Minimality and computability of languages of G-shifts

Djamel Eddine AMIR
Université Paris-Saclay, CNRS, LISN, 91400, Orsay, France
amir@lisn.fr https://amirdjameleddine.github.io/

Benjamin HELLOUIN de MENIBUS
Université Paris-Saclay, CNRS, LISN, 91400, Orsay, France
hellouin@lisn.fr https://www.lisn.fr/ hellouin/

Abstract

Motivated by the notion of strong computable type for sets in computable analysis, we define the notion of strong computable type for G-shifts, where G is a finitely generated group with decidable word problem. A G-shift has strong computable type if one can compute its language from the complement of its language. We obtain a characterization of G-shifts with strong computable type in terms of a notion of minimality with respect to properties with a bounded computational complexity. We provide a self-contained direct proof, and also explain how this characterization can be obtained from an existing similar characterization for sets by Amir and Hoyrup, and discuss its connexions with results by Jeandel on closure spaces. We apply this characterization to several classes of shifts that are minimal with respect to specific properties. This provides a unifying approach that not only generalizes many existing results but also has the potential to yield new findings effortlessly. In contrast to the case of sets, we prove that strong computable type for G-shifts is preserved under products. We conclude by discussing some generalizations and future directions.

1 Introduction

Shifts, or shift spaces, are sets of colourings of an infinite regular grid (also called configurations) submitted to local constraints usually given as forbidden patterns. In 1961, Wang first studied colourings of the two-dimensional square grid with finitely many constraints (shifts of finite type or SFT) to study some fragments of predicate calculus [37]. Undecidability phenomena were proved soon after, the first one being the undecidability of the so-called seeded Domino problem [28]: given a partial colouring, can it be extended to a full configuration that satisfies the constraints? The set of such partial colourings is called the *language* of the shift, so this result means that there are SFT with uncomputable language. This result was later extended to other grids, in particular Cayley graphs of finitely generated groups [12], which is the general setting for this article.

Consequently, understanding which properties make the language of a shift computable or not remains a fundamental question. A well-known folklore result states that the language of a minimal SFT, that is, a shift that contains no other non-empty subshift, is computable. However, later research showed that the finite-type assumption is not necessary: it suffices to be able to enumerate all patterns that do not appear in the shift (a notion known as effectiveness) for the result to hold (see [27, 26, 18]). Multiple other results point to the same phenomenon, where being able to enumerate the complement of the language is enough to compute the language itself, under various assumptions that can be seen as a form of minimality: the subshift satisfies a property P that none of its subshifts satisfies.

Similar results in other areas The connection between minimality and decidability also extends to problems in group theory, combinatorics, and computable analysis (see for example [22, 33, 34]). Jeandel attempted to develop a unified theory for groups, shifts and combinatorics [26]; we discuss the relationship with our results in Section 3.4.

In the setting of computable topology and descriptive complexity, a similar phenomenon occurs where some minimality assumption ensures that a description of a set "from the outside" is enough to provide a description "from the inside". Notions of strong computable type and minimality for sets were defined in [5] as follows.

Definition. *A **set** has **strong computable type** if one can compute the set from its semi-computable information (a description of its complement).*

*A **set** is **minimal** for some property if it satisfies this property but no proper subset of it satisfies the property.*

A characterization of sets which have strong computable type related to minimality was given in [5].

Theorem ([5])**.** *A **set** has **strong computable type** if and only if it is minimal satisfying some property with bounded computational complexity (Σ^0_2-computable).*

Our approach In this article, motivated by the above results for sets, we define analogous notions of strong computable type and minimality for G-shifts, where G is a finitely generated group with decidable word problem. To simplify, G-shifts will be called shifts.

Definition. *A **shift** has **strong computable type** if one can compute its language from the complement of its language.*

*A **shift** is **minimal** for some property if it satisfies this property but no proper subshift of it satisfies the property.*

We characterize shifts with strong computable type using minimality, and show that this is implied by the characterization of strong computable type for sets. This motivates the question of classifying shifts according to their computability.

In addition to creating new connections between symbolic dynamics and computable analysis, positive results in this direction will motivate further efforts to find more general theorems that unify the two fields.

Our results We prove the following main theorem.

Theorem (Theorem 3.2)**.** *A **shift** has **strong computable type** if and only if it is minimal for some property with bounded computational complexity (Σ_2^0-computable).*

We apply our theorem to several classes of shifts which are minimal for some specific properties. This provides a unifying approach, as it not only implies many existing results but also introduces new findings effortlessly.

Proposition (Section 3.5)**.** *The following classes of shifts have strong computable type.*

- *Minimal shifts,*
- *Entropy-minimal shifts with a left-computable entropy,*
- *P-isolated shifts,*
- *Infinite-minimal shifts (periodic-minimal shifts and quasi-minimal shifts).*

More details and precise definitions will be provided in the following sections, along with an appendix for clarifications.

Many other natural questions arise, such as: can we obtain other analogous results for shifts to those on computable type for sets ([1, 3, 5, 4, 6, 7, 2, 38, 16])? Furthermore, can we reverse the process by obtaining results for computable type analogous to those on shifts?

Sections are organized as follows: In Section 2, we give a minimal background to understand the main results. In Section 3, we define the notion of strong computable type for shifts, state and prove our main theorem, explain its relation with strong computable type for sets and closure spaces, apply it to classes of shifts and study some properties of strong computable type for shifts. In Section 4, we discuss generalizations of our results and future directions. In Appendix A, we provide more details about topology and descriptive complexity. In Appendix B, we prove the relation between strong computable type for shifts and sets.

2 Preliminaries

To present our results, which are interdisciplinary in nature, as they relate computability and symbolic dynamics to topology and descriptive complexity, we first provide some basic concepts. However, for a more comprehensive understanding, the reader should refer to the relevant references provided throughout the text.

2.1 Basics on groups

We provide some classical basic concepts about groups, particularly the notions of finitely generated groups and the word problem. These results and their applications to shifts can be found in [8].

Definition 2.1. An **alphabet** is a set. A **word** of **length** $n \in \mathbb{N}^*$ on an alphabet A is an element in the finite product $\Pi_{1 \leq i \leq n} A$, the **empty word** is of length 0.

We denote by A^* the set of all finite words on A.

Now, we have the necessary ingredients to define key concepts for groups.

Definition 2.2. Let G be a group and $S \subseteq G$. For $w \in S^*$, we denote the corresponding group element by w_G (the evaluation of w as an element of G, where the product is the group operation of G).

S is a **generating set** for G if every element in G can be written as a word in S^*. G is **finitely generated** if S is finite.

To simplify, we assume that generating sets are symmetric, that is, $g \in S \Rightarrow g^{-1} \in S$.

Definition 2.3. Let G be a group that is finitely generated by S with identity element 1_G.

The **word problem** is the set $\{w \in S^* : w_G = 1_G\}$.

G has **decidable word problem** if this set is decidable, i.e. there is an algorithm which given a word w, decides on finite time whether w_G equals 1_G. This does not depend of the chosen generating set.

2.2 G-shifts

As outlined in the introduction, shifts of dimension 2 can be defined as sets of colorings of planes. Specifically, given an alphabet A (a set of colors), a configuration corresponds to an element of $A^{\mathbb{Z}^2}$, i.e., an assignation of colors $(a_{(i,j)})_{(i,j) \in \mathbb{Z}^2}$ to every coordinate in the plane, where each $a_{(i,j)}$ is a color in A. A shift of dimension 2 is a shift-invariant closed set of configurations in $A^{\mathbb{Z}^2}$.

Similarly, this concept can be extended by indexing the colors with elements of a group G rather than $\mathbb{Z}^2$, i.e. configurations $(a_g)_{g \in G} \in A^G$, which allows us to define G-shifts (see [15], [8]). A configuration $(a_g)_{g \in G}$ can be seen as a function $G \to A$ sending g to a_g.

Now, let us give precise definitions for G-shifts.

Let G be a group and let A be a finite alphabet. The set $A^G = \{x : G \to A\}$ is called the **full shift**, and its elements are called **configurations**. A^G can be endowed with the **pro-discrete** topology, which is metrizable if G is countable and computably metrizable if G is finitely generated. For the latter case, see Appendix A.1 for more details.

Definition 2.4 (G-shifts)**.** A **pattern** is an element $p \in A^F$, where $F \subseteq G$ is finite, and it determines the **cylinder** $[p] = \{x \in A^G : x|_F = p\}$. Let $x : G \to A$ be a configuration, p **appears** on x if there exists some $g \in G$ such that $gx \in [p]$, where $gx : G \to A$ is the function sending $h \in G$ to $x(g^{-1}h)$.

A G**-shift** is a subset $X \subseteq A^G$ which is topologically closed and **shift-invariant** i.e. for every $h \in G$ and every $x \in X$, one has $hx \in X$. We denote by $\mathcal{S}(A^G)$ the set of all non-empty shifts in A^G.

The **language** $\mathcal{L}(X)$ of a G-shift X in A^G (or more generally a set $X \subset A^G$) is the set of patterns $p : F \to A$ that **appear** in X, that is, such that $x|_F = p$ for some $x \in X$. Its complement is denoted $\mathcal{L}^c(X)$.

A G-shift X is **of finite type (SFT)** if it can be defined by a finite set F of forbidden patterns, in the sense that a configuration x is in X if and only if no pattern from F appears in x.

To simplify, we call G-shifts just shifts, the underlying group will be clearly known. In the literature, shifts may be called subshifts since every shift is a subshift of the full shift.

If X is a shift, then a pattern p appears in some $x \in X$ if and only if $X \cap [p] \neq \emptyset$ (this will be used in the proofs).

Note that the set of cylinders $[p]$ is a clopen sub-basis of the pro-discrete topology on A^G.

2.3 Effectiveness of G-shifts

Let G be a finitely generated group with decidable word problem, $S \subseteq G$ be a finite generating set and A be a finite alphabet. To define notions of effectiveness on G-shifts (see [23]), let us define some classical notions in computability theory.

Definition 2.5. A subset $I \subseteq \mathbb{N}$ is **computably enumerable (c.e.)** if there exists an algorithm that runs forever and enumerates all elements of I; equivalently, there is an algorithm that **semi-decides** I, that is, given $i \in \mathbb{N}$, the algorithm will enumerate i after a finite time if $i \in I$ and keeps running forever otherwise.

A set is **co-computably enumerable (co-c.e.)** if its complement is computably enumerable.

A set is **computable (decidable)** if it is both c.e. and co-c.e..

Let $p \in A^F$ be a pattern, that is, a function $F \to A$, for some finite set $F \subseteq G$. Let $f : S^* \to G$ be the function sending a word w to the corresponding group element w_G. When manipulated by an algorithm, the pattern p is represented by a function $p' : W \to A$, with $W \subset S^*$ finite, such that $p \circ f|_W = p'$ and $W \cap f^{-1}(i) \neq \emptyset$ for all $i \in F$. The set $\mathcal{C}$ of all such functions is countable as they are functions from a finite set of words to a finite alphabet:

$$\mathcal{C} = \{p \circ f_F : p \in A^F, F \subseteq G \text{ is finite}\}.$$

Note that $\mathcal{C}$ can be effectively enumerated (using an algorithm), given S and the fact that G has decidable word problem.

Hence the notions of c.e., co-c.e. and computable sets can be extended from subsets of $\mathbb{N}$ to subsets of $\mathcal{C}$.

Definition 2.6 (Effectively closed shifts)**.** A shift $X \subseteq A^G$ is **effectively closed** if it has a co-computably enumerable language.

Such shifts are sometimes simply called *effective.*

Fact 2.7 (Proposition 2.1 in [8])**.** *Equivalently, a shift is effectively closed if it can be defined by a computably enumerable set of forbidden patterns.*

In particular, an SFT is effectively closed because a finite set of forbidden patterns is computably enumerable.

This terminology comes from the fact that a shift is effectively closed if and only if it is topologically effectively closed (see [9] and Appendix A.2.1). Since a shift is always topologically closed, an effectively closed shift corresponds to an effectively topologically closed shift.

2.4 Descriptive complexity

In this section, we define the notion of properties of shifts, their descriptive complexity and minimal elements satisfying them.

Let G be a finitely generated group with decidable word problem and let A be a finite alphabet. The set $\mathcal{S}(A^G)$ of all non-empty shifts in A^G can be endowed with a computable metric structure, see Appendix A.2.

When we say that an algorithm is given an enumeration as input, this formally means that the enumeration is given as an oracle to the algorithm (that is, the algorithm has access to it as a special input of infinite length).

A **property** is a subset of $\mathcal{S}(A^G)$.

Definition 2.8 (See Section 2 in [17])**.** The **descriptive complexity** of a property P can be defined as follows: a property is Π_1^0 if it is effectively closed. Equivalently (see Fact A.4), there exists an algorithm such that for every $X \in \mathcal{S}(A^G)$, given two enumerations of its language $\mathcal{L}(X)$ and of its complement $\mathcal{L}^c(X)$, the algorithm halts if and only if $X \notin P$ (it semi-decides whether $X \notin P$).

A property is Σ_1^0 if its complement is Σ_1^0 (the algorithm semi-decides whether $X \in P$). It is Σ_2^0 if it is a uniform union of Π_1^0 properties, that is, $P = \bigcup_{i \in \mathbb{N}} P_i$ and there is an algorithm that, given i, semi-decides if a shift X is not in P_i.

Definition 2.9. Let $P \subseteq \mathcal{S}(A^G)$ be a property and $X \in \mathcal{S}(A^G)$ be a non-empty shift. We say that X is P**-minimal** if $X \in P$ and, for every non-empty subshift $Y \subsetneq X$, $Y \notin P$.

A subshift Y of a shift X can always be obtained by forbidding an additional set of patterns; that is, $Y = X \setminus \bigcup_{p \in P} \bigcup_{g \in G} g[p]$ for some set $P \subseteq \mathcal{L}(X)$, where for $p : F \to A$ we define $g[p] = \{gx \in A^G : x \in A^G \text{ and } x|_F = p\}$.

3 Strong computable type for G-shifts

3.1 Definition

We introduce the notion of strong computable type for G-shifts, which originates from a similar concept in computable analysis (see Section 3.3).

As mentioned in the introduction, an effectively closed shift has computable language if it satisfies additional properties, such as being minimal. Identifying such sufficient conditions is of particular interest. We generalize this question as follows: given information on the co-language of a shift (semi-computable or "negative" information), when can we use it to compute its language? This corresponds to asking under which conditions there is an algorithm that enumerates $\mathcal{L}(X)$ when provided with any oracle O that makes $\mathcal{L}(X)^c$ computably enumerable.

This definition is formalised as follows, where an **oracle** is viewed as a subset of $\mathbb{N}$.

Definition 3.1 (Strong computable type for G-shifts)**.** Let G be a finitely generated group with decidable word problem, A be a finite alphabet and $X \subseteq A^G$ be a shift.

X has **strong computable type** if for every oracle $O \subset \mathbb{N}$, the fact that X is effectively closed relative to O implies that the language of X is computable relative to O.

This is equivalent, by Selman's theorem [36], to the fact that $\mathcal{L}(X)$ is enumeration-reducible to $\mathcal{L}^c(X)$.

Note that for effectively closed shifts (in particular SFTs) having computable language is equivalent to having strong computable type.

There exist shifts that have strong computable type but do not have computable language. Indeed, a shift X which has a c.e. language which is not co-c.e. clearly has strong computable type; consider for example the shift on $\{0,1\}^{\mathbb{Z}}$ obtained by forbidding all patterns 01^k0 such that the k-th Turing machine doesn't halt.

This notion is interesting because it provides a characterization that unifies various arguments, offers new results, and connects the computability of languages to the descriptive complexity of topological properties (see Section 3.2).

3.2 Shifts and minimality

In this section, we prove our main theorem, which characterizes shifts with strong computable type by the fact that they must satisfy a property that makes them minimal. This is motivated by Theorem 4.7. in [5].

Theorem 3.2 (Main theorem). *Let G be a finitely generated group with decidable word problem and let A be a finite alphabet. Let $X \subseteq A^G$ be a non-empty shift. The following are equivalent.*

1. *X has strong computable type,*
2. *There exists a Σ_2^0 property P in $\mathcal{S}(A^G)$ such that X is P-minimal.*

Remark 3.3. From the proof of the implication 2. $\Rightarrow$ 1. below, we can see that Theorem 3.2 holds when the property P is only Σ_2^0 relative to $\mathcal{L}^c(X)$ (that is, the algorithm used to prove that P is Σ_2^0 receives $\mathcal{L}^c(X)$ as an additional oracle, where X is the fixed subshift in the theorem statement).

Proof. By Selman's theorem, Condition 1. is equivalent to the fact that an effective procedure produces an enumeration of $\mathcal{L}(X)$ from any enumeration of its complement $\mathcal{L}^c(X)$.

1. $\Rightarrow$ 2. Assume that there is an effective procedure (a machine M) producing an enumeration of $\mathcal{L}(X)$ from any enumeration of $\mathcal{L}^c(X)$, let us define some Π_1^0 (and hence Σ_2^0) property P, for which X is P-minimal.

Let Γ be the set of all pairs (σ, p) where $\sigma = \sigma_0 \dots \sigma_k$ is a finite sequence of patterns such that the machine M outputs the pattern p after reading σ; clearly Γ is c.e..

Let $\mathcal{U} = \bigcup_{(\sigma,p)\in\Gamma} \mathcal{U}_{(\sigma,p)}$ with

$$\mathcal{U}_{(\sigma,p)} = \{X' \in \mathcal{S}(A^G) : \forall i \leq k, X' \cap [\sigma_i] = \emptyset \text{ and } X' \cap [p] = \emptyset\}.$$

Note that a c.e. union of Σ_1^0 properties is a Σ_1^0 property. $\mathcal{U}$ is a Σ_1^0 property because Γ is c.e. and for every $(\sigma, p) \in \Gamma$, $\mathcal{U}_{(\sigma,p)}$ is a Σ_1^0 property (given an enumeration of $\mathcal{L}^c(X')$ one can semi-decide whether the σ_i's and p are in $\mathcal{L}^c(X')$, equivalently whether the corresponding cylinders do not intersect X').

Let P be the complement of $\mathcal{U}$, it is hence a Π_1^0 property. Let us prove that X is P-minimal.

X is in P because otherwise the machine M fails on X, i.e. after reading a finite sequence of patterns σ that are in $\mathcal{L}^c(X)$ the machine outputs a pattern p that is in $\mathcal{L}^c(X)$ (which contradicts our assumption).

Now, let $Y \subsetneq X$ be a shift and $p \in \mathcal{L}(X) \setminus \mathcal{L}(Y)$. We have $Y \notin P$ because the machine eventually outputs p after reading some finite sequence σ, which implies that $Y \in \mathcal{U}_{(\sigma,p)}$.

2. $\Rightarrow$ 1. Assume that P is Σ_2^0 and that X is P-minimal. Hence, $P = \bigcup_{i\in I} P_i$ with $P_i \in \Pi_1^0$ for all i. It is easy to see that there exists some i_0 such that X is P_{i_0}-minimal.

By Fact A.5, we can suppose that P_{i_0} is Π_1^0 in a stronger sense: there exists an algorithm such that for every $Y \in \mathcal{S}(A^G)$, given an enumeration of the complement of its language $\mathcal{L}^c(Y)$, the algorithm halts if and only if $Y \notin P$; see Appendix A.2.2 for more details.

Given an enumeration of $\{q : [q] \cap X = \emptyset\}$ (that is, an enumeration of $\mathcal{L}^c(X)$), we need to enumerate $\mathcal{L}(X)$. Note that $p \in \mathcal{L}(X)$ iff $X' = X \setminus \bigcup_{g \in G} g[p]$ is a proper subshift in X; as X is P_{i_0}-minimal, it is equivalent to $X' \notin P_{i_0}$.

X' can be defined by the set of forbidden patterns $\mathcal{L}^c(X) \cup \{p\}$. There is an algorithm that, for any shift Y, computes an enumeration of $\mathcal{L}^c(Y)$ from any enumeration of a set of forbidden patterns that defines Y; this can be proved exactly as Fact 2.7 relative to an oracle. In particular, from an enumeration of $\mathcal{L}^c(X) \cup \{p\}$ we can compute, uniformly in p, an enumeration of $\mathcal{L}^c(X')$.

Since P_{i_0} is Π^0_1, from an enumeration of $\mathcal{L}^c(X')$ we can semi-decide the condition $X' \notin P_{i_0}$. Therefore, whether $p \in \mathcal{L}(X)$ is semi-decidable relative to an enumeration of $\mathcal{L}^c(X)$. In other words, from an enumeration of $\mathcal{L}^c(X)$ we can compute an enumeration of $\mathcal{L}(X)$. $\square$

If we remove oracles from Theorem 3.2, we obtain the following characterization of the decidability of languages of effectively closed shifts.

Corollary 3.4. *Let G be a finitely generated group with decidable word problem and let A be a finite alphabet. Let $X \subseteq A^G$ be a non-empty effectively closed shift. The following are equivalent.*

1. *X has computable language,*
2. *There exists a Σ^0_2 property P in $\mathcal{S}(A^G)$ such that X is P-minimal.*

3.3 Relation with strong computable type for sets

The notion of strong computable type was initially defined for sets (see [5]), using the notion of copies in the **Hilbert cube** $Q = [0,1]^{\mathbb{N}}$, and generalized the notion of computable type for sets defined in [25]. For a comprehensive understanding of strong computable type for sets, see Amir's PhD thesis [1].

A **copy** of a set $X \subseteq Q$ is the image of X by a homeomorphism $Q \to Q$ (a continuous bijection with a continuous inverse).

Q is a computable metric space, and hence there exists a computable dense sequence $(q_i)_{i \in \mathbb{N}}$ in Q. Let $(B_i)_{i \in \mathbb{N}}$ be an enumeration of all the balls $B(q_i, r_j)$ with center q_i and radius $r_j \in \mathbb{Q}$.

A compact set X in Q is **semi-computable** if the set $\{i \in \mathbb{N} : X \cap B_i = \emptyset\}$ is c.e.. It is **computable** if it is semi-computable and in addition the set $\{i \in \mathbb{N} : X \cap B_i \neq \emptyset\}$ is c.e.. The same notions can be defined by replacing Q by any space with a computable metric structure.

Now, let us define **strong computable type for sets**. A compact set X in Q has strong computable type if for every copy Y of it in Q, and every oracle which makes Y semi-computable, Y is also computable using the oracle.

One might attempt to define strong computable type for G-shifts using copies. However, this notion is very restrictive: for example, since any shift without isolated points is homeomorphic to the Cantor set, any such shift is also homeomorphic to an effectively closed shift whose language is not computable, so it does not have strong computable type.

Another possible definition involves using analogs of homeomorphisms in symbolic dynamics, namely conjugacies. However, since conjugate shifts preserve the decidability of languages, proving the decidability of the language of one shift automatically implies the decidability of the languages of all conjugate shifts. Therefore, we define strong computable type for G-shifts without considering either copies or conjugacies.

A similar result for the computability of sets, relating computable type with minimality was proved (see [5, 1]). We prove in Appendix B that Theorem 3.2 can be obtained by this characterization of strong computable type for sets, though it is not immediate.

3.4 Relation with maximal elements in quasivarieties

In Theorem 5 of [26], Jeandel proves a similar result for quasivarieties, which is a general framework including subshifts as well as finitely generated groups, first-order theories, etc. We compare and contrast it with our results.

Jeandel's result is a sort of dual of our Theorem 3.2 as it applies to maximal elements and yields an enumeration reduction that is the opposite direction from strong computable type; this is only a consequence of the fact that it applies to $\mathcal{L}^c(X)$ instead of X. Quasivarieties correspond to (in our context) effectively closed properties that are stable under finite unions. Our result does not requires this last assumption, although note that:

- in the proof of the implication 1. $\Rightarrow$ 2. in Theorem 3.2, the property P is stable by union, and therefore the theorem would hold with this additional assumption;
- properties used in our examples in Section 3.5 are stable by union, so we believe these results could be proved in Jeandel's framework (indeed, examples from Sections 3.5.5 and 3.5.6 appeared in [26]).

Additionally, Theorem 5 of [26] is not an equivalence; note that the converse given in Theorem 6 in *op.cit.* is not comparable to our main theorem, and shows in the case of subshifts that the language of a subshift with strong computable type is enumeration-reducible to the language of a minimal subshift.

We are not sure of whether Π_1^0 properties and quasivarieties are equivalent in terms of expressing minimality properties (see Question 3). We believe that, at least, Π_1^0 properties provide an easier description of shifts with strong computable type for researchers in the symbolic dynamics community.

3.5 Applications

In this section, we provide examples of G-shifts that have strong computable type based on our characterization. Most of these results are generalizations of existing results to arbitrary oracles, and a main interest of our characterization is to unify the underlying arguments.

Let G be a finitely generated group with decidable word problem, and let A be a finite alphabet.

3.5.1 Minimal shifts

Recall from the introduction that a non-empty shift is **minimal** if it contains no other subshifts except itself and the empty shift.

The following result has been first stated, to the best of our knowledge, in [18]; it is a generalisation of a folklore result for the finite type case.

Theorem 3.5 ([18])**.** *An effectively closed minimal shift has computable language.*

We prove a generalization of this theorem (relative to any oracles) based on our characterization.

Proposition 3.6. *A minimal shift has strong computable type.*

Proof. A minimal shift is P-minimal for the property $P = \mathcal{S}(A^G)$ (remember that the empty shift is not a member of $\mathcal{S}(A^G)$ by definition). This property is trivially Π^0_1, using an algorithm that never halts. □

3.5.2 Entropy-minimal shifts

In this section we suppose in addition that the group G is amenable (we do not give the definition of being amenable as we don't explicitly use it).

Definition 3.7. Let h be the entropy map $\mathcal{S}(A^G) \to \mathbb{R}^+$ that to a shift associates its topological entropy (see Chapter 9 in [29] for more details of this notion). A G-shift X is called **entropy-minimal** when every proper subshift $Y \subset X$ satisfies $h(Y) < h(X)$.

The following result is stated in a recent unpublished work by Carrasco-Vargas, Herrera Nunez and Sablik that appeared in Chapter 6 in Carrasco-Vargas's PhD Thesis [14].

Theorem 3.8 ([14])**.** *An entropy-minimal SFT whose entropy is a computable real number has computable language.*

A real number is **left-computable** if there exists a computable increasing sequence of rational numbers converging to it.

In Proposition 3.9, we generalize this result in several ways: from SFTs to effectively closed shifts, from computable entropies to left-computable entropies, and finally from having computable languages to having strong computable type, based on our characterization in Theorem 3.2.

Proposition 3.9. *An entropy-minimal shift X whose entropy is a real number that is left-computable relative to $\mathcal{L}^c(X)$ has strong computable type.*

Proof. Let X be an entropy-minimal shift with $q = h(X)$. Take the property $P = h^{-1}([q, +\infty))$. Relative to $\mathcal{L}^c(X)$, P is Π_1^0 because q is left-computable and h is upper semi-computable (combine Proposition 6.17 in [14] with Fact A.3). Clearly, X is P-minimal, so by Theorem 3.2 it has strong computable type. $\square$

Remark 3.10. The previous proof also works when $h(X)$ is not left-computable, under the assumption that $\sup_{Y \subsetneq X} h(Y) < h(X)$. In this case, pick q to be any left-computable number in the interval $[\sup_{Y \subsetneq X} h(Y), h(X)]$.

Entropy-minimality alone is not sufficient We give an example of a shift in $\{0,1\}^{\mathbb{Z}}$ that is entropy-minimal and does not have strong computable type.

Definition 3.11. Let $\alpha \leq \beta$ be two real numbers. Define the shift $X_{[\alpha,\beta]} \subset \{0,1\}^{\mathbb{Z}}$ by forbidding the set of finite patterns

$$\{w \in \{0,1\}^n : \#_1 w < \lfloor \alpha n \rfloor \text{ or } \#_1 w > \lceil \beta n \rceil\},$$

where $\#_1 w$ is the number of occurrences of the symbol 1 in w. $X_\alpha = X_{[\alpha,\alpha]}$ is known as the Sturmian shift of slope α.

See [32, Chapter 2] for more properties of Sturmian configurations and Sturmian shifts. In particular, the language of a Sturmian shift with irrational slope contains exactly $n+1$ different patterns of length n.

Proposition 3.12. *The shift $X_{[0,\alpha]}$ is entropy-minimal and, for some right-computable value of α, does not have strong computable type.*

Proof. The shift $X_{[0,\alpha]}$ is *strongly irreducible*, that is, for two patterns $u, v \in \mathcal{L}(X_{[0,\alpha]})$, we have $u0^k v \in \mathcal{L}(X_{[0,\alpha]})$ for some constant $k = \lceil \frac{2}{\alpha} \rceil$. Indeed, check that $\#_1 u0v = \#_1 u + \#_1 v \leq \lceil \alpha |u| \rceil + \lceil \alpha |v| \rceil \leq \alpha(|uv| + 2) \leq \lceil \alpha(|uv| + k) \rceil$, and the same argument applies to any subpattern of $u0^k v$. A strongly irreducible shift is entropy-minimal: see Corollary 4.7 in [20].

If α is a right-computable real number (namely, $-\alpha$ is left-computable), then $X_{[0,\alpha]}$ is an effectively closed shift since the set $\{w \in \{0,1\}^n : \#_1 w > \lceil \alpha n \rceil\}$ is c.e.. In particular, $\mathcal{L}(X_{[0,\alpha]})$ is co-computably enumerable.

By an argument that can be found e.g. in the proof of Theorem 3.6 in [21], for a strongly irreducible shift Y, from the number of patterns of size n in $\mathcal{L}(Y)$ we can compute an approximation of $h(Y)$ with a known error rate. In our case, having a right-approximation of number of patterns of size n in $\mathcal{L}(X_{[0,\alpha]})$, we obtain that $h(X_{[0,\alpha]})$ is right-computable.

Now assume that $X_{[0,\alpha]}$ has computable language. Compute all patterns of length n in $\mathcal{L}(X_{[0,\alpha]})$ and count the maximum number of symbols 1 that appear in a pattern: this number is $\lfloor \alpha n \rfloor$, from which we get an approximation of α up to error $1/n$. Therefore, α is a computable real number in this case.

It follows that $X_{[0,\alpha]}$, for α right-computable but not computable, is entropy-minimal, effectively closed and its language is not computable, so it does not have strong computable type. $\square$

3.5.3 P-isolated shifts

As in the previous section, studying isolated points is motivated by the unpublished work of Carrasco-Vargas, Herrera Nunez and Sablik that appeared in [14].

In their work they find an alternative proof that a minimal SFT has computable language, using the fact that it is isolated in $\mathcal{S}(A^G)$.

Similarly, they prove that an entropy-minimal SFT with computable real entropy q has computable language, using the fact that it is isolated in $h^{-1}([q, +\infty))$, see Definition 3.7.

The common property between $\mathcal{S}(A^G)$ and $h^{-1}([q, +\infty))$ is that they are Π_1^0 (see Fact A.3 in the Appendix for $\mathcal{S}(A^G)$). In general, isolated points in Π_1^0 classes are computable: see Fact 2.14 in [19]. It motivates us to generalize the result as follows.

Proposition 3.13. *Let $X \in \mathcal{S}(A^G)$ be a non-empty shift and P be a property in $\mathcal{S}(A^G)$ which is Σ_2^0 relative to $\mathcal{L}^c(X)$. If X is isolated in P, then X has strong computable type.*

Proof. Since X is isolated in P, we can find a cylinder $[p]$ such that $X \cap [p] \neq \emptyset$ and $Y \in P \Rightarrow Y \cap [p] = \emptyset$. The property $C = \{Y : Y \cap [p] \neq \emptyset\}$ is both Σ_1^0 and Π_1^0. Let $P' = P \cap C = \{X\}$. P' is Σ_2^0 relative to $\mathcal{L}^c(X)$ because it is the intersection of a property that is Σ_2^0 relative to $\mathcal{L}^c(X)$ with a Σ_1^0 property. Clearly, X is P'-minimal, hence it has strong computable type. □

3.5.4 Periodic-minimal shifts

Definition 3.14. The *orbit* $\mathrm{Orb}(x)$ of a configuration $x \in \mathcal{A}^G$ is the closure of the set $\{g \cdot x : g \in G\}$. A (strongly) periodic configuration is a configuration x such that the number of elements $\#\,\mathrm{Orb}(x)$ is finite.

To a shift X we associate the vector $\mathrm{Per}(X) = (\mathrm{Per}_i(X))_{i \in \mathbb{N}}$, where $\mathrm{Per}_i(X)$ is the number of periodic points in X with orbit size i or less. This is a classical conjugacy invariant for shifts, usually presented under the form of a Zeta function; see e.g. [31] for more information on this object.

A shift X is **period-minimal** if, for any subshift $Y \subsetneq X$, $\mathrm{Per}_i(Y) < \mathrm{Per}_i(X)$ for some $i \in \mathbb{N}$. This corresponds to minimality for the Π_1^0 property $P_X = \{Y : \forall i, \mathrm{Per}_i(Y) \geq \mathrm{Per}_i(X)\}$; therefore a period-minimal shift has strong computable type.

Proposition 3.15. *A shift is period-minimal if, and only if, it has dense periodic points. These shifts have strong computable type.*

Proof. If X is period-minimal, then the subshift obtained from X by forbidding any pattern $p \in \mathcal{L}(X)$ has strictly less periodic points. This means that any pattern $p \in \mathcal{L}(X)$ appears in some periodic point of X, that is, $[p] \cap X$ contains a periodic point. Since cylinders are a basis of the topology, periodic points are dense in X. Conversely, if periodic points are dense in X, then any strict subshift $Y \subsetneq X$ has strictly less periodic points. □

We obtained hence a stronger version of a classical result:

Theorem 3.16 ([30])**.** *An effectively closed shift with dense periodic points has computable language.*

3.5.5 Quasi-minimal shifts

Quasi-minimal shifts as introduced by Salo [35] are shifts with finitely many distinct subshifts.Let

Theorem 3.17 ([35], Theorem 9.)**.** *An effectively closed quasi-minimal shift has computable language.*

We strenghten this result by proving the following.

Proposition 3.18. *Quasi-minimal shifts have strong computable type.*

Proof. For a given quasi-minimal shift X, denote $X_1 \dots X_n$ its distinct subshifts and fix $p_i \in \mathcal{L}(X)\backslash\mathcal{L}(X_i)$ for all $1 \leq i \leq n$. X is then minimal for the Π_1^0 property $P_X = \{Y : \forall i \leq n, p_i \in \mathcal{L}(Y)\}$. □

In *op.cit*, Salo studies a different notion, initially introduced in [18], which is having finitely many distinct *minimal* subshifts. We do not believe that such shifts must have strong computable type.

3.5.6 Infinite-minimal shifts

It is well-known [10, Theorem 3.8] that a shift is finite if and only if it contains only strongly periodic points. We consider the property for a shift X to be **infinite-minimal**, that is, $|X| = \infty$ and all subshifts $Y \subsetneq X$ are finite. Equivalently, such a shift is minimal for the property of containing a non-periodic configuration. This property was called *just-infinite* in [26]. The property of being infinite is known to be Π_0^1 in $\mathbb{Z}^2$ but not in $\mathbb{Z}^d$, for $d > 2$ (see [13]), so we do a direct proof.

Proposition 3.19. *An infinite-minimal shift has strong computable type.*

Proof. If there are finitely many such Y's, then X is quasi-minimal and we conclude by Theorem 3.17. Otherwise, X contains infinitely many periodic points. If there is a pattern p that appears in no periodic point, the subshift obtained from X by forbidding p contains all periodic points of X, so it is infinite, which contradicts infinite-minimality. We conclude that periodic points are dense in X, which is enough by Theorem 3.16. □

3.5.7 Full extensions

Jeandel and Vanier proved the following characterization.

Theorem 3.20 ([27]). *A shift X in $A^{\mathbb{Z}^d}$ has computable language if and only if it is the* projective subdynamics *of a minimal effectively closed shift Y (possibly with larger alphabet and dimension); in other words, $\mathcal{L}(X) = \mathcal{L}(Y) \cap \mathcal{L}(A^{\mathbb{Z}^d})$, which are patterns $p : F \to A \in \mathcal{L}(Y)$ such that $F \subset \mathbb{Z}^d$.*

We show that if X is the projective subdynamics of a minimal shift Y then it has strong computable type, which corresponds to one direction of the previous result relative to an oracle. It is possible that the proof of [27] can also be generalised for the other direction.

Assume that $Y \subset B^{\mathbb{Z}^{d'}}$ with $A \subset B$ and $d \leq d'$. We remark that in the previous theorem, we have a Π^0_1 property P for which X is minimal:

$$P = \{Z \in \mathcal{S}(A^{\mathbb{Z}^d}) : Y \setminus \bigcup_{g \in \mathbb{Z}^{d'}} \bigcup_{w \in \mathcal{L}^c(Z)} g[w] \neq \emptyset\}.$$

Notice that since Z is a subshift of $A^{\mathbb{Z}^d}$, $\mathcal{L}^c(Z)$ contains patterns $F \to A$ with $F \subset \mathbb{Z}^d$, which are seen as patterns $F \to B$ with $F \subset \mathbb{Z}^{d'}$.

Let us prove that X is P-minimal. $Y \setminus \bigcup_{g \in \mathbb{Z}^{d'}} \bigcup_{w \in \mathcal{L}^c(X)} g[w] = Y \neq \emptyset$ so $X \in P$. Now let X' be a proper subshift in X: this means that there exists some pattern $p \in \mathcal{L}(X) \setminus \mathcal{L}(X')$. By assumption, $p \in \mathcal{L}(Y)$. Since Y is minimal, $Y \setminus \bigcup_{g \in \mathbb{Z}^{d'}} \bigcup_{w \in \mathcal{L}^c(X')} g[w] = \emptyset$ hence $X' \notin P$.

As Y is effectively closed and minimal, it has computable language (see Section 3.5.1), which makes the condition $Y \setminus \bigcup_{g \in \mathbb{Z}^{d'}} \bigcup_{w \in \mathcal{L}^c(Z)} g[w] = \emptyset$ semi-decidable given an enumeration of $\mathcal{L}^c(Z)$. Hence P is a Π^0_1 property.

3.6 Invariance under transformations

Strong computable type is preserved under products and factors. However, it is not preserved under unions and intersections.

3.6.1 Factors and computable morphisms

Let A and B be two finite alphabets. A factor $F : \mathcal{S}(A^G) \to \mathcal{S}(B^G)$ is a surjective morphism between G-shifts that commutes with the shift action. It sends shifts with computable languages in $\mathcal{S}(A^G)$ to shifts with computable languages in $\mathcal{S}(B^G)$ because it is a computable surjective function. Using oracles, it sends shifts with strong computable type to shifts with strong computable type. More generally, an image of a shift with strong computable type by a computable morphism has strong computable type.

3.6.2 Products

In contrast to the case of sets, where products do not always preserve strong computable type (see [6]), strong computable type for shifts is preserved under finite products.

Let A and B be two finite alphabets. Let X be a shift in $\mathcal{S}(A^G)$ and Y be a shift in $\mathcal{S}(B^G)$, the product $X \times Y$ is a shift in $\mathcal{S}(A^G) \times \mathcal{S}(B^G) \subseteq \mathcal{S}(A^G \times B^G) \equiv \mathcal{S}((A \times B)^G)$.

Proposition 3.21. *If two shifts X and Y have strong computable type, then $X \times Y$ has strong computable type.*

Proof. Assume that $X \subset A^G$ and $Y \subset B^G$. A pattern $w : F \to A \times B \in \mathcal{L}^c(X \times Y)$ can be seen as the product of two patterns $u : F \to A$ and $v : F \to B$ such that either $u \in \mathcal{L}^c(X)$ or $v \in \mathcal{L}^c(Y)$. For a pattern $p : F \to A$, $p \in \mathcal{L}^c(X)$ is equivalent to the fact that all the pairs (p, q) with $q : F \to B$ are in $\mathcal{L}^c(X \times Y)$. Indeed, if $p \in \mathcal{L}(X)$, then only pairs such that $q \in \mathcal{L}^c(Y)$ will be enumerated and $\mathcal{L}(Y) \neq \emptyset$. Therefore, given an enumeration of $\mathcal{L}^c(X \times Y)$ one can enumerate the elements of $\mathcal{L}^c(X)$.

By symmetry, one can enumerate the elements of $\mathcal{L}^c(Y)$. As X and Y have strong computable type, $\mathcal{L}(X)$ and $\mathcal{L}(Y)$ are computable. Hence $\mathcal{L}(X \times Y) = \mathcal{L}(X) \times \mathcal{L}(Y)$ is computable and $X \times Y$ has strong computable type. □

3.6.3 Unions and intersections

Proposition 3.22. *There exists two shifts X, Y that have strong computable type such that $X \cup Y$ does not have strong computable type. Similarly, there exists two shifts X' and Y' that have strong computable type such that $X' \cap Y'$ does not have strong computable type.*

Proof. Consider $X_{[\alpha,\alpha+1/2]}$ and $X_{[\beta-1/2,\beta]}$ for some real numbers $\alpha < \beta - 1/2 < \alpha + 1/2 < \beta$ (see Definition 3.11). It is clear that $X_{[\alpha,\alpha+1/2]} \cup X_{[\beta-1/2,\beta]} = X_{[\alpha,\beta]}$. If α is left-computable and β is right-computable (namely, $-\beta$ is left-computable), but one of them is not computable, then $X_{[\alpha,\beta]}$ is effectively closed but its language is not computable by the same argument as in Proposition 3.12, so it does not have strong computable type. However, from an enumeration of $\mathcal{L}^c(X_{[\alpha,\alpha+1/2]})$ and by counting the number of symbols 1 in each, it is not hard to compute an upper approximation of α and a lower approximation of $\alpha - 1/2$, and therefore to compute arbitrarily good approximations of α with known error, from which it is straightforward to compute the language of $X_{[\alpha,\alpha+1/2]}$. It follows that $X_{[\alpha,\alpha+1/2]}$ has strong computable type. The case of $X_{[\beta-1/2,\beta]}$ is symmetric.

For the intersection, apply a similar argument to the same shifts when $\beta - 1/2 < \alpha < \beta < \alpha + 1/2$. □

The next proposition shows that strong computable type is conserved under disjoint unions. It is not enough that the intersection $X \cap Y$ is simple (in a computational sense) since, in the previous proof, we had $X_{[\alpha,\alpha+1/2]} \cap X_{[\beta-1/2,\beta]} = X_{[\beta-1/2,\alpha+1/2]}$ which has computable language.

Proposition 3.23. *Let X and Y be two shifts with strong computable type such that $X \cap Y = \emptyset$. Then $X \cup Y$ have strong computable type.*

Proof. In this proof we assume that the underlying group G is generated by the set of generators S and we denote $B_S(n) = \{w_G : w \in S^{\leq n}\}$.

Since $X \cap Y = \emptyset$, a compactness argument shows that there is a $N \in \mathbb{N}$ such that, if $p \in A^{B_S(n)}$ for $n > N$, $[p] \cap X = \emptyset$ or $[p] \cap Y = \emptyset$. Indeed, if that was not the case, we would get a sequence of patterns with increasing support from which we could extract by compacity an element of $X \cap Y$.

Denote by E_Y the set of patterns defined as follows. For any pattern p of support F,

- if $F \subset B_S(N)$, $p \in E_Y$ if and only if $[p] \cap X = \emptyset$;
- otherwise, let n be the smallest number such that $F \subset B_S(n)$. Then $p \in E_Y$ if and only if, for any pattern p' of support $B_S(N)$ that extends p (that is, $p'|_{F \cap B_S(N)} = p|_{B_S(N)}$), $[p'|_{B_S(N)}] \cap Y \neq \emptyset$.

Checking whether $p \in E_Y$ requires finitely many checks of the form $[q] \cap X = \emptyset$ or $[q] \cap Y \neq \emptyset$ for a pattern q of support included in $B_S(N)$. There are finitely many such patterns, so E_Y is a computable set.

Now assume that we receive as input an enumeration of $\mathcal{L}^c(X \cup Y) = \mathcal{L}^c(X) \cap \mathcal{L}^c(Y)$. We show that $\mathcal{L}^c(X \cup Y) \cup E_Y = \mathcal{L}^c(X)$. It is clear that $E_Y \subset \mathcal{L}^c(X)$ by definition of N and E_Y. Furthermore, $\mathcal{L}(Y) \cap \mathcal{L}^c(X) \subset E_Y$; in fact the first point includes all "small" patterns in $\mathcal{L}^c(X)$ and the second point includes all "large" patterns in $\mathcal{L}(Y)$.

We compute an enumeration of $\mathcal{L}^c(X \cup Y) \cup E_Y = \mathcal{L}^c(X)$. Since X has strong computable type, we can compute a enumeration of $\mathcal{L}(X)$. Doing similarly for Y, we compute an enumeration of $\mathcal{L}(Y)$. From this we obtain an enumeration of $\mathcal{L}(X \cup Y) = \mathcal{L}(X) \cup \mathcal{L}(Y)$, so $X \cup Y$ has strong computable type. □

The previous proof relies on the fact that, in A^G, two disjoint shifts (or more generally two disjoint closed sets) X and Y are computably separable; E_Y plays the role of a computable superset of Y that is disjoint from X.

4 Discussion, generalizations and future directions

Several results in this article can be generalized to recursively presented groups.

The notion of strong computable type for shifts can be extended to pairs of shifts, consisting of a shift and a subshift of it, in the same way as the notion of strong computable type for pairs of sets. Similar results can then be proved, and we will address this in a subsequent article that extends this one.

Our results for strong computable type for shifts can be analogously extended to strong computable type for sets. For example, P-isolated sets, where P is a topological Σ_2^0 invariant, have strong computable type.

Question 1. *Let us consider further results on the computable type of sets. Can we obtain analogous results for shifts, and vice versa?*

The results here also motivate an independent research direction: the study and characterization of the descriptive complexity of properties of shifts, similar to how such properties have been studied for sets in [5, 7].

Question 2. *Can we characterize Σ^0_2 and Π^0_1 properties of shifts?*

Motivated by the more general approach of [26], we would like to better understand the relationship between Jeandel's and our results and find a unifying argument that can be developed using computable analysis, as we have done here for shifts.

Question 3. *Is it possible to obtain results based on minimality similar to Theorem 3.2 for other theories, such as group theory and combinatorics, in a unifying framework based on computable analysis?*

Acknowledgements

The authors are grateful to anonymous reviewers, as well as to Pierre Guillon and Guillaume Theyssier for discussions and numerous remarks that helped to significantly improve this article. We particularly thank Pierre Guillon for pointing out links with results from [26].
We gratefully acknowledge funding from the ANR Project IZES-ANR-22-CE40-0011 (Inside Zero Entropy Systems) for this work.

References

[1] Djamel Eddine Amir. *Computability of Topological Spaces*. PhD thesis, Université de Lorraine, 2023.

[2] Djamel Eddine Amir. Spaces with unfixable type, invited speaker, CCA 2024. Computability and Complexity in Analysis (CCA).

[3] Djamel Eddine Amir and Mathieu Hoyrup. Computability of finite simplicial complexes. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 111:1–111:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[4] Djamel Eddine Amir and Mathieu Hoyrup. Comparing computability in two topologies. *The Journal of Symbolic Logic*, pages 1–19, 2023.

[5] Djamel Eddine Amir and Mathieu Hoyrup. Strong computable type. *Computability*, pages 227–269, January 1 2023.

[6] Djamel Eddine Amir and Mathieu Hoyrup. The surjection property and computable type. *Topology and its Applications*, 355:109020, 2024.

[7] Djamel Eddine Amir and Mathieu Hoyrup. Descriptive complexity of topological invariants. *Annals of Pure and Applied Logic*, 176(8):103611, 2025.

[8] Nathalie Aubrun, Sebastián Barbieri, and Emmanuel Jeandel. About the domino problem for subshifts on groups. *Sequences, groups, and number theory*, pages 331–389, 2018.

[9] Nathalie Aubrun, Sebastián Barbieri, and Mathieu Sablik. A notion of effectiveness for subshifts on finitely generated groups. *Theoretical Computer Science*, 661:35–55, 2017.

[10] Alexis Ballier, Bruno Durand, and Emmanuel Jeandel. Structural aspects of tilings. In *STACS 2008*, pages 61–72. IBFI Schloss Dagstuhl, 2008.

[11] Sebastián Barbieri, Nicanor Carrasco-Vargas, and Cristóbal Rojas. Effective dynamical systems beyond dimension zero and factors of sfts. *Ergodic Theory and Dynamical Systems*, pages 1–41, 2024.

[12] Nicolás Bitar. Contributions to the domino problem: Seeding, recurrence and satisfiability. *arXiv preprint arXiv:2312.08911*, 2023.

[13] Antonin Callard and Benjamin Hellouin de Menibus. The aperiodic domino problem in higher dimension. In *39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2022.

[14] Nicanor Carrasco-Vargas. *Subshifts on groups and computable analysis*. PhD thesis, Pontifical Catholic University of Chile, 2024.

[15] T. Ceccherini-Silberstein and M. Coornaert. *Cellular Automata and Groups*. Springer Monographs in Mathematics. Springer Berlin Heidelberg, 2010.

[16] Matea Čelar and Zvonko Iljazović. Computability of products of chainable continua. *Theory of Computing Systems*, 65(2):410–427, Feb 2021.

[17] Douglas Cenzer. Chapter 2 - pi^1_o classes in computability theory. In Edward R. Griffor, editor, *Handbook of Computability Theory*, volume 140 of *Studies in Logic and the Foundations of Mathematics*, pages 37–85. Elsevier, 1999.

[18] Jean-Charles Delvenne, Petr Kurka, and Vincent Blondel. Decidability and universality in symbolic dynamical systems. *Fundamenta Informaticae*, 74(4):463–490, 2006.

[19] Rodney G. Downey and Alexander G. Melnikov. Computably compact metric spaces. *The Bulletin of Symbolic Logic*, 29(2):170–263, 2023.

[20] Francesca Fiorenzi. Semi-strongly irreducible shifts. *Advances in Applied Mathematics*, 32(3):421–438, 2004.

[21] Silvère Gangloff and Benjamin Hellouin de Menibus. Effect of quantified irreducibility on the computability of subshift entropy. *arXiv preprint arXiv:1602.06166*, 2016.

[22] Graham Higman. Subgroups of finitely presented groups. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 262(1311):455–475, 1961.

[23] Michael Hochman. On the dynamics and recursive properties of multidimensional symbolic systems. *Inventiones mathematicae*, 176(1):131–167, Apr 2009.

[24] Zvonko Iljazović and Takayuki Kihara. Computability of subsets of metric spaces. In Vasco Brattka and Peter Hertling, editors, *Handbook of Computability and Complexity in Analysis*, pages 29–69. Springer International Publishing, Cham, 2021.

[25] Zvonko Iljazović and Igor Sušić. Semicomputable manifolds in computable topological spaces. *J. Complex.*, 45:83–114, 2018.

[26] Emmanuel Jeandel. Enumeration reducibility in closure spaces with applications to logic and algebra. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–11. IEEE, 2017.

[27] Emmanuel Jeandel and Pascal Vanier. A characterization of subshifts with computable language. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[28] Andrew S Kahr, Edward F Moore, and Hao Wang. Entscheidungsproblem reduced to the aea case. *Proceedings of the National Academy of Sciences*, 48(3):365–377, 1962.

[29] D. Kerr and H. Li. *Ergodic Theory: Independence and Dichotomies*. Springer Monographs in Mathematics. Springer International Publishing, 2017.

[30] Bruce Kitchens and Klaus Schmidt. Periodic points, decidability and markov subgroups. In *Dynamical Systems: Proceedings of the Special Year held at the University of Maryland, College Park, 1986–87*, pages 440–454. Springer, 2006.

[31] D. A. Lind. *A zeta function for $\mathbb{Z}^d$-actions*, page 433–450. London Mathematical Society Lecture Note Series. Cambridge University Press, 1996.

[32] M Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002.

[33] Roger C Lyndon and Paul E Schupp. *Combinatorial group theory*, volume 89. Springer, 1977.

[34] Joseph S. Miller. Effectiveness for embedded spheres and balls. *Electronic Notes in Theoretical Computer Science*, 66(1):127 – 138, 2002. CCA 2002, Computability and Complexity in Analysis (ICALP 2002 Satellite Workshop).

[35] Ville Salo. Decidability and universality of quasiminimal subshifts. *Journal of Computer and System Sciences*, 89:288–314, 2017.

[36] Alan L. Selman. Arithmetical reducibilities i. *Mathematical Logic Quarterly*, 17(1):335–350, 1971.

[37] Hao Wang. Proving theorems by pattern recognition—ii. *Bell system technical journal*, 40(1):1–41, 1961.

[38] Vedran Čačić, Matea Čelar, Marko Horvat, and Zvonko Iljazović. Computable approximations of semicomputable graphs, 2024.

A Topology on A^G and its hyperspace

A.1 Topology on A^G

To define G-shifts we need to induce A^G with a topology as follows (see [15]).

Definition A.1. Let G be a group and A be an alphabet. We endow A^G with the **pro-discrete** topology obtained by inducing A with the discrete topology (each point in A is open and closed (clopen)) and then taking the product topology.

If $G = \{g_i : i \in \mathbb{N}\}$ is countable, this topology is metrizable using the metric

$$d(x, y) = \inf\{\frac{1}{n} : n \in \mathbb{N}, \text{ and } x(i) = y(i) \text{ for all } i \leq n\}.$$

If G is finitely generated and S is a generating set, they induce a **word length** $|\cdot|_S : G \to \mathbb{N}$ such that if $g \in G$, $|g|_S$ is the length of the shortest word in S^* with $g = w_G$. Consequently, a **word metric** d_S on G can be defined as follows: for every $(g, h) \in G^2$, $d_S(g, h) = |g^{-1}h|_S$. In this case another metric for the pro-discrete topology on A^G is

$$d(x, y) = \inf\{\frac{1}{n} : n \in \mathbb{N}, \text{ and } x(g) = y(g) \text{ for all } |g|_S \leq n\}.$$

Fact A.2 (Section 3.3. in [11])**.** *Let A be an alphabet and G a finitely generated group with decidable word problem. The space A^G can be endowed with a computable metric structure corresponding to the distance d defined above: that is, there exists a dense sequence $(x_i)_{i\in\mathbb{N}} \subseteq A^G$ which is uniformly computable (i.e. there exists an algorithm which given any pair $(i, j) \in \mathbb{N}^2$ and any $n \in \mathbb{N}$ computes some $\alpha \in \mathbb{Q}$ such that $|d(x_i, x_j) - \alpha| < 1/n$).*

A.2 The hyperspace of A^G

Let G be a finitely generated group with decidable word problem and let A be a finite alphabet. We denote by $\mathcal{K}(A^G)$ the **hyperspace** of A^G i.e. the space of all non-empty compact subsets of A^G. It can be endowed with the induced Hausdorff metric (see Section 2.3. in [1]) which induces also a computable metric structure (see [24]) from the computable metric structure defined in Fact A.2. This metric structure induces a topology which is called Vietoris topology, see Section 2.3. in [1].

A **property** is a subset of $\mathcal{K}(A^G)$.

A.2.1 Descriptive complexity for properties in $\mathcal{K}(A^G)$ and $\mathcal{S}(A^G)$

Descriptive complexity for properties in $\mathcal{K}(A^G)$ can be defined in the same way as for properties in $\mathcal{S}(A^G)$ in Definition 2.8. As $\mathcal{S}(A^G)$ is a property in $\mathcal{K}(A^G)$ it is not hard to see its descriptive complexity.

Fact A.3. *$\mathcal{S}(A^G)$ is a Π_1^0 property in $\mathcal{K}(A^G)$.*

Proof. Let $X \in \mathcal{K}(A^G)$ be a set, given an enumeration of $\mathcal{L}(X)$ and its complement, we can semi-decide whether $X \notin \mathcal{S}(A^G)$ i.e. the fact that X is not shift-invariant. Indeed, eventually the machine will enumerate some pattern p and some $g \in G$ such that $[p] \cap X \neq \emptyset$ and $g[p] \cap X = \emptyset$. $\square$

In Section 3.5, the properties we define in $\mathcal{S}(A^G)$ can be defined more generally in $\mathcal{K}(A^G)$, for instance the property of being non-empty is Π_1^0 in $\mathcal{K}(A^G)$.

Now, let us discuss the relation between topologies on $\mathcal{K}(A^G)$ and $\mathcal{S}(A^G)$ and the descriptive complexity of properties.

As $\mathcal{K}(A^G)$ is a computable metric space, let $(k_i)_{i\in\mathbb{N}}$ be a computable dense sequence in $\mathcal{K}(A^G)$. One can enumerate all the balls of the form $B(k_i, r_j)$ with center k_i and radius $r_j \in \mathbb{Q}$, let $(B_k)_{k\in\mathbb{N}}$ be such an enumeration. $\mathcal{S}(A^G)$ can then be endowed with the induced topology as a subspace of $\mathcal{K}(A^G)$.

A property is **effectively open** if there is some c.e. $I \subseteq \mathbb{N}$, such that $P = \bigcup_{i\in I} B_i$.

A property is **effectively closed** if it is the complement of an effectively open property.

Here is a reformulation of descriptive complexity using effective topology.

Fact A.4. *A property in $\mathcal{K}(A^G)$ is Σ_1^0 iff it is effectively open in $\mathcal{K}(A^G)$. It is Π_1^0 if it is effectively closed in $\mathcal{K}(A^G)$. The same is true for properties in $\mathcal{S}(A^G)$.*

A.2.2 Descriptive complexity and upper Vietoris topology

We explained in Fact A.4 that being Π_1^0 in $\mathcal{K}(A^G)$ is equivalent to being effectively closed in the Vietoris topology induced by the Hausdorff metric. The same is true for properties in $\mathcal{S}(A^G)$.

There is a stronger notion that corresponds to P being effectively closed in the upper Vietoris topology. This is equivalent to the existence of an algorithm that, for every $X \in \mathcal{K}(A^G)$, given an enumeration of $\mathcal{L}^c(X)$, halts if and only if $X \notin P$. The difference with Vietoris topology is that the algorithm is given as input only the enumeration of the complement. The same is true for properties in $\mathcal{S}(A^G)$.

The reason why we could assume this stronger notion in the proof of Theorem 3.2 is the following fact, see Proposition 4.4.4. in Amir's PhD thesis [1].

Fact A.5. *If P is a Σ_2^0 property in the Vietoris topology, then there exists some property P' which is Σ_2^0 in the upper Vietoris topology and such that they induce the same minimal elements, i.e. X is P-minimal if and only if X is P'-minimal.*

B Another proof of Theorem 3.2

In [5], a characterization for sets with strong computable type was provided. Our Theorem 3.2 is motivated by this result and, in fact, can be derived from it (though not directly). Let us explain why.

Let G be a finitely generated group with decidable word problem, and let A be a finite alphabet.

First, let us state the result for sets, reformulated without using copies (see Theorem 4.7. in [5]).

Theorem B.1 (A characterization of strong computable type for set)**.** *Let X be a compact subset in A^G. The following are equivalent.*

1. *For every oracle $O \subset \mathbb{N}$, if X is semi-computable relative to O, then it is computable relative to O.*
2. *There exists a Σ_2^0 property P in $\mathcal{K}(A^G)$ such that X is P-minimal.*

Fact B.2. *Effectively closed G-shifts are exactly the semi-computable compact shift-invariant subsets of A^G. (This is not straightforward; it is a theorem, see [9], since for effectively closed shifts we enumerate words, and for semi-computable subsets, we enumerate balls.)*

The results from a recent, yet unpublished work by Carrasco-Vargas, Herrera Nunez and Sablik, (see Proposition 6.12. in [14]) imply that G-shifts with computable languages are exactly the computable shift-invariant compact subsets of A^G. To show that, one needs to see (semi-)computable compact subsets of A^G as computable points in the (upper) Vietoris topology, as explained in Section 2.3. in [1].

We claim that this result can be relativized in the following way.

Proposition B.3. *Let $O \subset \mathbb{N}$ be an oracle, a G-shift has computable language relative to O iff it is a computable point relative to O in the Vietoris topology.*

Let us see how to obtain to Theorem 3.2 from Theorem B.1.

Suppose we have 1. in Theorem 3.2 for some shift X. By Proposition B.3 and the relative version of Fact B.2, we have 1. in Theorem B.1, so we find a Σ_2^0 property P in $\mathcal{K}(A^G)$ such that X is P-minimal. Hence, X is minimal for the property $P' = P \cap \mathcal{S}(A^G)$ which is Σ_2^0 in $\mathcal{S}(A^G)$ (2. in Theorem 3.2). For the other direction, remark that a Σ_2^0 property in $\mathcal{S}(A^G)$ is also Σ_2^0 in $\mathcal{K}(A^G)$, as $\mathcal{S}(A^G)$ is Π_1^0 in $\mathcal{K}(A^G)$ by Fact A.3.