Subshifts, MSO Logic, and Collapsing Hierarchies

Ilkka Törmä

▶ **To cite this version:**

Ilkka Törmä. Subshifts, MSO Logic, and Collapsing Hierarchies. 8th IFIP International Conference on Theoretical Computer Science (TCS), Sep 2014, Rome, Italy. pp.111-122, 10.1007/978-3-662-44602-7_10 . hal-01402035

HAL Id: hal-01402035
https://inria.hal.science/hal-01402035

Submitted on 24 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Subshifts, MSO Logic, and Collapsing Hierarchies

Ilkka Törmä

TUCS – Turku Center for Computer Science
University of Turku, Finland
iatorm@utu.fi

Abstract. We use monadic second-order logic to define two-dimensional subshifts, or sets of colorings of the infinite plane. We present a natural family of quantifier alternation hierarchies, and show that they all collapse to the third level. In particular, this solves an open problem of [Jeandel & Theyssier 2013]. The results are in stark contrast with picture languages, where such hierarchies are usually infinite.

Keywords: subshift, MSO logic, quantifier alternation

1 Introduction

A two-dimensional subshift is a set of colorings of the infinite plane with finitely many colors. Concrete examples are given by sets of *Wang tiles*, or squares with colored edges, introduced by Wang in [13]. The associated *tiling system* consists of all tilings of the plane where overlapping edges have the same color. The initial motivation for Wang tiles was to use a possible algorithm for the infinite tiling problem to recognize tautologies in first-order logic. The tiling problem was proved undecidable by Berger [2], and more undecidability results for tiling systems followed. More recently, strong connections between multidimensional subshifts and computability theory have been found. For example, it was shown in [3], [1] that every vertically constant co-RE subshift can be implemented as a letter-to-letter projection of a tiling system. The topological entropies of tiling systems were characterized in [4] as the right recursively enumerable nonnegative reals. It seems that every conceivable behavior occurs in the class of (projections of) tiling systems, if there is no obvious geometric or computational obstruction.

In this article, we follow the approach of [5,6] and define two-dimensional subshifts by monadic second-order (MSO) logical formulas. We show that certain hierarchies obtained by counting quantifier alternations are finite, solving an open problem posed in [6]. Classes of finite structures defined by MSO formulas have been studied extensively. Examples include finite words, trees, grids and graphs; see [8] and references therein. For words and trees, MSO formulas define exactly the regular languages, and the quantifier alternation hierarchy collapses to the second level. On the other hand, the analogous hierarchy of picture languages was shown to be infinite in [9] and strict in [11]. Although subshifts behave more like sets of words or trees than picture languages in this sense, the reasons are different: MSO-definable languages are regular because the geometry is so simple, while the subshift hierarchy collapses since we can simulate arbitrary computation already on the third level. The concept of constructing subshifts by quantifying over infinite configurations has also been studied in [7] under the name of *multi-choice shift spaces*, and in [12] under the more general framework of *quantifier extensions*. Both formalisms are subsumed by MSO logic.

2 Preliminary Definitions

2.1 Patterns and Subshifts

Fix a finite *alphabet* A. A *pattern* is a map $P : D \to A$ from an arbitrary *domain* $D = D(P) \subset \mathbb{Z}^2$ to A. A pattern with domain $\mathbb{Z}^2$ is a *configuration*, and the set $A^{\mathbb{Z}^2}$ of all configurations is the *full shift over* A. The set of finite patterns over A is denoted by A^{**}, and those with domain $D \subset \mathbb{Z}^2$ by A^D. The restriction of a pattern P to a smaller domain $E \subset D(P)$ is denoted $P|_E$. A pattern P *occurs at* $\boldsymbol{v} \in \mathbb{Z}^2$ in another pattern Q, if we have $\boldsymbol{v} + \boldsymbol{w} \in D(Q)$ and $Q_{\boldsymbol{v}+\boldsymbol{w}} = P_{\boldsymbol{w}}$ for all $\boldsymbol{w} \in D(P)$. We denote $P \sqsubset Q$ if P occurs in Q at some coordinate. For a set of patterns X, we denote $P \sqsubset \mathsf{X}$ if P occurs in some element of X.

A set of finite patterns $F \subset A^{**}$ defines a *subshift* as the set of configurations $\mathsf{X}_F = \{x \in A^{\mathbb{Z}^2} \mid \forall P \in F : P \not\sqsubset x\}$ where no pattern of F occurs. If F is finite, then X_F is *of finite type*, or SFT. The *language* of a subshift $\mathsf{X} \subset A^{\mathbb{Z}^2}$ is $\mathcal{B}(\mathsf{X}) = \{P \in A^{**} \mid P \sqsubset \mathsf{X}\}$. For a finite $D \subset \mathbb{Z}^2$, we denote $\mathcal{B}_D(\mathsf{X}) = \mathcal{B}(\mathsf{X}) \cap A^D$. For $\boldsymbol{v} \in \mathbb{Z}^2$, we denote by $\sigma^{\boldsymbol{v}} : A^{\mathbb{Z}^2} \to A^{\mathbb{Z}^2}$ the *shift by* $\boldsymbol{v}$, defined by $\sigma^{\boldsymbol{v}}(x)_{\boldsymbol{w}} = x_{\boldsymbol{w}+\boldsymbol{v}}$ for all $x \in A^{\mathbb{Z}^2}$ and $\boldsymbol{w} \in \mathbb{Z}^2$. Subshift are invariant under the shift maps.

A *block map* is a function $f : \mathsf{X} \to \mathsf{Y}$ between two subshifts $\mathsf{X} \subset A^{\mathbb{Z}^2}$ and $\mathsf{Y} \subset B^{\mathbb{Z}^2}$ defined by a finite *neighborhood* $D \subset \mathbb{Z}^2$ and a *local function* $F : \mathcal{B}_D(\mathsf{X}) \to B$ which is applied to every coordinate synchronously: $f(x)_{\boldsymbol{v}} = F(x|_{D+\boldsymbol{v}})$ for all $x \in \mathsf{X}$ and $\boldsymbol{v} \in \mathbb{Z}^2$. The image of an SFT under a block map is a *sofic shift*.

Example 1. Let $A = \{0, 1\}$, and let $F \subset A^{**}$ be the set of patterns where 1 occurs twice. Then $\mathsf{X}_F \subset A^{\mathbb{Z}^2}$ is the set of configurations containing at most one letter 1. This subshift is sometimes called the *sunny side up shift*, and it is sofic.

A famous example of an SFT is the *two-dimensional golden mean shift* on the same alphabet, defined by the forbidden patterns $1\,1$ and $\substack{1\\1}$. In its configurations, no two letters 1 can be adjacent, but there are no other restrictions.

2.2 Logical Formulas

We continue the line of research of [5,6], and define subshifts by monadic second-order (MSO) formulas. We now introduce the terminology used in these articles, and then expand upon it. A *structure* is a tuple $\mathfrak{M} = (U, \tau)$, where U is an *underlying set*, and τ a *signature* consisting of functions $f : U^n \to U$ and relations $r \subset U^n$ of different *arities* $n \in \mathbb{N}$. A configuration $x \in A^{\mathbb{Z}^2}$ defines a structure $\mathfrak{M}_x = (\mathbb{Z}^2, \tau_A)$, whose signature τ_A contains the following objects:

- Four unary functions, named North, South, East and West, and called *adjacency functions* in this article. They are interpreted in the structure $\mathfrak{M}_x$ as $\mathsf{North}^{\mathfrak{M}_x}((a,b)) = (a, b+1)$, $\mathsf{East}^{\mathfrak{M}_x}((a,b)) = (a+1, b)$ and so on for $a, b \in \mathbb{Z}$.
- For each symbol $a \in A$, a unary *symbol predicate* P_a. It is interpreted as $P_a^{\mathfrak{M}_x}(\boldsymbol{v})$ for $\boldsymbol{v} \in \mathbb{Z}^2$ being true if and only if $x_{\boldsymbol{v}} = a$.

The MSO formulas that we use are defined with the signature τ_A as follows.

- A *term (of depth $k \in \mathbb{N}$)* is a chain of k nested applications of the adjacency functions to a first-order variable.
- An *atomic formula* is either $t = t'$ or $P(t)$, where t and t' are terms and P is either a symbol predicate or a second-order variable.
- A *formula* is either an atomic formula, or an application of a logical connective $(\wedge, \vee, \neg, \ldots)$ or first- or second-order quantification to other formulas.

The *radius* of a formula is the maximal depth of a term in it. First-order variables (usually denoted $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_\ell$) hold elements of $\mathbb{Z}^2$, and second-order variables hold subsets of $\mathbb{Z}^2$. Formulas without second-order variables are *first-order.*

Let ϕ be a closed MSO formula, and let $D \subset \mathbb{Z}^2$. A configuration $x \in A^{\mathbb{Z}^2}$ is a *D-model* for ϕ, denoted $x \models_D \phi$, if ϕ is true in the structure $\mathfrak{M}_x$ when the quantification of the first-order variables in ϕ is restricted to D. If $D = \mathbb{Z}^2$, then we denote $x \models \phi$ and say that *x models ϕ*. We define a set of configurations $\mathsf{X}_\phi = \{x \in A^{\mathbb{Z}^2} \mid x \models \phi\}$, which is always shift-invariant, but may not be a subshift. A subshift is *MSO-definable* if it equals X_ϕ for some MSO formula ϕ.

As we find it more intuitive to quantify over configurations than subsets of $\mathbb{Z}^2$, and we later wish to quantify over the configurations of specific subshifts, we introduce the following definitions.

- The notations $\forall X[\mathsf{X}]$ and $\exists X[\mathsf{X}]$ (read *for all (or exists) X in* X) define a new *configuration variable* X, which represents a configuration of a subshift $\mathsf{X} \subset B^{\mathbb{Z}^2}$ over a new alphabet B.
- For $X[\mathsf{X}]$ quantified as above, $b \in B$ and a term t, the notation $X_t = b$ defines an atomic formula that is true if and only if the configuration represented by X has the letter b at the coordinate represented by t.

MSO formulas with configuration variables instead of ordinary second-order variables are called *extended MSO formulas*, and the relation $\models$ is extended to them. We state without proof that if the subshifts occurring in an extended MSO formula ϕ are MSO-definable, then so is X_ϕ. Conversely, we can convert an MSO formula to an extended MSO formula by replacing every second-order variable with a configuration variable over the binary full shift. Unless stated otherwise, by second-order variables (usually denoted $X_1, \ldots, X_n$) we mean configuration variables, and by MSO formulas we mean extended MSO formulas.

Example 2. The two-dimensional golden mean shift is defined by the formula

$$\forall \boldsymbol{n} \big(P_1(\boldsymbol{n}) \implies \big(P_0(\mathsf{North}(\boldsymbol{n})) \wedge P_0(\mathsf{East}(\boldsymbol{n})) \big) \big).$$

Also, the sunny side up shift is defined by the formula

$$\forall \boldsymbol{m} \forall \boldsymbol{n} \big(P_1(\boldsymbol{n}) \implies (P_0(\boldsymbol{m}) \vee \boldsymbol{m} = \boldsymbol{n}) \big).$$

Another way to define the sunny side up shift is to use a second-order quantifier:

$$\begin{aligned} \exists U \forall \boldsymbol{n} \big(U(\boldsymbol{n}) &\iff \big(U(\mathsf{North}(\boldsymbol{n})) \wedge U(\mathsf{West}(\boldsymbol{n})) \big) \big) \\ \wedge \big(P_1(\boldsymbol{n}) &\implies \big(U(\boldsymbol{n}) \wedge \neg U(\mathsf{South}(\boldsymbol{n})) \wedge \neg U(\mathsf{East}(\boldsymbol{n})) \big) \big). \end{aligned}$$

We can produce an equivalent extended MSO formula, as per the above remark:

$$\begin{aligned} \exists X[\{0,1\}^{\mathbb{Z}^2}] \forall \boldsymbol{n} \big(X_{\boldsymbol{n}} = 1 &\iff (X_{\mathsf{North}(\boldsymbol{n})} = 1 \wedge X_{\mathsf{West}(\boldsymbol{n})} = 1) \big) \\ \wedge \big(P_1(\boldsymbol{n}) &\implies (X_{\boldsymbol{n}} = 1 \wedge X_{\mathsf{South}(\boldsymbol{n})} = 0 \wedge X_{\mathsf{East}(\boldsymbol{n})} = 0) \big). \end{aligned}$$

2.3 Computability Theory

We recall the *arithmetical hierarchy*, a classical reference for which is [10]. A first-order arithmetical formula over $\mathbb{N}$ is Π^0_0 (equivalently, Σ^0_0), if it only contains bounded quantifiers (of the form $\forall n \leq k$ or $\exists n \leq k$). The formula is Π^0_{k+1} (Σ^0_{k+1}) if it is of the form $\forall n_1 \cdots \forall n_\ell \phi$ ($\exists n_1 \cdots \exists n_\ell \phi$) where ϕ is Σ^0_k (Π^0_k, respectively). Every such formula is equivalent to a Π^0_k or Σ^0_k one, and if it defines a subset of $\mathbb{N}$, that set is given the same classification. Completeness and hardness in the classes are defined using Turing reductions. For all $k \in \mathbb{N}$, the class $\Delta^0_{k+1} = \Pi^0_{k+1} \cap \Sigma^0_{k+1}$ contains exactly the languages decidable by Turing machines with Π^0_k oracles. Also, Σ^0_1 is the class of recursively enumerable subsets of $\mathbb{N}$.

When classifying subsets of countable sets other than $\mathbb{N}$, we assume they are in some natural and computable bijection with $\mathbb{N}$. For example, a co-recursively enumerable set of finite patterns is Π^0_1. A subshift X is given the same classification as its language $\mathcal{B}(\mathsf{X})$. If X is Π^0_k for some $k \in \mathbb{N}$, then it can be defined by a Σ^0_k set of forbidden patterns (the complement of $\mathcal{B}(\mathsf{X})$), and a subshift defined by such a set is always Π^0_{k+1}. In particular, SFTs and sofic shifts are Π^0_1.

Remark 1. We use several hierarchies of subshifts obtained by counting quantifier alternations in different kinds of formulas, and the notation for them can be confusing. In general, classes defined by computability conditions (the arithmetical hierarchy) are denoted by Π and Σ, while classes defined by MSO formulas via the modeling relation are denoted by $\bar{\Pi}$ and $\bar{\Sigma}$.

3 Hierarchies of MSO-Definable Subshifts

In this section, we recall the definition of a hierarchy of subshift classes defined in [5,6], and then generalize it. We also state some general lemmas.

Definition 1. *Let C be a class of subshifts. An MSO formula ψ is* over C with universal first-order quantifiers*, or C-u-MSO for short, if it is of the form*

$$\psi = Q_1 X_1[\mathsf{X}_1] Q_2 X_2[\mathsf{X}_2] \cdots Q_n X_n[\mathsf{X}_n] \forall \boldsymbol{n}_1 \cdots \forall \boldsymbol{n}_\ell \phi,$$

where each Q_i is a quantifier, $\mathsf{X}_i \in C$, and ϕ is quantifier-free. If there are k quantifier alternations and Q_1 is the existential quantifier $\exists$, then ψ is called $\bar{\Sigma}_k[C]$, and if Q_1 is $\forall$, then ψ is $\bar{\Pi}_k[C]$. The set X_ψ is given the same classification. If C is the singleton class containing only the binary full shift $\{0,1\}^{\mathbb{Z}^2}$, then ψ is called u-MSO, and we denote $\bar{\Sigma}_k[C] = \bar{\Sigma}_k$ and $\bar{\Pi}_k[C] = \bar{\Pi}_k$. The classes $\bar{\Sigma}_k$ and $\bar{\Pi}_k$ for $k \in \mathbb{N}$ form the u-MSO hierarchy*.*

In [6], the u-MSO hierarchy was denoted by the letter $\mathcal{C}$, but we use the longer name for clarity. In the rest of this article, C denotes an arbitrary class of subshifts, unless otherwise noted. We proceed with the following result, stated for u-MSO formulas in [6]. We omit the proof, as it is essentially the same.

Theorem 1 (Generalization of Theorem 13 of [6]). *Let ϕ be a C-u-MSO formula over an alphabet A. Then for all $x \in A^{\mathbb{Z}^2}$, we have $x \models \phi$ if and only if $x \models_D \phi$ for every finite domain $D \subset \mathbb{Z}^2$.*

Corollary 1. *Every C-u-MSO formula ϕ over an alphabet A defines a subshift.*

Proof. Let $r \in \mathbb{N}$ be the radius of ϕ. By Theorem 1, we have $\mathsf{X}_\phi = \mathsf{X}_F$, where $F = \{x|_{D+[-r,r]^2} \mid D \subset \mathbb{Z}^2 \text{ finite}, x \in A^{\mathbb{Z}^2}, x \not\models_D \phi\}$. $\square$

Corollary 2. *For all $k, n \in \mathbb{N}$, we have $\bar{\Pi}_n[\Pi^0_k] \subset \Pi^0_{k+1}$. In particular, the u-MSO hierarchy only contains Π^0_1 subshifts.*

Proof. Let $\phi = \forall X_1[\mathsf{X}_1]\exists X_2[\mathsf{X}_2] \ldots Q_n X_n[\mathsf{X}_n]\psi$ be a $\bar{\Pi}_n[\Pi^0_k]$ formula, where each $\mathsf{X}_i \subset A_i^{\mathbb{Z}^2}$ is a Π^0_k subshift and ψ is first-order. Then the product subshift $\prod_{i=1}^n \mathsf{X}_i$ is also Π^0_k. Let $P \in A^{**}$ be a finite pattern. Theorem 1, together with a basic compactness argument, implies that $P \in \mathcal{B}(\mathsf{X}_\phi)$ holds if and only if for all finite domains $D(P) \subset D \subset \mathbb{Z}^2$, there exists a configuration $x \in A^{\mathbb{Z}^2}$ such that $x|_{D(P)} = P$ and $x \models_D \phi$. For a fixed D, denote this condition by $C_P(D)$.

We show that deciding $C_P(D)$ for given pattern P and domain D is Δ^0_{k+1}. Denote $E = D+[-r,r]^2$, where $r \in \mathbb{N}$ is the radius of ϕ, and let $L = \mathcal{B}_E(\prod_{i=1}^n \mathsf{X}_i)$. For a configuration $x \in A^{\mathbb{Z}^2}$, the condition $x \models_D \phi$ only depends on the finite pattern $x|_E \in A^E$, and is computable from it and the set L. Thus $C_P(D)$ is equivalent to the existence of a pattern $Q \in A^E$ such that $x|_E = Q$ implies $x \models_D \phi$ for all $x \in A^{\mathbb{Z}^2}$. Moreover, this can be decided by the oracle Turing machine that computes L using a Π^0_k oracle, and then goes through the finite set A^E, searching for such a Q. Thus the condition $C_P(D)$ is Δ^0_{k+1}, which implies that deciding $P \in \mathcal{B}(\mathsf{X}_\phi)$ is Π^0_{k+1}. $\square$

Finally, if the final second-order quantifier of a u-MSO formula is universal, it can be dropped. This does not hold for C-u-MSO formulas in general. We omit the proof, as it is essentially the same as that of [6, Lemma 7].

Lemma 1. *If $k \geq 1$ is odd, then $\bar{\Pi}_k = \bar{\Pi}_{k-1}$, and if it is even, then $\bar{\Sigma}_k = \bar{\Sigma}_{k-1}$.*

Example 3. Define the *mirror shift* $\mathsf{M} \subset \{0,1,\#\}^{\mathbb{Z}^2}$ by the forbidden patterns $\substack{a \\ \#}$ and $\substack{\# \\ a}$ for $a \neq \#$, every pattern $\{\mathbf{0} \mapsto \#, (n,0) \mapsto \#\}$, and every pattern $\{(-n,0) \mapsto a, \mathbf{0} \mapsto \#, (n,0) \mapsto b\}$ for $n \in \mathbb{N}$ and $a \neq b$. A 'typical' configuration of M contains one infinite column of #-symbols, whose left and right sides are mirror images of each other. It is well-known that M is not sofic. We show that it can be implemented by an SFT-u-MSO formula $\psi = \forall X[\mathsf{X}]\forall \boldsymbol{n}_1 \forall \boldsymbol{n}_2 \forall \boldsymbol{n}_3 \phi$ in the class $\bar{\Pi}_1[\text{SFT}]$. This also shows that Lemma 1 fails outside the u-MSO hierarchy.

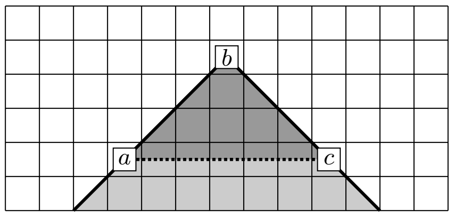

Fig. 1. A pattern of X in Example 3, containing its entire alphabet.

Let X be the SFT whose alphabet is seen in Figure 1, defined by the obvious 2×2 forbidden patterns. Define the formula ϕ as $\phi_1 \wedge (\phi_2 \implies \phi_3)$, where

$$\begin{aligned}
\phi_1 &= P_\#(\boldsymbol{n}_2) \iff P_\#(\mathsf{North}(\boldsymbol{n}_2)) \\
\phi_2 &= X_{\boldsymbol{n}_1} = a \wedge X_{\boldsymbol{n}_2} = b \wedge X_{\boldsymbol{n}_3} = c \wedge P_\#(\boldsymbol{n}_2) \\
\phi_3 &= \neg P_\#(\boldsymbol{n}_1) \wedge \neg P_\#(\boldsymbol{n}_3) \wedge (P_0(\boldsymbol{n}_1) \iff P_0(\boldsymbol{n}_3))
\end{aligned}$$

It is easy to see that the subshift X_ψ is exactly M, with ψ defined as above.

4 The u-MSO Hierarchy

The u-MSO hierarchy is a quite natural hierarchy of MSO-definable subshifts. Namely, the lack of existential first-order quantification makes it easy to prove that every u-MSO formula defines a subshift, and quantifier alternations give rise to interesting hierarchies in many contexts. The following is already known.

Theorem 2 ([6]). *The class of subshifts defined by formulas of the form $\forall \boldsymbol{n}\phi$, where ϕ is first-order, is exactly the class of SFTs. The class $\bar{\Pi}_0 = \bar{\Sigma}_0$ consists of the* threshold counting shifts*, which are obtained from subshifts of the form $\{x \in A^{\mathbb{Z}^2} \mid P$ occurs in x at most n times$\}$ for $P \in A^{**}$ and $n \in \mathbb{N}$ using finite unions and intersections. Finally, the class $\bar{\Sigma}_1$ consists of exactly the sofic shifts.*

We show that the hierarchy collapses to the third level, which consists of exactly the Π^0_1 subshifts. This gives negative answers to the questions posed in [6] of whether the hierarchy is infinite, and whether it only contains sofic shifts.

Theorem 3. *For all $n \geq 2$ we have $\Pi_1^0 = \bar{\Pi}_n$.*

Proof. As we have $\bar{\Pi}_n \subset \Pi_1^0$ by Corollary 2, and clearly $\bar{\Pi}_n \subset \bar{\Pi}_{n+1}$ also holds, it suffices to prove $\Pi_1^0 \subset \bar{\Pi}_2$. Let thus $\mathsf{X} \subset A^{\mathbb{Z}^2}$ be a Π_1^0 subshift. We construct an MSO formula of the form $\phi = \forall Y[B^{\mathbb{Z}^2}]\exists Z[C^{\mathbb{Z}^2}]\forall \boldsymbol{n}\psi(\boldsymbol{n}, Y, Z)$ such that $\mathsf{X}_\phi = \mathsf{X}$.

The main idea is the following. We use the universally quantified configuration Y to specify a finite square $R \subset \mathbb{Z}^2$ and a word $w \in A^*$, which may or may not encode the pattern x_R of a configuration $x \in A^{\mathbb{Z}^2}$. The existentially quantified Z enforces that either w does not correctly encode x_R, of that it encodes *some* pattern of $\mathcal{B}(\mathsf{X})$. As R and w are arbitrary and universally quantified, this guarantees $x \in \mathsf{X}$. The main difficulty is that Y comes from a full shift, so we have no control over it; there may be infinitely many squares, or none at all.

First, we define an auxiliary SFT $\mathsf{Y} \subset B^{\mathbb{Z}^2}$, whose configurations contain the aforementioned squares. The alphabet B consists of the tiles seen in Figure 2, where every w_i ranges over A, and it is defined by the set F_Y of 2×2 forbidden patterns where some colors or lines of neighboring tiles do not match. A configuration of Y contains at most one maximal pattern colored with the lightest gray in Figure 2, and if it is finite, its domain is a square. We call this domain the *input square*, and the word $w \in A^*$ that lies above it is called the *input word*.

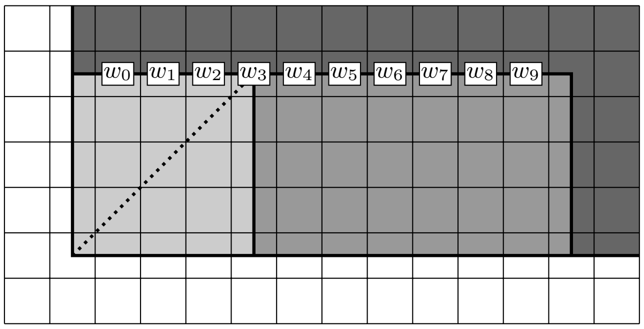

Fig. 2. A pattern of Y. In this example, the input word $w \in A^*$ is of length 10.

We now define another SFT S, this time on the alphabet $A \times B \times C$. The alphabet C is more complex than B, and we specify it in the course of the construction. The idea is to simulate a computation in the third layer to ensure that if the second layer contains a valid configuration of Y and the input word encodes the contents of the input square in the first layer, then that square pattern is in $\mathcal{B}(\mathsf{X})$. We also need to ensure that a valid configuration exists even if the encoding is incorrect, or if second layer is not in Y. For this, every locally valid square pattern of Y containing an input square will be covered by another square pattern in the third layer, inside which we perform the computations. We will force this pattern to be infinite if the second layer is a configuration of Y.

Now, we describe a configuration $(x, y, z) \in \mathsf{S}$. The coordinates of every 2×2 rectangle $R \subset \mathbb{Z}^2$ with $y|_R \in F_{\mathsf{Y}}$ are called *defects*. A non-defect coordinate $\boldsymbol{v} \in \mathbb{Z}^2$ such that $y_{\boldsymbol{v}} =$ ◳ is called a *seed*. Denote $C = C_1 \cup C_2$, where C_1 is the set of tiles depicted in Figure 3 (a). Their adjacency rules in S are analogous to those of Y. The rules of S also force the set of seeds to coincide with the coordinates $\boldsymbol{v} \in \mathbb{Z}^2$ such that $z_{\boldsymbol{v}} =$ ◳. These coordinates are the southwest corners of *computation squares* in z, whose square shape is again enforced by a diagonal signal. The southwest half of a computation square is colored with letters of C_2. See Figure 3 (b) for an example of a computation square.

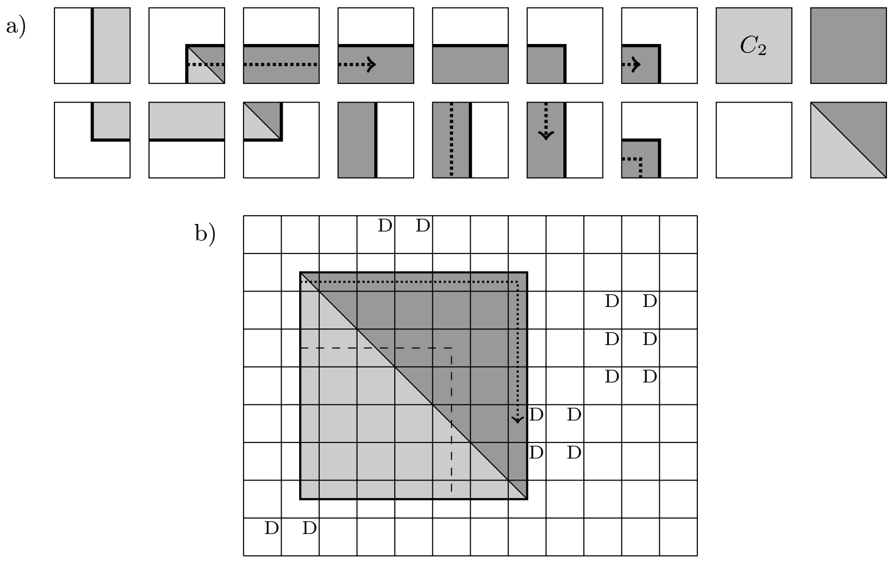

Fig. 3. The alphabet C (a) and a pattern of the third layer of S (b), with the elements of C_2 represented by the featureless light gray tiles. The dashed line represents the border of an input square on the second layer. Defects are marked with a small D.

A computation square may not contain defects or coordinates $\boldsymbol{v} \in \mathbb{Z}^2$ such that $y_{\boldsymbol{v}} = \square$ except on its north or east border, and conversely, one of the borders will contain a defect. This is enforced by a signal emitted from the northwest corner of the square (the dotted line in Figure 3 (b)), which travels along the north and east borders, and disappears when it encounters a defect.

We now describe the set C_2, and for that, let M be a Turing machine with input alphabet $\Sigma = A \times (A \cup \{0, 1, \#\})$ and two initial states q_1 and q_2. This machine is simulated on the southwest halves of the computation squares in a standard way, and we will fix its functionality later. The alphabet C_2 is shown in Figure 4. Note that the colors and lines in C_2 are disjoint from those in C_1, even though the figures suggest otherwise. The idea is to initialize the machine M

with either the input word (if it correctly encodes the input square), or a proof that the encoding is incorrect, in the form of one incorrectly encoded symbol.

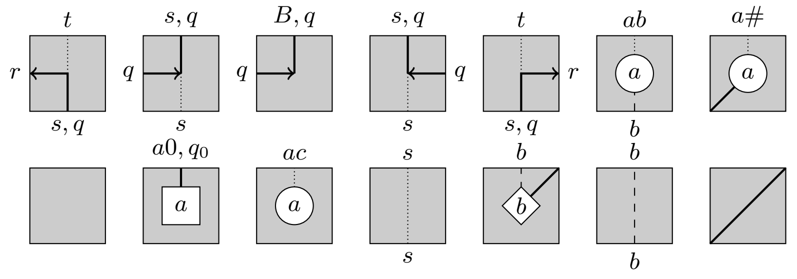

Fig. 4. The sub-alphabet C_2. The letters a and b range over A, c can be 0 or 1, the letter s over the tape alphabet of M, the letter q_0 can be either of the initial states q_1 and q_2, and in the first (fourth) tile on the top row we require that the machine M writes $t \in \Sigma$ on the tape, switches to state r and steps to the left (right, respectively) when reading the letter $s \in \Sigma$ in state q.

The white squares and circles of C_2 must be placed on the letters of the input word $w \in A^*$ of the computation square, the square on the leftmost letter and circles on the rest. The A-letters of these tiles must match the letters of w, and the second component is 1 if the tile lies on the corner of the input square, 0 if not, $b \in A$ in the presence of a vertical signal, and # in the presence of a diagonal signal. Such signals are sent by a white diamond tile (called a *candidate error*), which can only be placed on the interior tiles of the input square, and whose letter must match the letter on the first layer x. Other tiles of C_2 simulate the machine M, which can never halt in a valid configuration. See Figure 5 for a visualization. We also require that for a pattern $\substack{c_2\\c_1}$ to be valid, where $c_i \in C_i$ for $i \in \{1, 2\}$, the tile c_2 should have a gray south border with no lines. Other adjacency rules between tiles of C_1 and C_2 are explained by Figure 3 (a).

We now describe the machine M. Note first that from an input $u \in \Sigma^*$ one can deduce the input word $w \in A^*$, the height $h \in \mathbb{N}$ of the input square, and the positions and contents of all candidate errors. Now, when started in the state q_1, the machine checks that there are no candidate errors at all, that $|w| = h^2$, and that the square pattern $P \in A^{h \times h}$, defined by $P_{(i,j)} = w_{ih+j}$ for all $i, j \in [0, h-1]$, is in $\mathcal{B}(\mathsf{X})$. If all this holds, M runs forever (the check for $P \in \mathcal{B}(\mathsf{X})$ can indeed take infinitely many steps). When started in q_2, the machine checks that there is exactly one candidate error at some position $(i, j) \in [0, h-1]^2$ of the input square containing some letter $b \in A$, and that one of $|w| \neq h^2$ or $w_{ih+j} \neq b$ holds. If this is the case, M enters an infinite loop, and halts otherwise.

The definition of S is now complete, and it can be realized using a set F of forbidden patterns of size 3×3. We define the quantifier-free formula $\psi(\boldsymbol{n}, Y, Z)$ as $\neg \bigvee_{P \in F} \psi_P$, where ψ_P states that the pattern P occurs at the coordinate

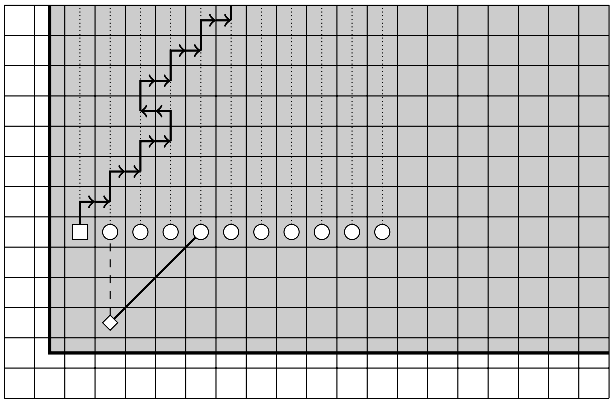

Fig. 5. An infinite computation square with an input word of length 11 and a single candidate error.

$\boldsymbol{n}$. This is easily doable using the adjacency functions, color predicates and the variables Y and Z. If we fix some values $y \in B^{\mathbb{Z}^2}$ and $z \in C^{\mathbb{Z}^2}$ for the variables Y and Z, then $x \models \forall \boldsymbol{n} \psi(\boldsymbol{n}, y, z)$ holds for a given $x \in A^{\mathbb{Z}^2}$ if and only if $(x, y, z) \in \mathsf{S}$.

Let $x \in A^{\mathbb{Z}^2}$ be arbitrary. We need to show that $x \models \phi$ holds if and only if $x \in \mathsf{X}$. Suppose first that x models ϕ, and let $\boldsymbol{v} \in \mathbb{Z}^2$ and $h \geq 1$. Let $y \in \mathsf{Y}$ be a configuration whose input square has interior $D = \boldsymbol{v} + [0, h-1]^2$, and whose input word correctly encodes the pattern $x|_D$. By assumption, there exists $z \in C^{\mathbb{Z}^2}$ such that $(x, y, z) \in \mathsf{S}$, so that the southwest neighbor of $\boldsymbol{v}$ is the southwest corner of a computation square in z, which is necessarily infinite, since no defects occur in y. In this square, M runs forever, and it cannot be initialized in the state q_2 as the encoding of the input square is correct. Thus its computation proves that $x|_D \in \mathcal{B}(\mathsf{X})$. Since D was an arbitrary square domain, we have $x \in \mathsf{X}$.

Suppose then $x \in \mathsf{X}$, and let $y \in B^{\mathbb{Z}^2}$ be arbitrary. We construct a configuration $z \in C^{\mathbb{Z}^2}$ such that $(x, y, z) \in \mathsf{S}$, which proves $x \models \phi$. First, let $S \subset \mathbb{Z}^2$ be the set of seeds in y, and for each $\boldsymbol{s} \in S$, let $\ell(\boldsymbol{s}) \in \mathbb{N} \cup \{\infty\}$ be the height of the maximal square $D(\boldsymbol{s}) = \boldsymbol{s} + [0, \ell(\boldsymbol{s}) - 1]^2$ whose interior contains no defects. We claim that $D(\boldsymbol{s}) \cap D(\boldsymbol{r}) = \emptyset$ holds for all $\boldsymbol{s} \neq \boldsymbol{r} \in S$. Suppose the contrary, and let $\boldsymbol{v} \in D(\boldsymbol{s}) \cap D(\boldsymbol{r})$ be lexicographically minimal. Then $\boldsymbol{v}$ is on the south border of $D(\boldsymbol{s})$ and the west border of $D(\boldsymbol{r})$ (or vice versa). Since these borders contain no defects, $y_{\boldsymbol{v}}$ is a south border tile and a west border tile, a contradiction.

Now, we can define every $D(\boldsymbol{s})$ to be a computation square in z. If it contains an input square and an associated input word which correctly encodes its contents, we initialize the simulated machine M in the state q_1. Then the computation does not halt, since the input square contains a pattern of $\mathcal{B}(\mathsf{X})$. Otherwise, we initialize M in the state q_2, and choose a single candidate error from the input square such that it does not halt, and thus produces no forbidden patterns. Then $(x, y, z) \in \mathsf{S}$, completing the proof. $\square$

We have now characterized every level of the u-MSO hierarchy. The first level $\bar{\Pi}_0 = \bar{\Sigma}_0$ contains the threshold counting shifts and equals $\bar{\Pi}_1$ by Lemma 1, the class $\bar{\Sigma}_1 = \bar{\Sigma}_2$ contains the sofic shifts, and the other levels coincide with Π^0_1.

The quantifier alternation hierarchy of MSO-definable picture languages was shown to be strict in [11]. It is slightly different from the u-MSO hierarchy, as existential first-order quantification is allowed. However, in the case of pictures we know the following. Any MSO formula $\mathcal{Q}_L \exists \boldsymbol{n} \mathcal{Q}_R \phi$, where $\mathcal{Q}_L$ and $\mathcal{Q}_R$ are strings of quantifiers, is equivalent to a formula of the form $\mathcal{Q}_L \exists X \mathcal{Q}_R \forall \boldsymbol{n} \psi$, where ϕ and ψ are quantifier-free. See [8, Section 4.3] for more details. Thus the analogue of the u-MSO hierarchy for picture languages is infinite. The proof of the result of [11] relies on the fact that one can simulate computation within the pictures, and the maximal time complexity depends on the number of alternations. In the case of infinite configurations, this argument naturally falls apart.

Finally, Theorem 3 has the following corollary (which was also proved in [6]).

Corollary 3. *Every Π^0_1 subshift is MSO-definable.*

5 Other C-u-MSO Hierarchies

Next, we generalize Theorem 3 to hierarchies of Π^0_k-u-MSO formulas. The construction is similar to the above but easier, since we can restrict the values of the variable Y to lie in a geometrically well-behaved subshift.

Theorem 4. *For all $k \geq 1$ and $n \geq 2$ we have $\Pi^0_{k+1} = \bar{\Pi}_n[\Pi^0_k]$. Furthermore, $\Pi^0_2 = \bar{\Pi}_n[\mathrm{SFT}]$ for all $n \geq 2$.*

Proof (sketch). As in Theorem 3, it suffices to show that for a given Π^0_{k+1} subshift $\mathsf{X} \subset A^{\mathbb{Z}^2}$, there is a $\bar{\Pi}_2[\Pi^0_k]$ formula $\phi = \forall Y[\mathsf{Y}]\exists Z[\mathsf{Z}]\forall \boldsymbol{n} \psi$ such that $\mathsf{X}_\phi = \mathsf{X}$. In our construction, $\mathsf{Y} \subset B^{\mathbb{Z}^2}$ is a Π^0_k subshift and $\mathsf{Z} = C^{\mathbb{Z}^2}$ is a full shift.

For a square pattern $P \in A^{h \times h}$, define the word $w(P) \in A^{h^2}$ by $w_{ih+j} = P_{(i,j)}$ for all $i, j \in [0, h-1]$. Let $R \subset A^* \times \mathbb{N}$ be a Π^0_k predicate such that the set

$$F = \{P \in A^{h \times h} \mid h \in \mathbb{N}, \exists n \in \mathbb{N} : R(w(P), n)\}$$

satisfies $\mathsf{X}_F = \mathsf{X}$. As in Theorem 3, configurations of Y may contain one input square with an associated input word. This time, the input word is of the form $w\#^n$ for some $w \in A^*$, $n \in \mathbb{N}$ and a new symbol $\#$. As Y is Π^0_k, we can enforce that $R(w, n)$ holds, so that w does *not* encode any square pattern of X. This can be enforced by SFT rules if $k = 1$: a simulated Turing machine checks $R(w, n)$ by running forever if it holds. As before, the existential layer Z enforces that w does *not* correctly encode the contents of the input square in the first layer.

Let $x \in \mathsf{X}$ and $y \in \mathsf{Y}$ be arbitrary. If y has a finite input square $D \in \mathbb{Z}^2$ and input word $w\#^n$, then $w \in A^*$ cannot correctly encode the pattern $x|_D \in \mathcal{B}(\mathsf{X})$, and thus a valid choice for the variable Z exists. Degenerate cases of y (with, say, an infinite input square) are handled as in Theorem 3. Thus we have $x \models \phi$. Next, suppose that $x \notin \mathsf{X}$, so there is a square domain $D \subset \mathbb{Z}^2$ with $x|_D \notin \mathcal{B}(\mathsf{X})$.

Construct $y \in \mathsf{Y}$ such that the input square has domain D, the word $w \in A^*$ correctly encodes $x|_D$, and the number $n \in \mathbb{N}$ of #-symbols is such that $R(w, n)$ holds. For this value of Y, no valid choice for Z exists, and thus $x \not\models \phi$. $\square$

Corollary 3, Theorem 4 and a simple induction argument show the following.

Corollary 4. *For every $k \in \mathbb{N}$, every Π^0_k subshift is MSO-definable.*

However, note that the converse does not hold, since one can construct an MSO-formula defining a subshift whose language is not Π^0_k for any $k \in \mathbb{N}$.

Acknowledgments

I am thankful to Emmanuel Jeandel for introducing me to [5,6] and the open problems therein, and to Ville Salo for many fruitful discussions.

References

1. Nathalie Aubrun and Mathieu Sablik. Simulation of effective subshifts by two-dimensional subshifts of finite type. *Acta Appl. Math.*, 126(1):35–63, August 2013.
2. Robert Berger. The undecidability of the domino problem. *Mem. Amer. Math. Soc. No.*, 66, 1966. 72 pages.
3. Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *J. Comput. System Sci.*, 78(3):731–764, 2012.
4. Michael Hochman and Tom Meyerovitch. A characterization of the entropies of multidimensional shifts of finite type. *Ann. of Math. (2)*, 171(3):2011–2038, 2010.
5. Emmanuel Jeandel and Guillaume Theyssier. Subshifts, languages and logic. In *Developments in language theory*, volume 5583 of *Lecture Notes in Comput. Sci.*, pages 288–299. Springer, Berlin, 2009.
6. Emmanuel Jeandel and Guillaume Theyssier. Subshifts as models for MSO logic. *Inform. and Comput.*, 225:1–15, 2013.
7. Erez Louidor, Brian Marcus, and Ronnie Pavlov. Independence entropy of $\mathbb{Z}^d$-shift spaces. *Acta Applicandae Mathematicae*, pages 1–21, 2013.
8. Oliver Matz and Nicole Schweikardt. Expressive power of monadic logics on words, trees, pictures, and graphs. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata*, volume 2 of *Texts in Logic and Games*, pages 531–552. Amsterdam University Press, 2008.
9. Oliver Matz and Wolfgang Thomas. The monadic quantifier alternation hierarchy over graphs is infinite. In *In Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 236–244. IEEE, 1997.
10. G.E. Sacks. *Higher recursion theory.* Perspectives in mathematical logic. Springer-Verlag, 1990.
11. Nicole Schweikardt. The monadic quantifier alternation hierarchy over grids and pictures. In *Computer science logic (Aarhus, 1997)*, volume 1414 of *Lecture Notes in Comput. Sci.*, pages 441–460. Springer, Berlin, 1998.
12. Ilkka Törmä. Quantifier Extensions of Multidimensional Sofic Shifts. *ArXiv e-prints*, January 2014.
13. Hao Wang. Proving theorems by pattern recognition II. *Bell System Technical Journal*, 40:1–42, 1961.