Group-Walking Automata

Ville Salo, Ilkka Törmä

Group-Walking Automata

Ville Salo[1] and Ilkka Törmä[2]

[1] Center of Mathematical Modeling
University of Chile
vsalo@dim.uchile.cl
[2] TUCS – Turku Centre for Computer Science
University of Turku, Finland
iatorm@utu.fi

Abstract. In the setting of symbolic dynamics on discrete finitely generated infinite groups, we define a model of multi-headed finite automata that walk on Cayley graphs, and use it to define subshifts. We characterize the torsion groups (also known as periodic groups) as those on which the group-walking automata are strictly weaker than Turing machines.

Keywords: group-walking automaton, torsion group, periodic group, multi-headed automaton, subshift

1 Introduction

One of the central objects in symbolic dynamics is the dynamical system S^G (where G is a discrete group and S a finite alphabet), called the *full shift*, where G acts by translations. In particular, one studies its subsystems, usually called *subshifts*, and classes of such subsystems. Some of the important classes studied are the SFTs (subshifts defined by a finite set of forbidden patterns), sofic shifts (the factors of SFTs) and the effective, or Π_1^0 subshifts (defined by a recursively enumerable set of forbidden patterns). SFTs and sofic shifts are natural objects to study on all groups, and a robust notion of effectiveness of subshifts on arbitrary groups is given in [2] (see also Section 5).

In this paper, continuing the work in [9], we define some new families of subshifts on an arbitrary (discrete finitely generated infinite) group G. Namely, we discuss the class of subshifts defined by certain multi-headed automata that walk on the Cayley graph of the group G. We have studied the case $G = \mathbb{Z}^d$ in [9], the main result being that three-headed finite-state automata define the same subshifts as general Turing machines.[3] It turns out that up to notational complications and a few simple tricks, the same result can be shown on all groups containing a copy of $\mathbb{Z}$. We show this in Theorem 1.

Most finitely generated groups of practical interest contain a copy of $\mathbb{Z}$. For example, in addition to infinite (finitely generated) abelian groups, this is true

[3] In the earlier article [4], essentially the same observation is made in a slightly different setting on the group $\mathbb{Z}^2$.

for free groups, Baumslag-Solitar groups, the Heisenberg group, the Thompson groups F, T and V, and the general linear groups $GL(n,\mathbb{Z})$. In fact, infinite finitely generated groups without a copy of $\mathbb{Z}$, known as torsion groups, are quite rare and hard to construct. Nevertheless, many examples exist in the literature. The question is particularly hard in the case that the torsion is bounded, that is, there exists $n \in \mathbb{N}$ such that every element of the group generates a subgroup of order at most n. See [1] for a discussion of groups with bounded torsion. In the case of unbounded torsion, there are examples that are relatively simple to define, and simple to prove torsion. We mention in particular [5,6].

Given that such groups exist, an obvious question is whether we can extend Theorem 1 to this case. It turns out that we cannot: in Theorem 2 we show that a subshift on a torsion group accepted by a multi-headed automaton 'cannot be too sparse', and as a further result we obtain Theorem 6, which characterizes the torsion groups as those on which multi-headed automata are strictly weaker than Turing machines.

2 Definitions and examples

2.1 Subshifts

In this section, we define some basic notions of of symbolic dynamics and computability. Some references on symbolic dynamics on general groups are [3,2], and a standard reference on $\mathbb{Z}$ is [8].

Let G be a group with identity element $1_G \in G$. Our groups are always infinite (the finite case being trivial) and finitely generated (since the notions we consider are local). For convenience, if G is finitely generated, we fix a symmetric finite set $s(G) \subset G$ of generators for it. The set $s(G)^*$ consists of all finite words over $s(G)$, and for $v, w \in s(G)^*$, we denote $v \sim w$ and $v \sim g$ if the words correspond to the same element $g \in G$. We denote by $B_G(n)$ the ball of radius n with respect to the fixed set of generators: $B_G(n) = \{g \in G \mid w \in s(G)^*, |w| \leq n, w \sim g\}$.

A *torsion element* of a group G is an element $g \in G$ that satisfies $g^n = 1_G$ for some $n \geq 1$. If all elements of G are torsions, then G is a *torsion group*. To each torsion element $g \in G$ we associate its *order* $t_G(g) = \min\{n \geq 1 \mid g^n = 1_G\}$, and to each finitely generated torsion group we associate the *torsion function* $T_G : \mathbb{N} \to \mathbb{N}$, defined by $T_G(n) = \max\{t_G(g) \mid g \in B_G(n)\}$. A non-torsion group, conversely, is one containing an isomorphic copy of $\mathbb{Z}$.

Both *alphabet* and *state set* mean any finite set. The symbol S always means an alphabet, and the set S^G is the *full G-shift over S*. Its elements, usually denoted by x, y, z, are called *configurations*. We define both a left and a right action of G on S^G, called the *left and right shifts*. The left action is given by $(g \cdot x)_h = x_{g^{-1}h}$. It is indeed an action because

$$(g_2 \cdot (g_1 \cdot x))_h = (g_1 \cdot x)_{g_2^{-1}h} = x_{g_1^{-1}g_2^{-1}h} = x_{(g_2 g_1)^{-1}h} = (g_2 g_1 \cdot x)_h.$$

The right action is given by $\sigma_g^R(x)_h = x_{hg}$. It is indeed an action because

$$\sigma_{g_2}^R(\sigma_{g_1}^R(x))_h = \sigma_{g_1}^R(x)_{hg_2} = x_{hg_2g_1} = \sigma_{g_2g_1}^R(x)_h.$$

We give S^G the product topology induced by the discrete topology on S, making it a compact metrizable space. It is easy to show that both actions are continuous in this topology. A *subshift* of S^G is a topologically closed subset of S^G closed under the left action of G. A *cellular automaton* on a subshift $X \subset S^G$ is a continuous map $f : X \to X$ that commutes with the left shifts in the sense that $g \cdot f(x) = f(g \cdot x)$ holds for all $x \in S^G$ and $g \in G$. We denote by $\mathrm{Aut}(X)$ the group of bijective cellular automata on X under composition.

For us, the main importance of the left action is that it allows for nice definitions of subshifts and cellular automata. On the other hand, when the right action of an element $g \in G$ is well-defined on a subshift, it is a cellular automaton. In particular, the right actions show that $\mathrm{Aut}(S^G)$ contains a copy of the group G, by the injective group homomorphism $g \mapsto \sigma_g^R$.

A *pattern (on G)* is a function $P \in S^D$, where $D = D(P)$ is a finite subset of G, called the *domain* of P. Each pattern P defines a *cylinder set* $[P] = \{x \in S^G \mid x|_D = P\}$. The clopen (topologically closed and open) sets in S^G are precisely the finite unions of cylinders, and form a basis for the topology. Subshifts can be characterized as sets $X \subset S^G$ for which there exists a set of *forbidden patterns* $\mathcal{F}$ such that

$$X = \{x \in S^G \mid \forall P \in \mathcal{F} : \forall g \in G : g \cdot x \notin [P]\}.$$

Each cellular automaton on X has a *radius* $r \in \mathbb{N}$ and a *local rule* $F : S^{B_G(r)} \to S$ such that $f(x)_g = F(g^{-1} \cdot x|_{B_G(r)})$ holds for all $x \in X$ and $g \in G$.

Example 1. Let G be the free group generated by $g, h \in G$, and $X \subset S^G$ the set

$$\{x \in S^G \mid \forall g \in G : \forall n \in \mathbb{Z} : x_{gh^n} = x_g\}.$$

We show that X is a subshift, and for that, let $x \in X$ and $g \in G$. We need to show $g \cdot x \in X$. Given $f \in G$ and $n \in \mathbb{Z}$, we have

$$(f \cdot x)_{gh^n} = x_{f^{-1}gh^n} = x_{f^{-1}g} = (f \cdot x)_g$$

by the definition of the left action, and the fact $x \in X$.

Definition 1. *If $S \ni 0$ is a finite alphabet, then the* one-S subshift *on a group G is the subshift $X_S^G \subset S^G$ where a finite pattern $P \in S^D$ is forbidden if and only if there exist $d \neq e \in D$ with $P_e \neq 0 \neq P_d$. If $0 \notin S$, we write $X_S^G = X_{S \cup \{0\}}^G$.*

The group G is usually clear from context, and we write X_S for X_S^G.

Definition 2. *Let $S \ni 0$ be a finite alphabet. A configuration $x \in S^G$ is* k-sparse *if it satisfies $|\{g \in G \mid x_g \neq 0\}| \leq k$. A subshift is* k-sparse *if each of its configurations is k-sparse, and* sparse *if it is k-sparse for some $k \in \mathbb{N}$.*

The one-S subshift X_S is of course a 1-sparse subshift on any group. Note that in a sparse subshift, there is a global bound on the number of nonzero

symbols. The *sum* $x+y$ of sparse configurations $x, y \in S^G$ with disjoint support (no $g \in G$ satisfies $x_g \neq 0$ and $y_g \neq 0$) is defined by

$$(x+y)_g = \begin{cases} x_g \text{ if } x_g \neq 0, \\ y_g \text{ otherwise.} \end{cases}$$

A finite pattern is represented computationally as a finite list of word-symbol pairs $(w, d) \in s(G)^* \times S$. Such a list is *inconsistent* if it contains two pairs (v, d) and (w, e) with $v \sim w$ and $d \neq e$ (in this case, it does not actually encode a pattern), and otherwise *consistent*.

Definition 3. *The* word problem of G *is the set* $E = \{w \in s(G)^* \mid w \sim 1_G\}$ *of words that represent the identity element of* G*. Whether the word problem is decidable is independent of the chosen generator set. We say* G *is* recursively presented *if* $G \cong \langle g_1, \ldots, g_k \mid w_1, w_2, \ldots \rangle$*, where* $(w_i)_{i \in \mathbb{N}}$ *is a computable sequence of relations.*[4] *This is equivalent to the set* E *being recursively enumerable.*

If G *has a decidable word problem, we say that a subshift* $X \subset S^G$ *is* Π_1^0 *if there exists a Turing machine that enumerates a list of consistent forbidden patterns defining it.*

A subshift X is Π_1^0 if and only if there exists an oracle Turing machine that, given an oracle for a configuration $x \in S^G$ (which returns the symbol $x_w \in S$ for a given word $w \in s(G)^*$), eventually halts if and only if $x \notin X$.

2.2 Automata

We now define group-walking automata and the subshifts they recognize. Here and henceforth, by π_i we mean the projection to the ith coordinate of a finite Cartesian product.

Definition 4. *A* k-headed group-walking automaton *on the full shift* S^G *is a tuple* $A = (\prod_{i=1}^k Q_i, f, I, F)$*, where* $Q_1, Q_2, \ldots, Q_k$ *are state sets not containing the symbol* 0*,* I *and* F *are finite clopen subsets of the product subshift* $Y = \prod_{i=1}^k X_{Q_i}$*, and* $f : S^G \times Y \to S^G \times Y$ *is a CA satisfying* $\pi_1 \circ f = \pi_1$*.*

We denote by $\mathcal{S}(G, k)$ *the class of subshifts* $X \subset S^G$ *for which there exists a* k*-headed automaton* A *as above such that*

$$X = \{x \in S^G \mid \forall g, h \in G, y \in I, n \in \mathbb{N} : h \cdot \pi_2(f^n(g \cdot x, y)) \notin F\}.$$

We also write $\mathcal{S}(G) = \bigcup_{k \geq 1} \mathcal{S}(G, k)$*.*

The intuition for these definitions is the following. A configuration $y \in Y = \prod_{i=1}^k X_{Q_i}$ consists of k layers $\pi_i(y)$, each of which contains at most one nonzero symbol $q_i \in Q_i$, representing the i'th head of the automaton in state q_i. The cellular automaton f is the update function of the heads: since f has a finite

[4] The term 'recursively presented' comes from the fact that one may always assume $\{w_i \mid i \in \mathbb{N}\}$ to be a recursive set of words.

radius, the heads can only move at a bounded speed, and interact over bounded distances. Also, the condition $\pi_1 \circ f = \pi_1$ ensures that the automaton cannot alter the configuration of S^G that it runs on. The clopen sets $I, F \subset Y$ are the *initial and final states* of the automaton. Each of them is a finite union of cylinder sets $[P]$, and since they are also finite as sets, each of the patterns P necessarily contains all k heads of the automaton. Thus, an initial or finite state specifies the position and internal state for each head, and we translate them by every element of G in the definition of $\mathcal{S}(G)$.

The definition is given in dynamical terms to make the connection with cellular automata clearer, and to facilitate the statement and proof of Lemma 3. With some work, one can show that this model is equivalent to the one we gave in [9] in terms of the classes of subshifts defined.

Example 2. Let G be again the free group generated by the elements $g, h \in G$, and let $S = \{0, 1\}$. We define a two-headed group-walking automaton $A = (Q_1 \times Q_2, f, I, F)$ on G as follows. The local state sets are $Q_1 = \{q_g, q_{g^{-1}}\}$ and $Q_2 = \{q_h, q_{h^{-1}}\}$, the set of initial states I contains only the cylinder set $\{x \in (Q_1 \times Q_2)^G \mid x_{1_G} = (q_g, q_h)\}$, and the set of final states F contains the cylinder $\{x \in (Q_1 \times Q_2)^G \mid x_{1_G} = (q_{g^{-1}}, q_{h^{-1}})\}$. This means that the heads of the automaton are initialized at the same coordinate in states q_g and q_h, and a configuration is rejected if they ever return to the same coordinate in states $q_{g^{-1}}$ and $q_{h^{-1}}$. The CA f moves each head by the step indicated in its state, and if a head encounters a symbol 1 in state q_g or q_h, it assumes the respective inverse state $q_{g^{-1}}$ or $q_{h^{-1}}$.

In a run of the automaton, the heads start moving in the directions g and h until they encounter symbols 1, and then turn back. If both of them turn at the same time, they will meet again where they started, in the states $q_{g^{-1}}$ and $q_{h^{-1}}$, so the configuration is rejected. If not, the configuration is not rejected. Thus the automaton A defines the subshift $X \subset S^G$ with the forbidden patterns

$$\{1_G \mapsto 0, g \mapsto 0, h \mapsto 0, \dots, g^{n-1} \mapsto 0, h^{n-1} \mapsto 0, g^n \mapsto 1, h^n \mapsto 1\}$$

for all $n \geq 1$. It is not an SFT.

Naturally, Turing machines are stronger than multi-headed finite automata.

Lemma 1. *If G has a decidable word problem and $X \in \mathcal{S}(G)$, then $X \in \Pi_1^0$.*

Proof. Let A be a group-walking automaton that defines X. We construct a Turing machine T_A that outputs its forbidden patterns. The machine T_A enumerates all consistent patterns over G (using the fact that G has a decidable word problem), and simulates a run of the automaton A on each of them, from every initial state. If one of the heads exits the pattern during such a simulation, or every head enters an infinite loop, that simulation is simply discarded. If one of the runs enters a rejecting state on the pattern P before exiting it (from any initial configuration and initial position on the domain $D(P)$), the machine T_A outputs the pattern P. It is clear that T_A defines the same subshift as A. □

3 Non-torsion groups with a decidable word problem

On non-torsion groups, there are essentially no restrictions on the types of computation a multi-headed finite state automaton can do, apart from the inherent limits of computation. In fact, we will implement all Π_1^0-subshifts on such groups, using just three heads. The construction is similar to that in [4] and [9].

Theorem 1. *If G is finitely generated, infinite and non-torsion, and has a decidable word problem, then $\mathcal{S}(G, 3)$ is exactly the class of Π_1^0-subshifts.*

Proof. By Lemma 1, all $\mathcal{S}(G, 3)$-subshifts are Π_1^0. To show that $\mathcal{S}(G, 3)$ contains all Π_1^0-subshifts, we repeat the proof of Theorem 5 in [9], where the same problem was considered for $G = \mathbb{Z}^d$, with one additional detail in the non-abelian case. Since there are not many changes, we refer to [9] for some of the details.

Let $X \subset S^G$ be a Π_1^0-subshift, and let $h \in G$ be an element of infinite order. Given a Turing machine T enumerating a list of forbidden patterns for X, we construct an automaton A_T with three heads, the *pointer head*, the *zig-zag head* and the *counter head*. The relative positions of these heads store a number, which we increment, decrement, multiply and divide by suitable constants, and test for equivalence and divisibility by constants, in order to perform arbitrary computation: such a model is Turing-complete by the results of [10].

More precisely, all heads are initialized on the same element of G, which we may assume to be 1_G. The run of the automaton proceeds in *sweeps*, each of which either corresponds to an arithmetical operation as described above, or moves the heads in some direction. Between these sweeps, the location of both the pointer head and the zig-zag head is some $g \in G$, and the position of the counter head is gh^p. The number $p \in \mathbb{N}$ is the *counter value*. Changes in the counter value are used to perform computation, and changes to the value g allow us to read the contents of every cell in the configuration.

The operations are implemented as in the case $G = Z^d$ (for example, see Proposition 3 in [9]). The only operations that are nontrivial to implement are multiplication and division, and they are dealt with by standard signaling techniques. The details of this are omitted in [9], so we outline the construction here: we explain how to multiply the counter value by a rational number $0 < \frac{m}{n} < 1$ assuming the counter value is divisible by n; to multiply by a rational number greater than 1, one essentially performs the same steps in reverse.

For this, let $g \in G$ be the position of the pointer head. The idea is that the zig-zag head moves to the counter head, which is at gh^p, along the progression $g, gh, gh^2, \ldots$. The two heads then perform a coordinated move along the path $g, gh, gh^2, \ldots, gh^c$, so that they meet exactly at $gh^{\frac{m}{n}p}$. The zig-zag head then returns to the pointer head, and computation continues. We have much freedom in performing these moves, but we fix a particular scheme that works: After the zig-zag head and the counter head meet, the counter head starts moving in steps of h towards the pointer head (so that from the cell gh^j, it moves to the cell gh^{j-1} in one step), until it meets the zig-zag head again. The zig-zag head moves towards the pointer head by h^n every step, until it meets the pointer

head. Note that n divides p, so that the zig-zag head indeed reaches exactly the cell g. After this, the zig-zag head starts moving back towards the counter head at speed $\frac{m}{n-m-1}$. More precisely, the zig-zag head carries a modular counter, starting at 0, and at each step it increments this counter. When the modular counter reaches $n-m-1$, the zig-zag head resets it to 0 and moves by h^m. When the zig-zag head reaches the counter head, it turns back, and returns to the pointer head. It is a simple calculation to check that the heads meet exactly at $gh^{\frac{m}{n}p}$, as required, so the counter value has been changed correctly.

Now that we can do arbitrary computation in the counter value, we give the algorithm we simulate in it. The algorithm is the same as in the proof of Theorem 5 of in [9], and we reproduce it in Algorithm 1 with trivial modifications. In the algorithm, objects related to the group are stored as they are output by the Turing machine: group elements are finite words over $s(G)$, and patterns $P \in S^D$ are lists of pairs $(w, s) \in s(G)^* \times S$ meaning $P_w = s$. We assume the Turing machine T outputs an infinite list of forbidden patterns, and enters the state q_{out} every time it outputs a new pattern.

Algorithm 1 The algorithm that the three-headed automaton A_T simulates.

$c \leftarrow c_0$ ▷ A configuration of T, set to the initial configuration
$u \leftarrow 1_G \in G$ ▷ The position of the pointer head relative to the initial position
$P : \emptyset \to S$ ▷ A finite pattern at the initial position
loop
  repeat
    $c \leftarrow \textsc{NextConf}_T(c)$ ▷ Simulate one step of T
  until $\textsc{State}(c) = q_{\text{out}}$ ▷ T outputs something
  $P' \leftarrow \textsc{OutputOf}(c)$ ▷ A forbidden pattern
  while $D(P') \not\subset D(P)$ **do**
    $w \leftarrow \textsc{LexMin}(D(P) \setminus D(P'))$ ▷ The lexicographically minimal element
    for $a = u_1^{-1}, u_2^{-1}, \ldots, u_{|u|}^{-1}, w_1, w_2, \ldots, w_{|w|}$ **do**
      $\textsc{MoveBy}(a)$ ▷ Move all heads of A_T by group element a
    $u \leftarrow w$ ▷ New position of the pointer is w
    $b \leftarrow \textsc{ReadSymbol}$ ▷ Read the symbol of x under the pointer head
    $P \leftarrow P \cup \{u \mapsto b\}$ ▷ Expand P by one coordinate
  if $P|_{D(P')} = P'$ **then halt** ▷ The forbidden pattern P' was found

The function ReadSymbol gives the symbol currently under the pointer head. The procedure MoveBy(a) causes the three heads to assume new positions: if the pointer head and zig-zag head are at g and the counter head is at gh^p, they are moved to ga and gah^p, respectively. This step is the main difference between the abelian and non-abelian cases, and we explain it below. We note that there are only finitely many different messages sent between the abstract computation and A_T, namely the exchange related to ReadSymbol and the commands MoveBy(a) for finitely many $a \in G$. This information exchange can

easily be performed by storing the state of the Turing machine T directly in the finite state of the pointer head.

It is easy to see that this algorithm does what we want: whenever the Turing machine T enumerates a forbidden pattern P', we expand the stored pattern P by reading the configuration until its domain contains that of P'. If P' occurs in the configuration, it is eventually found by the algorithm from some starting position, and conversely, if the automaton halts, this is because it found a forbidden pattern.

To finish the proof, we explain how to perform MoveBy(a). If G is abelian, this can be done as in [9]: the zig-zag head moves to the counter head, informs it of the element of G by which it should move, and returns back. The counter head moves as instructed, and the pointer head does so as well. If the pointer head was previously at g and the counter head at gh^P, and both move by $a \in G$, then after this sequence of moves, the pointer head will be at ga, and the counter head at $gh^P a = gah^P$, as required. More generally, this works if h is in the center of G. Otherwise, we may have $gh^P a \neq gah^P$. Since we do not necessarily have $gah^P \in g\langle h \rangle a$, the counter head may not even encode a valid counter value.

However, using the same trick we used to perform multiplications, we can perform the movement in general. First, the zig-zag head moves to the counter head. Then, both heads start moving toward the pointer head. The counter head moves in steps of h^{-1}, computing the parity of p on the way, and the zig-zag head moves in steps of h^{-2}. If p is even, then the zig-zag head reaches the anchor head exactly, moves to ga, and starts moving along the sequence $ga, gah, gah^2, \ldots$ in steps of h. If p is odd, then the zig-zag head reaches the cell gh^{-1} instead, moves to gah, and starts moving in steps of h as before. The counter head performs the same task, but with the speeds reversed: after reaching the anchor head with speed h^{-1}, it starts moving from ga in steps of h^2 if p was even, and from gah in steps of h^2 if p was odd. When the counter head reaches the pointer head, the pointer head also moves to ga. It is easy to check that the counter head and the zig-zag head meet at the cell gah^p. The counter head stops, and the zig-zag head returns to the pointer head. □

4 Walking on torsion groups

A torsion group is one where every element generates a finite subgroup. In this section, we show that on such groups, non-trivial sparse subshifts cannot be recognized by multi-headed automata. We also show two results about cellular automata and automorphism groups of sparse subshifts on torsion groups. These follow from a curious property, Lemma 3, of CA on sparse subshifts on torsion groups. In its proof, we use the following lemma about finite metric spaces.

Lemma 2. *Let X be a finite metric space with $|X| = k \geq 2$. For all $c < \mathrm{diam}(X)/(k-1)$, there exists a nontrivial partition $X = Y \cup Z$ with $d(Y, Z) > c$.*

Proof. For a set $E \subset X$, write $B_E(r)$ for the closed ball of radius $r \geq 0$ around E. Let $\mathrm{diam}(X) = d(y, z)$ for some $y, z \in X$. Let $X_1 = \{y\}$, and inductively

define $X_{i+1} = B_{X_i}(c)$. For all $i \geq 1$ we have either $|X_{i+1}| > |X_i|$ or $X_{i+1} = X_i$, and in the latter case we have $X_j = X_i$ for all $j \geq i$. It follows that $X_i = X_{i+1}$ holds for some $i \leq k$.

If we have $X_i = X_{i+1} = X$, then $\operatorname{diam}(X) \leq (k-1)c$, since every element of X, including z, is in the ball $B_y((i-1)c) \subset B_y((k-1)c)$. This is a contradiction, so it must be the case that $X_i = X_{i+1} \neq X$. Then $Y = X_i$ and $Z = X \setminus X_i$ give the desired partition. □

Lemma 3. *For all torsion groups G, there exists a function $d : \mathbb{N}^3 \to \mathbb{N}$ with the following property. For all k-sparse subshifts $X \subset S^G$ over all alphabets $S \ni 0$ with $|S| = q + 1$, all cellular automata $f : X \to X$ with radius $r \in \mathbb{N}$, and all $x \in X$, we have*

$$(\exists n \in \mathbb{N} : f^n(x)_g \neq 0) \implies \exists h \in B_G(d(k, q, r)) : x_{gh} \neq 0.$$

Proof. We prove the existence of such a function d by induction. We define the function so that it is monotone in all the three parameters. Let t_G be the order function and T_G the torsion function of G.

Case 1: $k = 1$

First, let $k = 1$, and let $f : X \to X$ be a CA. It is easy to show that if $x_{gh} = 0$ for all $h \in B_G(r)$, then $f(x)_g = 0$. Intuitively, this means that nonzero symbols can 'spread' by at most r per time step, and one cannot appear from nowhere. Since X is a k-sparse subshift and $k = 1$, every point $x \in X$ contains at most one nonzero coordinate $x_g \neq 0$. Intuitively, we want to give an upper bound on how far the nonzero symbol can travel from its initial position g.

By shift-commutation, it is enough to analyze the case $x_{1_G} \neq 0$. Combining the previous observations and the fact $|S| = q+1$, it follows from the pigeonhole principle that $f^{n+m}(x) = \sigma_h^R(f^n(x))$ for some $0 \leq n < n + m \leq q + 1$ and $h \in B_G((q+1)r)$. Since f commutes with the shift, we have $f^{n+\ell m}(x) = \sigma_{h^k}^R(f^n(x))$ for all $\ell \in \mathbb{N}$. Since $h^{t_G(h)} = 1_G$, we have $f^{n+t_G(h)m}(x) = f^n(x)$. We have shown that $f^j(x)_{h'} \neq 0$ for some $j \in \mathbb{N}$ implies $h' \in B_G((q+1)r(1 + t_G(h)))$. Since $h \in B_G((q+1)r)$, we can define

$$d(1, q, r) = (q + 1)r(1 + T_G((q + 1)r)).$$

Next, consider the case $k > 1$. To each configuration $x \in X$, we associate the metric space $A(x)$ whose points are the nonzero coordinates of x, and whose distances are those induced by the natural (right) distance in G. We will split the analysis of the dynamics of f on the point x into two cases, depending on whether the diameter of $A(f^n(x))$ stays bounded (by an explicit constant) as n grows.

Intuitively, the idea is that as long as the diameter stays small, we can shrink all the information in x into a single symbol, reducing to the case $d(1, \cdot, \cdot)$, and if the configuration starts expanding, then it splits into two pieces that can never again communicate, and we apply induction to these smaller pieces.

More precisely, define $A(x) = (\{g \in G \mid x_g \neq 0\}, \delta)$ where

$$\delta(g,h) = \min\{\ell \in \mathbb{N} \mid \exists w \in s(G)^\ell : h = gw\}.$$

Define also $c = 2d(k-1,q,r) + r$, and note that since d is monotone, we in particular have

$$c \geq \max_{1 \leq \ell < k} d(\ell,q,r) + d(k-\ell,q,r) + r.$$

We say that a configuration $x \in X$ is *clustered* if $\operatorname{diam}(A(x)) \leq (k-1)c$ holds, and *scattered* otherwise.

Case 2: clustered configurations

First, suppose $x \in X$ and $N \in \mathbb{N}$ are such that $f^n(x)$ is clustered for all $n \leq N$. We will give an upper bound on how far nonzero symbols can travel from their original positions in these N steps. Let $Z \subset X$ be the subshift generated by the configurations $f^n(x)$ for $n \leq N$. It is easy to see that every configuration of Z is clustered. Note that that the subshift Z may not be closed under f.

Let $Y = X_{\{0\} \cup K}$, where $K \subset \mathcal{L}_{B_G((k-1)c)}(Z)$ is the set of patterns P of shape $B_G((k-1)c)$ occurring in Z such that $P_{1_G} \neq 0$. Clearly, Y is a 1-sparse subshift, and it should be thought of as a 'compressed' version of Z, where all the nonzero symbols have been encoded into a single coordinate. The idea is to simulate CA f on the compressed subshift Y, and reduce back to the $k = 1$ case. Let $\phi : Y \to S^G$ be the 'decompression function' defined by

$$\phi(y)_h = \begin{cases} (y_g)_{g^{-1}h}, & \text{if } \exists g \in G : y_g \neq 0 \wedge g^{-1}h \in B_G((k-1)c), \\ 0, & \text{otherwise.} \end{cases}$$

Let $Y' = \phi^{-1}(Z)$, so that $\phi : Y' \to Z$ is a surjective block map.[5] A visualization of ϕ is shown in Figure 1.

Claim. There exists a (not necessarily unique) cellular automaton $f_\phi : Y' \to Y'$ such that $\phi(f_\phi(y)) = f(\phi(y))$ holds for all $y \in Y'$ such that $f(\phi(y)) \in Z$.

Intuitively, the CA f_ϕ simulates f on the compressed configurations of Y', as long as their ϕ-images are clustered.

Proof (of claim). Observe that for each $z \in Z$ and $g \in G$ there is at most one configuration $y \in Y'$ such that $y_g \neq 0$ and $\phi(y) = z$. Let then $1 = h_1 < h_2 < h_3 < \cdots$ be any total order on the group G, not necessarily in any way compatible with its algebraic structure. Then we can define a map f_ϕ with the desired properties as follows.

First, for the all-0 configuration $0^G \in Y'$, we define $f_\phi(0^G) = 0^G$, and for all $y \in Y'$ such that $f(\phi(y)) \notin Z$, we also define $f'(y) = 0^G$. For all other $y \in Y'$, let $g \in G$ be the unique element with $y_g \neq 0$, and let $W \subset Y'$ be the set of configurations $y' \in Y'$ with $\phi(y') = f(\phi(y))$. The set W is nonempty since

[5] The fact that G is torsion prevents us, in general, from defining a bijective version of ϕ. Also, the subshift Y' may be strictly smaller than Y.

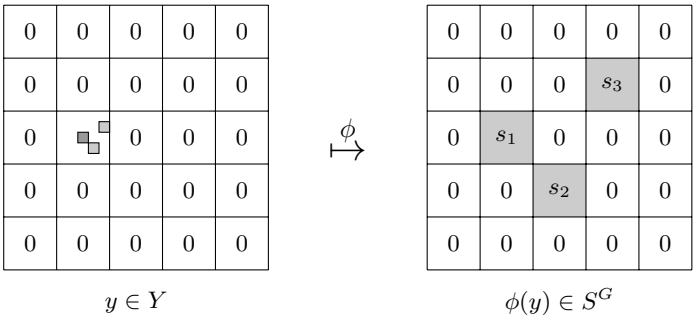

Fig. 1. The decompression function ϕ applied to a configuration $y \in Y$. We have chosen $G = \mathbb{Z}^2$ here for simplicity, even though it is not a torsion group. Note that the alphabet of Y consists of certain patterns of X and the symbol 0.

$\phi : Y' \to Z$ is surjective, and it is finite because the unique nonzero coordinate of each $y' \in W$ is among the coordinates gh where $h \in B_G((k-1)c + r)$, since we assumed $P_{1_G} \neq 0$ for each $P \in K$. Now, we choose $f_\phi(y)$ to be the unique configuration $y' \in W$ with $y'_{gh} \neq 0$, where $h \in G$ is minimal in the ordering $h_1 < h_2 < \cdots$. It is easy to check that f_ϕ is then continuous and shift-commuting. In fact, from the way we defined it, we see that its radius is at most $(k-1)c + r$. □

Recall the clustered configuration $x \in S^G$. We have $x \in Z$ by the definition of Z, so there exists a configuration $y \in Y'$ such that $\phi(y) = x$. By the above claim, we have $\phi(f_\phi^n(y)) = f^n(x)$ for all $n \leq N$. Since Y is a 1-sparse subshift with alphabet of size $|K|+1$ and f_ϕ is a CA on it with radius at most $(k-1)c+r$, we have

$$(\exists n : f_\phi^n(y)_g \neq 0) \implies \exists h \in B_G(d(1, |K|, (k-1)c + r)) : y_{gh} \neq 0 \qquad (1)$$

by Case 1 of this proof. We also remark that if we have $N > |K|$, then the configuration $f^n(x)$ is clustered for all $n \in \mathbb{N}$, since there exist $i < j \leq N$ such that $f_\phi^i(\phi(y))$ is a translated version of $f_\phi^j(\phi(y))$.

It remains to prove a variant of the above formula for f, and for that, let $f^n(x)_g \neq 0$ for some $g \in G$. Since the block map ϕ has radius $(k-1)c$, we have $\phi(f^n(x))_{gh'} = f_\phi^n(y)_{gh'} \neq 0$ for some $h' \in B_G((k-1)c)$. Equation (1) implies that $y_{gh'h} \neq 0$ for some $h \in B_G(d(1, |K|, (k-1)c + r))$, and from the definition of ϕ it follows that $x_{gh'h} \neq 0$ as well, since $(y_{gh'h})_{1_G} \neq 0$. We have shown that if $f^n(x)$ contains a nonzero symbol in some coordinate, then there is a nonzero coordinate in x at distance at most $d(1, |K|, (k-1)c + r) + (k-1)c$. Note that the cardinality of K is at most exponential in $(k-1)c$.

Case 3: scattered configurations

Suppose finally that the configuration $f^n(x)$ is scattered for some $n \in \mathbb{N}$, which we assume to be minimal. By the remark at the end of Case 2, we have

$n \leq |K|$. We apply Lemma 2 to the metric space $A(f^n(x))$, and obtain a partition for it into sets $C, D \subset G$ with distance at least c.

Denote $y = f^n(x)$. We define a partition of the configuration y by $y = y_C + y_D$, where $(y_C)_g = y_g$ when $g \in C$ and $(y_C)_g = 0$ otherwise, and y_D is defined analogously. By the definition of c, we have $c \geq d(|C|, q, r) + d(|D|, q, r) + r$. It is then easy to see that $f^n(y) = f^n(y_C) + f^n(y_D)$ for all $n \in \mathbb{N}$. In particular, if we have $f^j(y)_g \neq 0$ for some $j \in \mathbb{N}$ and $g \in G$, then $y_{gh} \neq 0$ for some $h \in B_G(\max_{\ell<k} d(\ell, q, r)) \subset B_G(d(k-1, q, r))$ by the induction hypothesis. Since we have $n \leq |K|$ and the CA f has radius r, this implies that $x_{ghh'} \neq 0$ for some $h' \in B_G(r|K|)$, which implies $hh' \in B_G(r|K| + d(k-1, q, r))$.

Putting all three cases together, we can define the function d recursively by

$$d(k, q, r) = d(1, |K|, (k-1)c + r) + (k-1)c + r|K| + d(k-1, q, r)$$

for all $k > 1$. □

The bounds we give are not very strong, but at least one can check that if the torsion function T_G is primitive recursive, then so is the function d.

Theorem 2. *If G is finitely generated, infinite and torsion, and $X \subset S^G$ is sparse and nontrivial, then $X \notin \mathcal{S}(G)$.*

Proof. Let A be a group-walking automaton and Y its associated subshift, and let $X' = \{x + y \mid x, y \in X, \forall g \in G : x_g = 0 \vee y_g = 0\}$. Since $X' \times Y$ is sparse, Lemma 3 implies that any head of A can only travel a bounded distance on any configuration of $X' \times Y$. Then, for all $x \in X$ and all but finitely many $g \in G$, the configuration $x + (g \cdot x)$ is rejected by A if and only if x is. If the support of x is maximal, this configuration is not in X. Thus A does not define X. □

Lemma 3 also restricts the structure of the automorphism group of a sparse subshift on a torsion group.

Theorem 3. *If G is torsion and $X \subset S^G$ is sparse, then* $\mathrm{Aut}(X)$ *is also torsion.*

The last theorem has an obvious converse: if G is not torsion, then the shift along a copy of $\mathbb{Z}$ is a non-torsion element of $\mathrm{Aut}(X)$ whenever X is sparse and nontrivial. One can construct such examples even in the quotient group $\mathrm{Aut}(G)/\langle \sigma_g^R \mid g \in G \rangle$.

5 Undecidable word problem

If the word problem for G is not necessarily decidable, one can give multiple definitions of Π_1^0. We give two, both of which correspond to our previous definition of Π_1^0 when the word problem is decidable. Recall that finite patterns are represented computationally as lists of pairs drawn from $s(G)^* \times S$.

Definition 5. *A subshift on G is Π_1^0 if there exists a Turing machine enumerating a set of (possibly inconsistent) forbidden lists of word-symbol pairs for it.*

Definition 6. *A subshift X on G is* intrinsically Π_1^0 *if there exists an oracle Turing machine that, given an oracle for the word problem of G, enumerates a set of consistent forbidden lists of word-symbol pairs for X.*

In [2], what we call intrinsically Π_1^0 is called G-effective, and this notion was first defined and studied there. Its actual definition in [2] uses 'group-walking Turing machines', but it is also shown to be equivalent to Definition 6. The following results, the first of which is a direct corollary of Theorem 1, relate these classes of subshifts to the hierarchy of group-walking automata.

Theorem 4. *If G is finitely generated, infinite and non-torsion, then $\mathcal{S}(G, 3)$ contains the class of Π_1^0-subshifts.*

Theorem 5. *If G is finitely generated, infinite and non-torsion, then $\mathcal{S}(G, 4)$ is exactly the class of subshifts on G which are intrinsically Π_1^0.*

Proof. Clearly, all $\mathcal{S}(G, 4)$ subshifts are intrinsically Π_1^0, since a Turing machine with an oracle for the word problem of G can simulate a multi-headed finite state machine on the group. The proof that $\mathcal{S}(G, 4)$ contains the intrinsically Π_1^0 subshifts is similar to that of Theorem 1, except that we must simulate a Turing machine with access to an oracle for G. Thus, we only need to describe how one can use four heads to check whether the identity $1 \sim w$ holds for an arbitrary $w \in s(G)^*$. For this, we use three heads to move by the letters of w, and leave the fourth head as a marker in the cell we started from. We return back on top of the fourth head if and only if $1 \sim w$. We can then move back by w^{-1} and pick up the fourth head. □

From these results, we obtain a characterization of torsion groups.

Lemma 4. *The X_S subshift is intrinsically Π_1^0 on every group.*

Theorem 6. *Let G be a finitely generated infinite group. Then G is torsion if and only if $\mathcal{S}(G, 4)$ is not equal to the class of all intrinsically Π_1^0 subshifts.*

Proof. This follows from Lemma 4, Theorem 5 and Theorem 2. □

Finally, we note that Lemma 4 requires the intrinsic notion of computability, as shown by the following corollary of [2, Proposition 2.3] (also proved in [7]).

Proposition 1. *Let G be a recursively presented and finitely generated group, and S is a nontrivial finite alphabet. The subshift X_S^G is Π_1^0 if and only if G has a decidable word problem.*

6 Future work and open questions

While we need four heads in the *proof* of Theorem 5, we are not able to separate the class $\mathcal{S}(G, 3)$ from $\mathcal{S}(G, 4)$ on any group G. We do have a general construction which separates these classes on all sufficiently complex torsion groups. Unfortunately, we do not know how to construct a group with the necessary properties, as the construction of torsion groups is quite complicated. Nevertheless, this leads us to believe that the classes are not always equal.

Conjecture 1. There exists an infinite finitely generated torsion group G such that $\mathcal{S}(G,3) \subsetneq \mathcal{S}(G,4)$. In particular, $\mathcal{S}(G,3)$ is not always equal to the class of intrinsically Π_1^0 subshifts.

We know that if G is not a torsion group, then the hierarchy $\mathcal{S}(G,k)_{k\geq 1}$ collapses to the fourth level (if not earlier), and $\mathcal{S}(G,4)$ is exactly the class of intrinsically Π_1^0 subshifts. On torsion groups, the hierarchy never reaches all intrinsically Π_1^0 subshifts, but we have not shown that it is infinite. We believe we have a general construction that proves exactly this, but it is relatively complicated, so for now we only state its conclusion as a conjecture.

Conjecture 2. If G is an infinite finitely generated torsion group, then the hierarchy $\mathcal{S}(G,k)_{k\geq 1}$ is infinite.

Some very basic questions about the abelian cases were left open in [9]. We have no progress on these questions.

Question 1. Do we have $\mathcal{S}(\mathbb{Z},2) = \mathcal{S}(\mathbb{Z},3)$ or $\mathcal{S}(\mathbb{Z}^2,2) = \mathcal{S}(\mathbb{Z}^2,3)$?

We note that in [4], a slightly different model of multi-headed group-walking automaton is studied on the group $\mathbb{Z}^2$, and it is shown that in this model, two-headed machines are strictly weaker than three-headed ones. It seems that the question is harder in our model. In [9], we only showed that $\mathcal{S}(\mathbb{Z}^d,2) \subsetneq \mathcal{S}(\mathbb{Z}^d,3)$ holds for $d \geq 3$.

References

1. S.I. Adian. The burnside problem on periodic groups and related questions. *Proceedings of the Steklov Institute of Mathematics*, 272(2):2–12, 2011.
2. N. Aubrun, S. Barbieri, and M. Sablik. A notion of effectiveness for subshifts on finitely generated groups. *ArXiv e-prints*, December 2014.
3. T. Ceccherini-Silberstein and M. Coornaert. *Cellular Automata and Groups*. Springer Monographs in Mathematics. Springer-Verlag, 2010.
4. Marianne Delorme and Jacques Mazoyer. Pebble automata. figures families recognition and universality. *Fundam. Inf.*, 52(1-3):81–132, January 2002.
5. R. I. Grigorčuk. On Burnside's problem on periodic groups. *Funktsional. Anal. i Prilozhen.*, 14(1):53–54, 1980.
6. Narain Gupta and Said Sidki. On the burnside problem for periodic groups. *Mathematische Zeitschrift*, 182(3):385–388, 1983.
7. E. Jeandel. Some Notes about Subshifts on Groups. *ArXiv e-prints*, January 2015.
8. Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, Cambridge, 1995.
9. Ville Salo and Ilkka Törmä. Plane-walking automata. *CoRR*, abs/1408.6701, 2014.
10. Rich Schroeppel. A two counter machine cannot calculate 2^N. 1972.