots d_\beta^*(1)_n : n \in \mathbb{N}\}$ that account for most of the resource usage. With that observation in hand, we may prove the following lemma.

Lemma 3: For all real numbers $\beta > 1$:

$$\mathcal{L}(X_\beta) \leq_T^p \{d_\beta^*(1)_1 \cdots d_\beta^*(1)_n : n \in \mathbb{N}\}$$

□

Proof: If $\beta \in \mathbb{Z}$, the result follows trivially, as $\mathcal{L}(X_\beta) = \{0, \dots, \beta - 1\}^*$, that is, all possible inputs are in $\mathcal{L}(X_\beta)$, whence $\mathcal{L}(X_\beta)$ is decidable in $\mathcal{O}(1)$.

If $\beta \notin \mathbb{Z}$, we reason as follows.

Assume that we have unit-time oracle access to $\{d_\beta^*(1)_1 \cdots d_\beta^*(1)_n : n \in \mathbb{N}\}$.

Let $x \in \{0, \dots, \lfloor \beta \rfloor\}^*$ with $|x| = n$.

By the first part of the proof of Lemma 2, the oracle will allow us to construct $d_\beta^*(1)_1 \cdots d_\beta^*(1)_n$ in time $\mathcal{O}(nt(n)) = \mathcal{O}(n)$.

By the second part of the proof of Lemma 2, when $d_\beta^*(1)_1 \cdots d_\beta^*(1)_n$ has been constructed, we may decide whether $x \in \mathcal{L}(X_\beta)$ using a further $\mathcal{O}(n^2)$ computation steps, hence there is a polynomial-time algorithm for deciding $\mathcal{L}(X_\beta)$, concluding the proof. \square

Corollary 1: For all real numbers $\beta > 1$: $\mathcal{L}(X_\beta) \in P/poly$. \square

Proof: Immediate by Lemma 3. \square

A converse of Lemma 2 is given in the below lemma.

Lemma 4: Let $t(n)$ and $s(n)$ be non-decreasing functions and let $\beta > 1$ be a real number. Assume that the following problem is decidable in deterministic time $t(n)$ (respectively deterministic space $s(n)$):

- Given: $x \in \{0, \dots, \lfloor \beta \rfloor\}^*$.
- To decide: Is $x \in \mathcal{L}(X_\beta)$?

Then the following problem is decidable in deterministic time $\mathcal{O}(nt(n))$ (respectively deterministic space $\mathcal{O}(s(n) + n)$).

- Given: $x \in \{0, \dots, \lfloor \beta \rfloor\}^*$.
- Is $x = d_\beta^*(1)_1 \cdots d_\beta^*(1)_n$?

\square

Proof: Set $d_\beta^*(1)_0 = \lambda$. For $0 < k < n$, if we have established $d_\beta^*(1)_1 \cdots d_\beta^*(1)_k$, we may establish $d_\beta^*(1)_1 \cdots d_\beta^*(1)_k d_\beta^*(1)_{k+1}$ as follows.

In *descending* order, for each $b \in \{0, \dots, \lfloor \beta \rfloor\}$, ask whether $d_\beta^*(1)_1 \cdots d_\beta^*(1)_k \cdot b \in \mathcal{L}(X_\beta)$. The first b with this property encountered must satisfy $b = d_\beta^*(1)_{k+1}$.

Thus, we need at most $\lfloor \beta \rfloor + 1$ queries to $\mathcal{L}(X_\beta)$; and each queried element is at most $k + 1$ symbols long. As we need to query for all $0 < k < n$ in succession (and we need to maintain the word $d_\beta^*(1)_1 \cdots d_\beta^*(1)_k$ in working memory, but can reuse the space used in each query), the above can be done in time $\mathcal{O}((\lfloor \beta \rfloor + 1)(t(1) + \cdots + t(n))) = \mathcal{O}(nt(n))$ and space $\mathcal{O}(s(n) + n)$. \square

Reasoning analogously to the proof of Lemma 3 allows us to prove the following corollary.

Corollary 2: $\{d_\beta^*(1)_1 \cdots d_\beta(1)_n : n \in \mathbb{N}\} \leq_T^p \mathcal{L}(X_\beta)$. \square

Proof: Inspecting the proof of Lemma 4, we see that at most n^2 (unit-time) queries to an oracle to $\mathcal{L}(X_\beta)$ are needed to establish $d_\beta^*(1)_1 \cdots d_\beta^*(1)_n$ (for each $0 < k < n$, we need at most a constant number of queries—corresponding to running through $\{0, \dots, \lfloor \beta \rfloor\}$ —to find $d_\beta^*(1)_{k+1}$). The overhead incurred by the algorithm of the proof of the lemma is clearly at most polynomial. \square

5 Hardness and completeness (and the lack thereof)

We now turn to the question of whether there are languages hard and complete for complexity classes under \leq_m^{logs} -reduction, hence *a fortiori* under \leq_m^p - and \leq_T^p -reduction), in particular hard and complete for well-known classes such as P, NP, EXPTIME, and so on. Unsurprisingly, given the conversion theorems we have established previously in this paper, the answer to the question is ‘yes’ for general symbolic dynamical systems. For β -shifts, however, it turns out that *if* the answer is ‘yes’, we immediately obtain that a number of well-known conjectures whose proofs (or disproofs) have been long-standing open problems, turn out to be *false*.

5.1 Hardness of languages of symbolic dynamical systems

For symbolic dynamical systems, we have the following.

Proposition 10: *If $L \subsetneq \Sigma^*$, then $L \leq_m^{\text{logs}} L^\# \leq_m^{\text{logs}} (\Sigma \cup \{\#\})^* \setminus \mathcal{L}(X_{L^\#})$.* \square

Proof: On input $x \in \Sigma^*$, outputting $\#x\#$ can be performed in constant (and hence logarithmic) space: One simply outputs $\#$, copies the input to the output, and suffixes $\#$.

We have $x \in L$ iff $\#x\# \in L^\#$, and thus $L \leq_m^{\text{logs}} L^\#$.

We have $L^\# \subsetneq (\Sigma \cup \{\#\})^*$, and there is thus $t \in (\Sigma \cup \{\#\})^*$ such that $t \notin \mathcal{L}(X_{L^\#})$.

We construct a logarithmic-space transformation as follows:

On input $y \in (\Sigma \cup \{\#\})^*$, we can check (in constant space) whether y is on the form $\#y'\#$ where $y' \in \Sigma^*$. If y is on this form, output y . If y is *not* on this form, output t . Clearly, this construction takes at most constant (hence at most logarithmic) space, as t is fixed.

We claim that $y \in L^\#$ iff the output of the above transformation is in $(\Sigma \cup \{\#\})^* \setminus \mathcal{L}(X_{L^\#})$. To see this, note that if y is on the form $\#y'\#$ where $y' \in \Sigma^*$, we have $y \in L^\#$ iff $y \notin \mathcal{L}(X_{L^\#})$ —as y does not contain any other elements of $L^\#$ as a subword and y thus occurs as a subword of an element of $X_{L^\#}$ (for instance, any element on the form $\cdots aayaaaa \cdots$ where $a \in \Sigma$). If y is not on this form, then $y \notin L^\#$, and the output, t , of the transformation in this case by construction satisfies $t \notin \mathcal{L}(X_{L^\#})$. \square

By the above proposition and transitivity of \leq_m^{logs} , if L is \leq_m^{logs} -hard for a complexity class \mathcal{C} , so are $L^\#$ and the complement of $\mathcal{L}(X_{L^\#})$.

Corollary 3: *For every alphabet Σ and every complexity class \mathcal{C} with an \leq_m^{logs} -hard set over alphabet Σ , there exists an anti-factorial set over $\Sigma \cup \{\#\}$ that is \leq_m^{logs} -hard for \mathcal{C} .* \square

Proof: Choose a \leq_m^{logs} -hard set, L over alphabet Σ , for \mathcal{C} . By Proposition 10, we obtain that $L^\#$ is a \leq_m^{logs} -hard set for the class. \square

Corollary 4: *For every alphabet Σ with $|\Sigma| \geq 3$, there are anti-factorial sets over Σ that are \leq_m^{logs} -hard for P , NP , coNP , $PSPACE$, $EXPTIME$.* \square

Proof: By the previous corollary, noting that all of the mentioned complexity classes have \leq_m^{logs} -hard languages over any two-letter alphabet. \square

Corollary 5: *For every alphabet Σ and every complexity class \mathcal{C} , consider the class $\text{co}\mathcal{C}$ consisting of the complements (in Σ^*) of sets in \mathcal{C} . If there is an \leq_m^{logs} -hard set over alphabet Σ for $\text{co}\mathcal{C}$, there exists a factorial, extensible language over $\Sigma \cup \{\#\}$ that is \leq_m^{logs} -hard for \mathcal{C} .* \square

Proof: Choose a \leq_m^{logs} -hard set, C over alphabet Σ , for $\text{co}\mathcal{C}$. By Proposition 10, the complement of $\mathcal{L}(X_{C^\#})$ (in $(\Sigma \cup \{\#\})^*$) is \leq_m^{logs} -hard for $\text{co}\mathcal{C}$.

By transitivity of \leq_m^{logs} , for any M in \mathcal{C} , there is a logspace-computable map f such that for all $y \in (\Sigma \cup \{\#\})^*$, we have $f(y) \in (\Sigma \cup \{\#\})^* \setminus \mathcal{L}(X_{C^\#})$ iff $y \in (\Sigma \cup \{\#\})^* \setminus M$, that is, $f(y) \in \mathcal{L}(X_{C^\#})$ iff $y \in M$, showing that $\mathcal{L}(X_{C^\#})$ is \leq_m^{logs} -hard for \mathcal{C} . \square

Theorem 3: *For every alphabet Σ with $|\Sigma| \geq 3$, there are \leq_m^{logs} -hard factorial, extensible sets for P , NP , coNP , $PSPACE$ and $EXPTIME$.* \square

Proof: By the preceding corollary, noting that there exist \leq_m^{logs} -hard sets for the co-classes for all of the mentioned classes. \square

5.2 Completeness of languages of symbolic dynamical systems

Having established hardness, we now turn to completeness. For deterministic complexity classes, this turns out to be straightforward; for non-deterministic classes, a little more work is required.

Proposition 11: *For every alphabet Σ with $|\Sigma| \geq 3$, there are \leq_m^{logs} -complete anti-factorial sets for P , NP , $coNP$, $PSPACE$, $EXPTIME$, etc.* \square

Proof: Let L be a \leq_m^{logs} -complete language for the class under consideration and assume that L is decidable in time $t(n)$ (respectively space $s(n)$) where $t(n)$ (respectively $s(n)$) is within the time (space) requirement for inclusion in the considered class.

The construction of Corollary 4 will transform L into a \leq_m^{logs} -hard anti-factorial language $L^\#$ that, by Proposition 5 takes time $\mathcal{O}(t(n) + n)$ and space $\mathcal{O}(s(n))$ to decide. As all of the considered classes have time (space) requirements closed under multiplication by constants and added polynomial overhead, we obtain the result. \square

To move from anti-factorial to factorial, extensible languages, we first consider *deterministic* complexity classes (thus, in particular, *not* the classes NP and $coNP$).

Theorem 4: *For every alphabet Σ with $|\Sigma| \geq 3$, there are symbolic dynamical systems with \leq_m^{logs} -complete languages for P , $PSPACE$, and $EXPTIME$.* \square

Proof: Employ the construction of the proof of Corollary 5. We only need to prove that if the set \mathcal{C} of that proof is in $co\mathcal{C}$, then $\mathcal{L}(X_{L^\#})$ is in \mathcal{C} . By Proposition 7 and the fact that the considered complexity classes are closed under multiplication of the time (space) measure by a polynomial, then $\mathcal{L}(X_{L^\#})$ is in $co\mathcal{C}$. As $co\mathcal{C} = \mathcal{C}$ for deterministic classes, we obtain the desired conclusion. \square

In principle, there is no reason to stop at $EXPTIME$, as the proof of the above theorem goes through for all deterministic classes whose resource measure is closed under multiplication by polynomials. For the purpose of clarity, we have chosen to focus on the well-known complexity classes in the backbone hierarchy, though.

For $|\Sigma| = 2$, the methods employed above do not work. We strongly conjecture that there are symbolic dynamical systems with languages over $\{0, 1\}$ that are hard for *all* of the usual complexity classes (see also Section 8).

Consider now the case of non-deterministic complexity classes NP and $coNP$. It turns out that we need to reason slightly differently than in the case with deterministic classes.

Theorem 5: *For every alphabet Σ with $|\Sigma| \geq 3$, there exists a symbolic dynamical system with \leq^{logs} -complete language for NP , resp. an symbolic dynamical system with \leq_m^{logs} -complete language for $coNP$.* \square

Proof: Choose a set C that is \leq_m^{logs} -hard for $coNP$. The construction of the proof of Corollary 5 gives that $\mathcal{L}(X_{C^\#})$ is NP -hard. We thus need only prove that $\mathcal{L}(X_{C^\#}) \in NP$, but this follows immediately from Proposition 8 and the fact that NP is closed under multiplication of time resource usage by any polynomial. The proof establishing an symbolic dynamical system with $coNP$ -complete language proceeds in the same way, *mutatis mutandis*. \square

5.3 β -shifts (probably) do not have hard languages

Having shown the existence of languages of symbolic dynamical systems hard and complete for the most well-known complexity classes, we now consider the special case of the β -shifts.

Recall the following famous result by Karp and Lipton [KL80]: If there is a \leq_T^p -hard set for NP ($PSPACE, EXPTIME$) that is contained in $P/poly$, then $PH = \Sigma_2^p$ ($PSPACE = \Sigma_2^p, EXPTIME = \Sigma_2^p$). We now immediately obtain:

Theorem 6: *If there exists $\beta > 1$ such that $\mathcal{L}(X_\beta)$ is a \leq_T^p hard set for $NP(PSPACE, EXPTIME)$, then $PH = \Sigma_2^p$ ($PSPACE = \Sigma_2^p, EXPTIME = \Sigma_2^p$).* \square

Proof: By Corollary 1, $\mathcal{L}(X_\beta) \in P/poly$, and the result follows immediately from Karp-Lipton Theorem above. \square

6 Complexity Hierarchies

In this section we establish hierarchies of symbolic dynamical systems and β -shifts where the systems are ranked by the hardness of deciding their respective languages.

We consider hierarchies of sets decidable by deterministic Turing machines and leave corresponding hierarchies for non-deterministic machines for future work.

To begin, observe that anti-factorial languages over a unary alphabet must have exactly one element, and if L is an infinite, factorial language over a unary alphabet Σ , then $L = \Sigma^*$. Hence, for unary alphabets, there can be no complexity hierarchies, neither for time, nor for space bounds, in the usual sense when $|\Sigma| = 1$. We will use our previous constructions to consider alphabets of size $|\Sigma| \geq 3$ for time bounds and alphabets of size $|\Sigma| \geq 2$ for space bounds.

6.1 Hierarchies for general symbolic dynamical systems

We have the following.

Proposition 12: *For every alphabet Σ with $|\Sigma| \geq 3$ and for every time-constructible $t(n) = \omega(n)$, there is an anti-factorial language decidable in deterministic time $\mathcal{O}(t(n))$, but not in deterministic time $o(t(n))$.* \square

Proof: Let Σ be an alphabet containing at least two elements. By Theorem 1, we obtain a language $L \subseteq \Sigma^*$ such that L is decidable by a Turing machine in time $t(n)$ but not in any time bound which is $o(t(n))$. By Proposition 5, $L^\#$ is decidable in time $\mathcal{O}(t(n) + n) = \mathcal{O}(t(n))$, and $L^\#$ is antifactorial by Proposition 4. If $L^\#$ were decidable in time $g(n) = o(t(n))$, Proposition 6 yields that L would be decidable in time $\mathcal{O}(g(n) + n) = o(t(n))$, a contradiction. \square

Note that the proposition is somewhat weaker than the tight Theorem 1 as we must require that $t(n)$ grows superlinearly.

We now obtain a hierarchy theorem for the languages of symbolic dynamical systems.

Theorem 7: *Let Σ be an alphabet with $|\Sigma| \geq 3$. For every proper time bound $t(n) \in \omega(n)$, there is a symbolic dynamical system with language over Σ decidable in deterministic time $\mathcal{O}(n^2 t(n))$, but not in deterministic time $o(t(n))$.* \square

Proof: Proposition 12 yields existence of an anti-factorial language F decidable in time $\mathcal{O}(t(n))$ but not in any time bound which is $o(t(n))$. Consider the symbolic dynamical system X_F and its language $\mathcal{L}(X_F)$. Proposition 7 yields that $\mathcal{L}(X_F)$ is decidable in time $\mathcal{O}(n^2 t(n))$.

Were $\mathcal{L}(X_F)$ decidable in time $o(t(n))$, then Proposition 9 yields that F would be decidable in time $o(t(n))$, contradicting the assumptions on F . \square

Thus, we for instance obtain a hierarchy of symbolic dynamical systems with languages decidable in time $\mathcal{O}(n^3 \log n)$, $\mathcal{O}(n^3 \log^2 n)$, $\mathcal{O}(n^3 \log^3 n)$, et cetera, where the languages at each level cannot be decided in any time bound at a lower level of the hierarchy.

One drawback of Theorem 7 is the rather hairy lower bound on asymptotic time at which the hierarchies start: In essence, Theorem 7 does very little to differentiate time bounds below cubic time.

For space, we can obtain a tighter hierarchy.

Proposition 13: *For every alphabet Σ with $|\Sigma| \geq 2$ and for every space-constructible $s(n) = \omega(n)$, there is an anti-factorial language decidable in deterministic space $\mathcal{O}(s(n))$, but not in deterministic space $o(s(n))$.* \square

Proof: Without loss of generality we may write $\Sigma = \{0, 1\}$. Theorem 2 ensures existence of a language $L \subseteq \{0\}^*$ that is decidable in deterministic space $\mathcal{O}(s(n))$, but not in deterministic space $o(s(n))$.

By Proposition 5, $L^\#$ is decidable in space $\mathcal{O}(s(n) + n) = \mathcal{O}(s(n))$, and $L^\#$ is antifactorial by Proposition 4. If $L^\#$ were decidable in space $g(n) = o(s(n))$, Proposition 6 yields that L would be decidable in space $\mathcal{O}(g(n) + n) = o(s(n))$, a contradiction. \square

Theorem 8: *For every alphabet Σ with $|\Sigma| \geq 2$ and for every proper space bound $s(n) = \omega(n)$, there is a symbolic dynamical system with language L over Σ decidable in deterministic space $\mathcal{O}(s(n))$, but not in deterministic space $o(s(n))$.* \square

Proof: Proposition 13 yields existence of an anti-factorial language F decidable in space $\mathcal{O}(s(n))$ but not in any space bound which is $o(s(n))$. Consider the symbolic dynamical system X_F and its language $\mathcal{L}(X_F)$. Proposition 7 yields that $\mathcal{L}(X_F)$ is decidable in space $\mathcal{O}(s(n))$.

Were $\mathcal{L}(X_F)$ decidable in space $o(s(n))$, then Proposition 9 yields that F would be decidable in space $o(s(n))$, contradicting the assumptions on F . \square

6.2 Hierarchies for β -shifts

The straightforward way to construct languages $\mathcal{L}(X_\beta)$ is by considering the greedy expansion $d_\beta^*(1)$ of 1 in powers of β^{-1} , in particular by considering the language $\{d_\beta^*(1)_1 \cdots d_\beta^*(1)_n : n \in \mathbb{N}\}$, cf. Lemma 3. This leaves us in a quandary when considering complexity hierarchies as $\{d_\beta^*(1)_1 \cdots d_\beta^*(1)_n : n \in \mathbb{N}\}$ is extremely sparse: It contains exactly one word of each length n —and unfortunately, the obvious way of constructing complexity hierarchies is to work with the languages of already existing hierarchy theorems, languages that are invariably *not* sparse.

Furthermore, the added requirement for β -shifts that $\forall j \in \mathbb{N}. \sigma^j(d_\beta(1)) \leq_{\text{lex}} d_\beta(1)$ complicates any translation of existing hierarchy results. Due to these problems, we have had to adopt a fairly pedestrian solution that—for time complexity—decodes binary languages to integers which are then used to construct large “gaps”—sequences of 0s—between successive occurrences of 1s in $d_\beta^*(1)$ for $1 < \beta < 2$. While this does establish a hierarchy result for time complexity (Theorem 9), the gulfs between successive levels of the hierarchies are, unfortunately, quite large.

For space complexity, the situation is much better due to the availability of a hierarchy theorem for unary languages (Theorem 2), and we are thus able to derive a somewhat tighter hierarchy theorem for space complexity of the languages of β -shifts (Theorem 10).

Let $\xi : \{0, 1\}^* \rightarrow \mathbb{N}$ be a map decoding a binary word to the integer it represents in the natural fashion, that is, $\xi(0) = 0$, $\xi(1) = \xi(01) = 1$, $\xi(10) = 2$, et cetera. Prefixed zeroes are ignored.

Proposition 14: *The function $g_\xi : \{0, 1\}^* \rightarrow \{0\}^*$ defined by $g_\xi(\lambda) = \lambda$ and $g_\xi(x) \triangleq 0^{\xi(x)}$ is computable in time $\mathcal{O}(\xi(x))$ and space $\mathcal{O}(\xi(x))$. The inverse, $g_\xi^{-1} : \{0\}^* \rightarrow \{0, 1\}^*$, of g_ξ is computable in time $\mathcal{O}(n)$ and space $\mathcal{O}(\log n)$. \square*

Proof: Converting binary to unary on a multi-tape Turing machine can obviously be done in time and space proportional to the length of the unary output—in this case $\mathcal{O}(\xi(x))$.

Converting from unary to binary can be done in time proportional to the length of the unary input and space proportional to the binary output by performing a single pass over the unary input word while constructing the binary output by adding one to the binary representation on an auxiliary tape for each memory cell in the input tape (containing the unary word). This incurs a total time cost of $\mathcal{O}(n)$ and space cost of $\mathcal{O}(\log n)$. \square

Given an infinite set B of binary words, we now construct an infinite binary word with successively greater gaps between successive occurrences of ‘1’ where the size of the gaps are based on the elements of B in their natural ordering.

Definition 11: *Let $B \subseteq \{0, 1\}^*$ be infinite and consider $\xi(B) = \{\xi(x) : x \in B\}$. Let the elements of $\xi(B)$ written in increasing order be $\xi(x_1) < \xi(x_2) < \dots$. Define $x_B \in \{0, 1\}^{\mathbb{N}}$ by:*

$$x_B \triangleq 10^{\xi(x_1)}10^{\xi(x_2)}1\dots$$

\square

Observe that we can also write x_B as $1g_\xi(x_1)1g_\xi(x_2)1\dots$.

Note also that $\sigma^j(x_B) <_{\text{lex}} x_B$ for all $j > 0$. By a fundamental result of Parry [Par60], there exists a real number $1 < \beta_B < 2$ such that $x_B = d_{\beta_B}(1)$. By the fact that B is infinite, x_B does not end in an infinite word of zeroes, whence $d_{\beta_B}(1)$ is not finite and we thus have $d_{\beta_B}^*(1) = d_{\beta_B}(1)$.

Proposition 15: *Let $t(n)$ be a non-decreasing function. If B is decidable in deterministic time $\mathcal{O}(t(n))$, then the following problem is decidable in deterministic time $\mathcal{O}(nt(\log n + 1))$.*

- Given: $y \in \{0, 1\}^*$
- To decide: Is y a prefix of x_B ?

\square

Proof: On input y , establish the prefix z of x_B of length $|y|$ by iteratively ascertaining whether $g_\xi^{-1}(\lambda)$, $g_\xi^{-1}(0)$, $g_\xi^{-1}(00)$, \dots , $g_\xi^{-1}(0^{|y|-1})$ are in B . Subsequently compare this prefix to y .

Computing each of the $g_\xi^{-1}(0^i)$ for $0 \leq i \leq |y| - 1$ can be done in time $\mathcal{O}(i) = \mathcal{O}(|y|) = \mathcal{O}(n)$ by Proposition 14 and non-decreasingness of $t(n)$.

Note that there are $|y| = \lfloor \log y \rfloor + 1$ elements in the list $0, 1, \dots, |y| - 1$, and by monotonicity of $t(n)$, the queries establishing z can thus be performed in time

$$\mathcal{O}(|y| \cdot t(|g_\xi^{-1}(0^{|y|}) - 1|)) = \mathcal{O}(n \cdot t(|g_\xi^{-1}(0^{|y|}) - 1|)) = \mathcal{O}(n \cdot t(\lfloor \log(|y|) \rfloor + 1)) = \mathcal{O}(nt(\log n + 1))$$

Accounting for linear-time overhead to perform concatenations, perform comparison of y with z , and other household tasks, we can thus ascertain whether y is a prefix of x_B in time $\mathcal{O}(nt(\log n + 1))$. \square

Proposition 16: *Let $T(n) \geq n$ be a non-decreasing function. If the set*

$$\{y \in \{0, 1\}^* : y \text{ is a prefix of } x_B\}$$

is decidable in deterministic time $\mathcal{O}(T(n))$, then the following problem is decidable in deterministic time $\mathcal{O}(4^n T(4^n))$:

- Given: $x \in \{0, 1\}^*$.
- To decide: Is $x \in B$?

\square

Proof: By construction of x_B , we have $x \in B$ iff there is a subword on the form $10^{\xi(x)}1 = 1g_\xi(x)1$ in some prefix of x_B . Let K be the (possibly empty) set of prefixes of x_B on the form $s \cdot 1g_\xi(x)1$ and note that by construction of x_B , $g_\xi(x)$ occurs at most once in each prefix.

The longest possible element of K is $p \triangleq 10^1 10^2 10^3 1 \dots 10^{\xi(x)} 1$, which is of length $M = \xi(x)(\xi(x) + 1)/2 + \xi(x) + 1$. Hence, constructing the prefix of x_B of length M and subsequently checking whether it contains $1g_\xi(x)1$ as a subword is sufficient to ascertain whether $x \in B$.

We can establish the prefix of length M by maintaining a temporary word *temp* in memory as follows: Start by setting *temp* $\triangleq 1$ and obtain a prefix of x_B one bit longer than before by asking whether *temp*·0 is a prefix of x_B (if not, then *temp*·1 is), and update *temp* accordingly. Continue until $|temp| = M$, in which case *temp* is the prefix of x_B of length M .

The time to establish the prefix of length M of x_B by the procedure above is bounded above by $\mathcal{O}(\sum_{j=1}^M t(j)) = \mathcal{O}(MT(M))$ by non-decreasingness of $T(n)$. As $M \leq (\xi(x))^2$ for $\xi(x) \geq 4$, and as $\xi(x) \geq 2^{|x|}$ in the worst case, the time usage is $\mathcal{O}(4^{|x|} T(4^{|x|})) = \mathcal{O}(4^n T(4^n))$.

When the prefix of x_B is established, ascertaining whether the word $1g_\xi(x)1$ is contained within the prefix can be done in time $\mathcal{O}(M)$. Total time use for deciding whether $x \in B$ is thus $\mathcal{O}(4^{|x|} T(4^{|x|}))$. \square

Lemma 5: *Let $\beta > 1$ be a real number and let $t(n)$ be a non-decreasing function. Define Problem C as:*

- Given: $x \in \{0, 1\}^*$.
- To decide: Is $x \in B$?

And define Problem D as:

- Given: $x \in \{0, 1\}^*$.
- To decide: is $x \in \mathcal{L}(X_{\beta_B})$?

If Problem C is decidable in deterministic time $\mathcal{O}(t(n))$, then Problem D is decidable in deterministic time $\mathcal{O}(n^2t(\log n + 1))$

Conversely, if Problem D is decidable in deterministic time $\mathcal{O}(T(n))$, then Problem C is decidable in deterministic time $\mathcal{O}(16^n T(4^n))$. \square

Proof: Assume that Problem C is decidable in time $\mathcal{O}(t(n))$. It then follows from Proposition 15 and Lemma 2 that Problem D is decidable in time $\mathcal{O}(n(nt(\log n + 1) + n^2) = \mathcal{O}(n^2t(\log n + 1) + n^2) = \mathcal{O}(n^2t(\log n + 1))$.

Conversely, assume that Problem D is decidable in time $\mathcal{O}(T(n))$. It follows from Lemma 4 and Proposition 16 that Problem C is decidable in time $\mathcal{O}(4^n \cdot 4^n T(4^n)) = \mathcal{O}(16^n T(4^n))$. \square

We can now obtain a — fairly weak — hierarchy result for β -shifts:

Theorem 9: For every time-constructible, non-decreasing function $t(n) \geq n$, there is a real number $1 < \beta < 2$ such that $\mathcal{L}(X_\beta)$ is decidable in deterministic time $\mathcal{O}(n^2t(\log n + 1))$, but not in deterministic time $g(n)$ where $g(n)$ is any function satisfying $g(4^n) = o(t(n)/16^n)$. \square

Proof: For every time-constructible, non-decreasing function $t(n) \geq n$, Theorem 1 yields $B \subseteq \{0, 1\}^*$ decidable in time $\mathcal{O}(t(n))$, but not in any time bound which is $o(t(n))$. By Lemma 5, $\mathcal{L}(X_{\beta_B})$ is decidable in time $\mathcal{O}(n^2t(\log n))$. Were $\mathcal{L}(X_{\beta_B})$ decidable in time $g(n)$ with $g(4^n) = o(t(n)/16^n)$, then by Lemma 5, B would be decidable in time $\mathcal{O}(16^n g(4^n)) = o(16^n t(n)/16^n) = o(t(n))$, an impossibility. \square

The statement of the previous theorem is somewhat convoluted compared to the usual crisp and intuitive hierarchy results. However, the main point is that there *is* a hierarchy, its apparent non-tightness notwithstanding. For instance, when starting from $t(n) = 64^n \cdot n$, we obtain that there is a real number $1 < \beta < 2$ with $\mathcal{L}(X_\beta)$ decidable in time $\mathcal{O}(n^2 64^{\log n} \log n) = \mathcal{O}(n^3 \log n)$ that is not decidable in time $g(n) = n$, as $4^n = o(64^n \cdot n/16^n)$, a $1 < \beta' < 2$ with $\mathcal{L}(X_{\beta'})$ decidable in exponential time, but not in time $\mathcal{O}(n^3 \log n)$, etc.

We now turn to hierarchies of space complexity. Analogously to Definition 11, consider the following definition.

Definition 12: Let $C \subseteq \{0\}^*$ be infinite and write the elements of C in increasing order of their length as $|x_1| < |x_2| < \dots$. Define $x_C \in \{0, 1\}^{\mathbb{N}}$ by:

$$x_C \triangleq 1x_11x_21\dots$$

\square

Again, observe that for all $j \in \mathbb{N}$, we have $\sigma^j(x_C) \leq_{\text{lex}} x_C$ and thus that $x_C = d_{\beta_C}(1)$ for some $1 < \beta_C < 2$.

Proposition 17: Let $s(n)$ be a non-decreasing function. If $C \subseteq \{0\}^*$ is decidable in deterministic space $s(n)$, then the following problem is decidable in deterministic space $\mathcal{O}(s(n) + n)$:

- Given: $y \in \{0, 1\}^*$.
- To decide: Is y a prefix of x_C ?

\square

Proof: On input y , establish the prefix, z of length $|y|$ of x_C by iteratively ascertaining whether $0^0, 0^1, 0^2, \dots, 0^{|y|-1}$ are in C . Subsequently compare this prefix to y . We can construct the 0^i in space $\mathcal{O}(|y|)$. The space needed for each query to C can be reused and is $\mathcal{O}(s(|y|)) = \mathcal{O}(s(n))$ by the assumption that

$s(n)$ is non-decreasing. Holding the prefix z in memory requires space $\mathcal{O}(|y|)$, for a total space usage of $\mathcal{O}(s(n) + n)$. \square

Proposition 18: *Let $S(n)$ be a non-decreasing function. If the set*

$$\{y \in \{0, 1\}^* : y \text{ is a prefix of } x_C\}$$

is decidable in space $S(n)$, then the following problem is decidable in space $\mathcal{O}(S(n^2) + n^2)$.

- *Given: $x \in \{0\}^*$*
- *To decide: Is $x \in C$.*

\square

Proof: To find whether $x \in C$, we need to ascertain whether there is a word on the form $1x1 \cdots$ in some prefix of x_C . Let K be the set of prefixes of x_C on the form $s \cdot 1x1$ and note that by construction of x_C , $1x1$ occurs at most once in any such prefix. The longest possible element of K is $p \triangleq 1010010001 \cdots 1x1$ which is of length $M \triangleq |x|(|x| + 1)/2 + |x| + 1$. Hence, constructing p and subsequently checking whether it contains $1x1$ as a subword is sufficient to ascertain whether $x \in C$. The space needed to construct this prefix is the space needed to hold each prefix of length at most M , that is, $\mathcal{O}(M) = \mathcal{O}(|x|^2)$, plus the space needed for each query to the set $\{y \in \{0, 1\}^* : y \text{ is a prefix of } x_C\}$. The latter space usage is by assumption bounded above by $\mathcal{O}(S(M)) = \mathcal{O}(S(|y|^2))$, and as usual the space used for each query can be reused. The total space usage is thus $\mathcal{O}(S(n^2) + n^2)$. \square

The space hierarchy result can now be proved.

Theorem 10: *For every space-constructible, non-decreasing function $s(n) = \omega(n^2)$, there is a real number $1 < \beta < 2$ such that $\mathcal{L}(X_\beta)$ is decidable in space $\mathcal{O}(s(n))$, but not in space $g(n)$ where $g(n)$ is any function satisfying $g(n^2) = o(s(n))$.* \square

Proof: By Theorem 2, let $C \subseteq \{0\}^*$ be such that C is decidable in space $\mathcal{O}(s(n))$ but not in space $o(s(n))$. By Proposition 17 and Lemma 2, $\mathcal{L}(X_{\beta_C})$ is decidable in space $\mathcal{O}(s(n) + n)$.

Assume, for the purpose of contradiction, that $\mathcal{L}(X_{\beta_C})$ were decidable in a space bound $g(n)$ such that $g(n^2) = o(s(n))$. By Lemma 4 and Proposition 18, C would then be decidable in space $\mathcal{O}(g(n^2) + n^2 + n) = o(s(n))$, a contradiction. \square

Thus, when starting from, say, a space bound of $n^2 \log n$, we could obtain a hierarchy $n^2 \log n, n^4 \log^3 n, n^8 \log^7 n$, etc. such that for each level in the hierarchy, there were real numbers $1 < \beta < 2$ with languages decidable with space resources at said level, but at none of the space resources of the lower levels. We also immediately obtain that there is a real number $1 < \beta < 2$ such that $\mathcal{L}(X_\beta)$ is decidable in space $\mathcal{O}(2^n)$, but $\mathcal{L}(X_\beta) \notin \text{PSPACE}$.

As a further consequence, we can give a positive answer to the open question of Johnson of whether there exists a β -shift with recursive, but not context-sensitive language [Joh99, Sec. 4.5.5].

Theorem 11: *There exists a real number $1 < \beta < 2$ such that $\mathcal{L}(X_\beta)$ is recursive, but not context-sensitive.* \square

Proof: Theorem 10 gives an infinite number of decidable languages of β -shifts, in particular an infinite number of decidable languages *not* decidable in space $\mathcal{O}(2^n)$. Fix any such language $\mathcal{L}(X_\beta)$. The problem “given a context-sensitive grammar G over alphabet Σ and $w \in \Sigma^*$, decide whether $w \in L(G)$ ” is PSPACE-complete when the size of the input is measured as $|G| + |w|$ [GJ79], hence there exists a polynomial-space algorithm—and thus *a fortiori* an exponential-space algorithm for the problem.

Assume, for the purpose of contradiction, that K were context-sensitive. Then there would exist a context-sensitive-grammar G with $L(G) = \mathcal{L}(X_\beta)$ and by the above observation we could, for each $w \in \{0, 1\}^*$ decide in space $\mathcal{O}(2^{|G|+|w|})$ whether $w \in \mathcal{L}(X_\beta)$. As G can be chosen to be fixed once $\mathcal{L}(X_\beta)$ is fixed, the algorithm above runs in space $\mathcal{O}(2^{|G|+|w|}) = \mathcal{O}(2^{|G|} \cdot 2^{|w|}) = \mathcal{O}(2^{|w|})$. But then $\mathcal{L}(X_\beta)$ would be decidable in space $\mathcal{O}(2^{|w|}) = \mathcal{O}(2^n)$, which is impossible by construction. \square

7 Symbolic Dynamical Systems with Decidable Languages not in P/poly

As any sparse language is in P/poly, and any singleton set is both anti-factorial and sparse, we immediately obtain:

Proposition 19: *For any non-empty alphabet Σ , there are both factorial and anti-factorial languages over Σ in P/poly.* \square

The nature of factorial languages tends to make them either very “fat” (contain *many* words, due to closure under subwords), or very “meager” (e.g. languages over a unary alphabet). It is thus not obvious how to construct decidable factorial and extensible languages *not* in P/poly.

However, two straightforward results are:

Proposition 20: *For any language where $L, L^\# \in P/poly$, we have $L \in P/poly$,* \square

Proof: The construction of $L^\#$ implies that we can decide L in polynomial time given an oracle to $L^\#$ (as $x \in L$ iff $\#x\# \in L^\#$). Thus, by Lemma 1, $L^\# \in P/poly$ implies $L \in P/poly$. \square

By contraposition, we thus have that $L \notin P/poly$ implies $L^\# \notin P/poly$.

Proposition 21: *For any language L where $\mathcal{L}(X_{L^\#}) \in P/poly$, we have $L^\# \in P/poly$.* \square

Proof: By Lemma 1, Proposition 10, and the fact (1) that $A \in P/poly$ implies $\Sigma^* \setminus A \in P/poly$, and (2) that $A \leq_m^{logs} B$ implies $A \leq_T^p B$. \square

We could use Propositions 20 and 21 to find anti-factorial sets and symbolic dynamical systems with decidable languages not in P/poly by finding appropriate decidable languages L over $\{0,1\}$ in P/poly and constructing $L^\#$. This would leave the case of alphabet size $|\Sigma| = 2$ open.

As it turns out, we can take another route, as there is a simple, direct proof of existence of a decidable, anti-factorial language over $\{0,1\}$. To establish this fact, we use the characterization of P/poly as the class of languages decidable by boolean circuits of polynomial size: For any $A \subseteq \{0,1\}^*$, $A \in P/poly$ iff there is a polynomial P such that for each $n \in \mathbb{N}$, there is a boolean circuit of size at most $P(n)$ that accepts exactly $A \cap \{0,1\}^n$ [KD00].

We first need a lemma:

Lemma 6: *Set $F \triangleq \{11\}$. There is a decidable language $J \subseteq \{0,1\}^*$ such that $J \subseteq \mathcal{L}(X_F)$ and $J \notin P/poly$.* \square

Proof: By standard results in dynamical systems theory [LM95], the number of words of length $n \geq 1$ in $\mathcal{L}(X_{\{11\}})$ of length is the $(n+2)$ th Fibonacci number $F(n+2)$ where:

$$F(n) \triangleq \left\lceil \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n \right\rceil$$

In the above, we have used brackets $\lceil \cdot \rceil$ to denote the Nearest Integer Function.

Clearly, every polynomial $P(n)$ is $o(F(n+2))$, and $F(n) \leq 2^n$ for all $n \in \mathbb{N}$. The total number of boolean circuits of n variables and size at most s is known to be bounded above by $s(2(2+s+2n)^2)^s$, see e.g. [KD00, Thm. 6.1].

For $s = F(n)/4n$, we then have:

$$\begin{aligned}
s(2(2+s+2n)^2)^s &\stackrel{a.e.}{\leq} \frac{F(n)}{4n} \left(2 \left(\frac{F(n)}{2n} \right)^2 \right)^{\frac{F(n)}{4n}} \\
&\leq \frac{F(n)}{4n} (2^n)^{\frac{F(n)}{2n}} \\
&= \frac{F(n)}{4n} 2^{\frac{F(n)}{2}} \\
&\leq 2^n \cdot 2^{F(n)/2} \\
&= 2^{F(n)/2+n}
\end{aligned}$$

There are $2^{F(n+2)}$ possible boolean functions of $F(n+2)$ variables. Hence, we can compute an $N \in \mathbb{N}$ such that for $n > N$, there are $2^{F(n-2)} - 2^{F(n+2)/2+n+2} > 0$ boolean functions of $F(n+2)$ variables that have circuit size greater than $F(n+2)/4(n+2)$. We can thus choose, for each sufficiently large $n \in \mathbb{N}$ a (necessarily finite) subset J_n of $\mathcal{L}(X_F) \cap \{0,1\}^n$ such that the circuit size of the subset is at least $F(n+2)/4(n+2)$ (for the actual computation, we can obtain J_n by brute force search through all subsets and small boolean circuits). Set $J_n = \emptyset$ for $n \leq N$ and set $J = \bigcup_{n \in \mathbb{N}} J_n$. As $F(n+2)$ grows faster than any polynomial, so does $F(n+2)/4(n+2)$, and thus J cannot be decided by boolean circuits of polynomial size, whence $J \notin \text{P/poly}$. However, we can construct an algorithm that, on input $x \in \{0,1\}^n$ computes J_n and checks whether $x \in J_n$, whence J is decidable, as desired. \square

To obtain a language of a dynamical system, we will use the set J of the previous lemma to construct an anti-factorial language:

Definition 13: *Let J be the language of Lemma 6. We define \tilde{J} as the language containing the word $011^n 0x$ iff $x \in J$.* \square

Proposition 22: *\tilde{J} is decidable and anti-factorial.* \square

Proof: Decidability of \tilde{J} follows immediately from decidability of J . As J by construction does not contain any word $z \in \{0,1\}^*$ such that the word 11 is a subword of z , the only way that a word $011^m 0x$ can be a proper subword of a word $011^n 0y$ (both in \tilde{J}) is if $m = n$ and $011^m 0x$ is a prefix of $011^n 0y$. But this entails that $x = y$, hence $011^m 0x = 011^n 0y$, i.e. no word in \tilde{J} contains another word in \tilde{J} , whence \tilde{J} is anti-factorial. \square

We can then finally show that there are symbolic dynamical systems with decidable languages not in P/poly:

Theorem 12: *For any alphabet Σ , there are anti-factorial, resp. factorial and extensible languages (hence languages of symbolic dynamical systems), over Σ that are decidable and not in P/poly iff $|\Sigma| \geq 2$.* \square

Proof: For $|\Sigma| = 1$, note that all languages over Σ are sparse, hence in P/poly.

For $|\Sigma| \geq 2$, we obtain an anti-factorial language \tilde{J} with the desired properties by Proposition 22; if $|\Sigma| > 2$, we simply choose $\Sigma' \subseteq \Sigma$ with $|\Sigma'| = 2$ and apply Proposition 22 to Σ' .

Proposition 3 shows that if F is decidable, then so is $\mathcal{L}(X_F)$. By the same proposition, we have $F \leq_T^p L(X_F)$, showing that if $\mathcal{L}(X_F)$ were in P/poly then so would F be, by transitivity of \leq_T^p . Setting $F \triangleq \tilde{J}$ and observing that $F \notin \text{P/poly}$, we obtain a decidable, factorial, extensible language $\mathcal{L}(X_F)$ over alphabet $\{0,1\}$ not in P/poly. \square

8 Further Questions

This paper has barely begun to touch upon the myriad of interesting questions regarding the complexity of languages of symbolic dynamical systems. We mention a few particularly interesting problems for the reader's pleasure:

1. Does there exist languages of β -shifts that are complete for classes in the *arithmetical* hierarchy (beyond the decidable languages). How about the *analytical* hierarchy and beyond? We believe the answer is 'yes' (see also [Sim05, Sim08], though completeness is not treated there).
2. In Section 5.3 we used the notion of \leq_T^p -reduction. For the stronger notions of \leq_m^p and $\leq_m^{\log s}$ -reduction, celebrated results by Mahaney [Mah82], Fortune [For79], and Cai and Sivakumar [CS99] state that if there exists a sparse set \leq_m^p -hard for NP (or for coNP), then $P = NP$, and if there exists a sparse hard set for Punder LOGSPACE-reduction, then $LOGSPACE = P$.

Using our current methods, we *cannot* prove that existence of an NP-hard language of some β -shift implies that $P = NP$, but we conjecture the implication to be true.

3. Can the hierarchy results for β -shifts presented in this paper be made — much — tighter? Again, we believe the answer to be 'yes'.
4. Do the hierarchy results presented in the paper generalize to non-deterministic machines? We conjecture 'yes' due to most of our constructions having conversion between languages, rather than the internal makeup of the machine model, as fulcrum.
5. Is there a more uniform way to construct (anti-)factorial languages over a binary alphabet complete for complexity classes. The methods of this paper work for $|\Sigma| \geq 3$, but not for $|\Sigma| = 2$. We conjecture that there is a simple coding of the languages with $|\Sigma| = 3$ as languages over $\{0, 1\}$ that preserves anti-factorialness and only introduces polynomial computational overhead. Such a coding would immediately yield symbolic dynamical systems over $\{0, 1\}$ with hard languages for P, NP, etc.
6. In the vein of the last question above: One *possibly* easy way to obtain NP- and PSPACE-complete languages of symbolic dynamical systems over binary alphabets is to note that the problem: Given a pair $(T, 0^k)$, where T is an encoding of a Turing machine, the problems “Does T halt in at most k steps”, respectively, “Does T halt using at most k tape cells” are NP- and PSPACE-complete, respectively. Observing that the standard encoding of Turing machines usually *do not* contain a certain word (for instance, contain no words on the form 1^i for $i \geq 3$ in [KD00]), it appears very possible to treat such a words as an “extra” character like $\#$ in our construction of $L^\#$.
7. While we have treated languages of symbolic dynamical systems in P/poly and P/poly is the most studied non-uniform complexity class, it could be instructive to find a complete demarcation of which non-uniform classes contain decidable languages of symbolic dynamical systems not in, say, $P/f(n)$ where f is a suitable (computable) bound on circuit size. More ambitiously, one could study a greater range of classes $\mathcal{C}/f(n)$ where \mathcal{C} is any class in the backbone hierarchy, or an even more exotic class.

9 Acknowledgments

The author extends his thanks to an anonymous referee as well as Lasse Nielsen and Anders Starcke Henriksen for valuable comments.

References

- [BCM⁺03] 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.
- [Bla89] F. Blanchard. β -expansions and symbolic dynamics. *Theoretical Computer Science*, 65:131–141, 1989.
- [BMRS00] M.-P. Béal, F. Mignosi, A. Restivo, and M. Sciortino. Forbidden words in symbolic dynamics. *Advances in Applied Mathematics*, 25:163–193, 2000.
- [BP97] 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.
- [CS99] J.-Y. Cai and D. Sivakumar. Sparse hard sets for P: Resolution of a conjecture of Hartmanis. *Journal of Computer and Systems Sciences*, 58(2):280–296, 1999.
- [DdV05] K. Dajani and M. de Vries. Measures of maximal entropy for random β -expansions. *Journal of the European Mathematical Society*, 1:51–68, 2005.
- [DK02] K. Dajani and C. Kraaikamp. *Ergodic Theory of Numbers*, volume 29 of *The Carus Mathematical Monographs*. The Mathematical Association of America, 2002.
- [For79] S. Fortune. A note on sparse complete sets. *SIAM Journal of Computing*, 8(3):431–433, 1979.
- [FS92] C. Frougny and B. Solomyak. Finite beta-expansions. *Ergodic Theory and Dynamical Systems*, 12:713–723, 1992.
- [Für82] M. Fürer. The tight deterministic time hierarchy. In *Proceedings of the 14th ACM Symposium on the Theory of Computing (STOC '82)*, pages 8–16. The ACM Press, 1982.
- [Gef03] V. Geffert. Space hierarchy theorem revised. *Theoretical Computer Science*, 295:171–187, 2003.
- [GJ79] M. Garey and D. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. Freeman, 1979.
- [HS08a] P. Hertling and C. Spandl. Computability theoretic properties of the entropy of gap shifts. *Fundamenta Informaticae*, 83(1–2):141–157, 2008.
- [HS08b] P. Hertling and C. Spandl. Shifts with decidable language and non-computable entropy. *Discrete Mathematics and Theoretical Computer Science*, 10(3):75–94, 2008.
- [Joh99] K.C. Johnson. *Beta-Shift Dynamical Systems and Their Associated Languages*. PhD thesis, University of North Carolina, 1999.
- [Jon97] N.D. Jones. *Computability and Complexity from a Programming Perspective*. The MIT Press, 1997.
- [KD00] 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.
- [KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity. In *Proceedings of the 12th Symposium on Theory of Computing (STOC '80)*, pages 302–309. The ACM Press, 1980. Final version: Turing Machines that take Advice, *L'enseignement Mathématique* 28 (1982), 191–209.
- [LM95] D. Lind and B. Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1995.

- [Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and Systems Sciences*, 25(2):130–143, 1982.
- [Par60] W. Parry. On the β -expansion of real numbers. *Acta Math. Acad. Sci. Hung.*, 11:401–416, 1960.
- [Sav70] W.J. Savitch. Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and Systems Sciences*, 4:177–192, 1970.
- [Sch97] J. Schmelling. Symbolic dynamics for β -shifts and self-normal numbers. *Ergodic Theory and Dynamical Systems*, 17:675–694, 1997.
- [Sim05] J.G. 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.
- [Sim06] J.G. Simonsen. On the computability of the topological entropy of subshifts. *Discrete Mathematics and Theoretical Computer Science*, 8:83–96, 2006.
- [Sim08] J.G. Simonsen. On beta-shifts, computable numbers, decidable languages and the arithmetical hierarchy. Preprint, 2008.
- [Sip06] M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, 2nd edition, 2006.
- [Wal81] P. Walters. *An Introduction to Ergodic Theory*, volume 79 of *Graduate Texts in Mathematics*. Springer-Verlag, 1981.