Newcastle University

School of Mathematics and Statistics

MMath Dissertation

Symbolic Dynamics

Author: Christopher Barrett
Supervisor: Dr. Evgenios Kakariadis

Abstract

First we introduce the topic of Symbolic dynamics and the main objects and morphisms of study - shift spaces and sliding block codes respectively. We see an appropriate metric that can be put on these spaces, and that the interpretation of these objects under the induced topology is neatly consistent with our raw definitions. Specifically, it turns out we are studying exactly the continuous and shift commuting functions between compact and shift invariant spaces. Next we inspect the link between shift spaces and graphs. Shifts of finite type are exactly conjugate to those which can be represented by an unlabeled graph, while sofic shifts are a superset of these that can be represented by a labeled graph. We proceed to give the details of a specific graphical representation of sofic shifts - follower set graphs - along with an algorithm (and implementation) for finding such graphs for shifts of finite type. For a subset of sofic shifts we also show that there exists a certain unique minimal representation, which can be found as a subgraph of these follower set graphs. Finally we present a result showing that unlabeled graph isomorphism of follower set graphs implies an unlabeled isomorphism of these minimal representations, when they exist.

Acknowledgments

I would like to thank my project supervisor, Evgenios Kakariadis, who has been incredibly supportive and dedicated throughout this project. Without him, I never would have discovered this subject - nor would I have had half as much fun working on it.

Contents

1 Introduction

Symbolic dynamics was initially motivated by the study of dynamical systems. One can discretize time in such a system, so that the state is observed only at regular intervals of time rather than continuously. A further simplification is to also discretize the space by dividing it into a finite number of disjoint areas. Taking a point in space, we can approximate its orbit - where it is mapped to in the space - by recording which area it is in at each discrete time. By associating each area with a letter, and each time with an integer, we can represent this orbit using a bi-infinite sequence of letters drawn from a finite alphabet. The study of such sequences is what we focus our attention on.

Shift invariance is a natural consequence of modeling dynamical systems, since a point and its position one interval further in time correspond exactly to a sequence and its image under the shift map. Our definition of shift spaces is restricted to only those that can be generated by a set of "forbidden words" - that is, the shift space would be the set of bi-infinite sequences which do not contain any such word. Clearly this definition reflects such shift invariance, which in turn is reflected in the naming of these objects.

The techniques and ideas from this subject have found significant applications in data storage and transmission. For example, when storing audio data on a compact disc, we often use sequences of 1's separated by at least two, but no more than ten, 0's. There are physical reasons for this. Along one groove of a CD, binary data is read based on how the track is magnetized. If two 1's are too close together, the magnetic forces representing the 1's are at risk of canceling out and the signal is harder to detect. However, two successive 1's are read as two pulses separated by some length of time. The length of time is used to calculate the number of 0's in between. But the clocking circuit used to measure the time interval may drift ever so slightly - which is not a problem at small time intervals, but adds up over larger ones. Hence, the clocking circuit is reset after every detection of a 1 - and we wish not to leave too long between successive 1's. When studying methods of encoding in such a way, we are actually asking questions about mappings from the space of arbitrary sequences to the space of sequences constrained in the above manner (named the run length limited shift). This is Symbolic Dynamics [6].

In Section 2, we will give the basic definitions of shift spaces, along with a characterization using allowable words instead of forbidden words. We then discuss the appropriate morphisms between these objects. A metric is given under which these objects and morphisms are proven to be exactly those satisfying and preserving certain fundamental properties.

Section 3 introduces the first specific type of shift space we will study - shifts of finite type. These are shift spaces which can be generated by a finite set of forbidden words, and they are fundamentally the same as those which can be represented on an unlabeled graph.

In Section 4, we introduce sofic shifts, a superset of shifts of finite type [8]. We define these to be exactly the shift spaces which can be represented on a labeled graph, but prove that they are in fact the closure of shifts of finite type under the appropriate surjective morphisms. Labeled graphical representations are non-unique, however for sofic shifts which can be represented on an irreducible graph, we show that there exists a certain unique minimal representation [4]. We also give a specific well defined presentation, the follower set graph, and show the minimal graph, when it exists, can be found as its subgraph.

We give a brief introduction to a measure of complexity called entropy, which proves to be a useful (in)variant of shift spaces. This is then used to prove a central result, originally drawn from our related paper [2]. The proof shows that unlabeled isomorphism of the follower set graphs implies unlabeled isomorphism of the unique minimal representations, when they exist.

In Section 5, we present an algorithm which calculates the follower set graph given a finite set of forbidden words. This is then easily extended to calculate the unique minimal presentation, again when it exists. In the appendices, we give a concrete implementation written in the programming language Haskell. This code was used in the aforementioned related paper, and a working version of the program can be found in the references [1]. It extends the core algorithm and uses it to search for distinct shifts of finite type which generate follower set graphs which are isomorphic when considered without labels.

2 Fundamentals of Shift Spaces

2.1 Shift Spaces

An alphabet $\mathcal{A}$ consists of a finite set of discrete symbols such as $\{0, 1, \ldots, 9\}$ or $\{a, b, \ldots, z\}$. The fundamental elements of study in Symbolic Dynamics are bi-infinite sequences of these "letters", drawn from a given alphabet. Such a sequence is denoted by $x = (x_i)_{i\in\mathbb{Z}}$, or by

$$x = \ldots x_{-2}x_{-1}.x_0x_1x_2\ldots,$$

where each $x_i \in \mathcal{A}$.

Definition 2.1 (Full Shift)**.** The *full $\mathcal{A}$-shift* is the set of all bi-infinite sequences of symbols from $\mathcal{A}$, written as

$$\mathcal{A}^{\mathbb{Z}} = \{x = (x_i)_{i\in\mathbb{Z}} : x_i \in \mathcal{A},\ \forall i \in \mathbb{Z}\}.$$

Note that $\mathcal{A}^{\mathbb{Z}}$ is the standard mathematical notation for the set of all functions from $\mathbb{Z}$ to $\mathcal{A}$ - functions which can be viewed as bi-infinite sequences with elements drawn from $\mathcal{A}$. The *full r-shift* is the full shift over the alphabet $\{0, 1, \ldots, r-1\}$. Sequences drawn from the full 2-shift are called *binary sequences.*

A *word* or *block* is a finite sequence of letters. Words of length n are commonly referred to as n-blocks or n-words. For words u and v, writing uv simply denotes concatenation. Given a word w, writing $|w|$ refers to the number of letters in the word. The same notation used on a set denotes the number of elements contained in it. Writing w^n for $n \in \mathbb{N}$ means w is repeated n time s. The empty word is counted as a word of 0 length, denotes $\varnothing$.

Given some $x \in \mathcal{A}^{\mathbb{Z}}$, we refer to the block of symbols from coordinates i to j as

$$x_{[i,j]} = x_ix_{i+1}\ldots x_j.$$

To write $w \in x$ means that there exists some $i, j \in \mathbb{Z}$ such that $x_{[i,j]} = w$.

Definition 2.2 (Shift Space)**.** Let $\mathcal{F}$ be a (possibly infinite) set of "forbidden words" over $\mathcal{A}$. Then the *shift space $X_{\mathcal{F}}$* is defined to be the set of all sequences in $\mathcal{A}^{\mathbb{Z}}$ that do *not* contain any word in $\mathcal{F}$.

Example 2.3. The set $\mathcal{A}^{\mathbb{Z}}$ is a shift space. Indeed if we take $\mathcal{F} = \emptyset$, the empty set, then no words are forbidden and so we allow all possible sequences over $\mathcal{A}$.

Remark 2.4. It is possible to have multiple sets of $\mathcal{F}$ generate the same shift space. For example, working with binary sequences, we have that $\mathcal{F}_1 = \{100\}$ generates the same shift space as $\mathcal{F}_2 = \{1001, 1000\}$.

Example 2.5. The intersection of finite shift spaces over the same alphabet is itself a shift space. Take $x \in X_{\mathcal{F}_1} \cap X_{\mathcal{F}_2}$. Then x contains no $w \in \mathcal{F}_1$ and x contains no $w \in \mathcal{F}_2$. This is equivalent to saying x contains no $w \in \mathcal{F}_1 \cup \mathcal{F}_2$, thus we have $x \in X_{\mathcal{F}_1} \cap X_{\mathcal{F}_2}$ if and only if $x \in X_{\mathcal{F}_1 \cup \mathcal{F}_2}$ i.e. $X_{\mathcal{F}_1} \cap X_{\mathcal{F}_2} = X_{\mathcal{F}_1 \cup \mathcal{F}_2}$. Using induction we can extend this to countable intersections.

Lemma 2.6. *Given $\mathcal{F}_1 \subseteq \mathcal{F}_2$, we have that $X_{\mathcal{F}_1} \supseteq X_{\mathcal{F}_2}$.*

Proof. Take an element $x \in X_{\mathcal{F}_2}$. Then x does not contain any word $w \in \mathcal{F}_2$, and thus x does not contain any word $w \in \mathcal{F}_1$. Thus $x \in X_{\mathcal{F}_1}$, giving the desired containment.

From this, we can see the relationship between $X_{\mathcal{F}_1 \cap \mathcal{F}_2}$ and $X_{\mathcal{F}_1} \cup X_{\mathcal{F}_2}$. Clearly, $\mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{F}_1$ and $\mathcal{F}_1 \cap \mathcal{F}_2 \subseteq \mathcal{F}_2$. Using the above, this gives us that $X_{\mathcal{F}_1 \cap \mathcal{F}_2} \supseteq X_{\mathcal{F}_1}$ and $X_{\mathcal{F}_1 \cap \mathcal{F}_2} \supseteq X_{\mathcal{F}_2}$ respectively. Together, we get $X_{\mathcal{F}_1 \cap \mathcal{F}_2} \supseteq X_{\mathcal{F}_1} \cup X_{\mathcal{F}_2}$. ∎

Example 2.7. Fix a nonempty, finite subset $S \subseteq N = \{0, 1, 2, \dots\}$. Then we can define a shift space $X = X(S)$ to be the set of all binary sequences of the form

$$x = \dots 10^{n_{-1}} 10^{n_0} 10^{n_1} \dots.$$

with $n_i \in S \; \forall i \in \mathbb{Z}$. In particular, there is an infinite number of 1's in either direction. We can see that for $\mathcal{F} = \{10^n1 : n \in N \setminus S\} \cup \{0^{n+1} : n = \max_{s \in S}(s)\}$, we get that $X = X_{\mathcal{F}}$.

For an infinite subset $S \subseteq N$, we must relax the condition for infinite 1's in each direction - and allow an infinite string of 0's before and after. This is because, by the above description of the points in $X = X(S)$, we can show that $0^\infty \in X$. Assume $0^\infty \notin X$. Then there exists a word $w \in 0^\infty$ such that $w \in \mathcal{F}$. This means w must be of the form 0^n, for some n. With infinite S, for every $n \in S$, there exists some $m \in S$ such that $m > n$, and so we have points in X containing words of the form 10^m1; thus they are allowed. But we also have that forbidden word w is contained in 10^m1 and so we have reached a contradiction. It follows that $0^\infty \in X$ when S infinite. The forbidden set in this case is simply $\mathcal{F} = \{10^n1 : n \in N \setminus S\}$.

These shift spaces are known as the *S-gap shifts*. Special cases include the (d, k) *run-length limited shift* with $S = \{d, d+1, \dots, k\}$, the *even shift* with $S = \{0, 2, 4, \dots\}$.

Definition 2.8 (Shift Map)**.** The *shift map* σ on the full shift $\mathcal{A}^{\mathbb{Z}}$ maps a point x to the point $y = \sigma(x)$ such that $y_i = x_{i+1} \; \forall i \in \mathbb{Z}$. That is

$$\text{if } x = \dots x_{-2}x_{-1}.x_0x_1x_2 \dots \text{ then } y = \dots x_{-1}x_0.x_1x_2x_3 \dots.$$

Thus every letter is shifted one place to the left. The inverse shift map σ^{-1} shifts every letter to the right. We can write σ^n to denote the composition of n copies of the function.

We say a map $\phi : X \to Y$ is *shift commuting* if $\sigma_Y \circ \phi(x) = \phi \circ \sigma_X(x)$ for every point $x \in X$. Every shift space X is *shift invariant*, that is that $\sigma(X) = X$. This is clear from the observation that the constraints on shift spaces are given purely in terms of forbidden words, with no reference to the coordinates.

2.2 Languages

Definition 2.9 (Language)**.** Let $X \subseteq \mathcal{A}^{\mathbb{Z}}$, and let $\mathcal{B}_n(X)$ denote the set of all n-words that occur in any of the sequences inside X. The *language* of X is the set

$$\mathcal{B}(X) = \bigcup_{n=0}^{\infty} \mathcal{B}_n(X),$$

that is, all possible "allowed" words in all points of X.

Not every set of words is the language of a shift space. In what follows, consider the complement of a set of blocks over $\mathcal{A}$ as being taken relative to the set of all possible words over $\mathcal{A}$, or the language of the full $\mathcal{A}$-shift. The third condition gives what is often a convenient way to determine when two shift spaces are the same.

Proposition 2.10.

1. *Let X be a shift space and $\mathcal{L} = \mathcal{B}(X)$ be its language. If $w \in \mathcal{L}$, then*
   - *(a) every subblock of w belongs to $\mathcal{L}$, and*
   - *(b) there are nonempty blocks u and v such that $uwv \in \mathcal{L}$.*
2. *A collection of blocks $\mathcal{L}$ is the language of a shift space if and only if $\mathcal{L}$ satisfies the above condition.*
3. *The language of a shift space determines the shift space. In fact, for any shift space, $X = X_{\mathcal{B}(X)^c}$. Thus two shift spaces are equal if and only if they have the same language.*

Proof. (1) If $w \in \mathcal{L} = \mathcal{B}(X)$, then w occurs in some point x in X. Every subblock of w then also occurs in x, and so is in $\mathcal{L}$. There also exist nonempty blocks $u, v \in x$ such that $uwv \in x$, thus there exist $u, v \in \mathcal{L}$ and $uwv \in \mathcal{L}$

(2) The implication is given by condition 1. For the converse, let $\mathcal{L}$ be a collection of blocks satisfying 1. We must show this is the language of some shift space. The shift space will be $X_{\mathcal{L}^c}$ and so we must show that $\mathcal{L} = \mathcal{B}(X_{\mathcal{L}^c})$. If $w \in \mathcal{B}(X_{\mathcal{L}^c})$, then w occurs at some point of $\mathcal{B}(X_{\mathcal{L}^c})$, so w is not a forbidden word, i.e. w is not in $\mathcal{L}^c$. That is, $w \in \mathcal{L}$. Thus $\mathcal{B}(X_{\mathcal{L}^c}) \subseteq \mathcal{L}$.

Now we want to show $\mathcal{L} \subseteq \mathcal{B}(X_{\mathcal{L}^c})$. For $w \in \mathcal{L}$ we can repeatedly apply 1b to get an arbitrarily large block $x_{[i,j]}$ such that - by 1a - every subblock is in $\mathcal{L}$. Thus there exist no $n, m \in \mathbb{Z}$ such that $x_{[n,m]}$ contains a forbidden word, and so there exists a sequence $x = (x_i)_{i \in \mathbb{Z}} \in X_{\mathcal{L}^c}$. Since w occurs in x, we have that $w \in \mathcal{B}(X_{\mathcal{L}^c})$ and thus $\mathcal{L} \subseteq \mathcal{B}(X_{\mathcal{L}^c})$.

(3) Take some $x \in X$. The set $\mathcal{B}(X)$ contains all blocks occuring in all points of X and so also in x specifically. So no block occuring in x is in $\mathcal{B}(X)^c$ and hence $x \in X_{\mathcal{B}(X)^c}$, showing $X \subseteq X_{\mathcal{B}(X)^c}$.

For the reverse inclusion, note that since X is a shift space, there exists a set $\mathcal{F}$ such that $X = X_{\mathcal{F}}$. Note that $\mathcal{F} \subseteq \mathcal{B}(X)^c$ since $\mathcal{B}(X)^c$ is the set of all words which do not appear in X. Hence by Lemma 2.6, $X_{\mathcal{B}(X)^c} \subseteq X_{\mathcal{F}} = X$. ∎

The complement of the language $\mathcal{B}(X_{\mathcal{F}})$ gives the largest possible set of forbidden words for $X_{\mathcal{F}}$. The language of any subset $X \subseteq \mathcal{A}^{\mathbb{Z}}$ - whether X is a shift space or not - satisfies condition 1 of the above proposition, and so is in fact the language of *some* shift space containing X.

Corollary 2.11. *Let X be a subset of the full $\mathcal{A}$-shift. Then X is a shift space if and only if $x \in X$ exactly when $x \in \mathcal{A}^{\mathbb{Z}}$ and each $x_{[i,j]} \in \mathcal{B}(X)$, for all $i, j \in \mathbb{Z}$.*

Example 2.12. In general, the intersection of two languages is not a language. Over binary sequences, take the languages of $X_{\mathcal{F}_1}$ and $X_{\mathcal{F}_2}$, with $\mathcal{F}_1 = \{10^n1 : n \in \mathbb{N}\}$ and $\mathcal{F}_2 = \{0\}$. Note that $X_{\mathcal{F}_1}$ will be the set of all points containing either a single 1 or none at all. Then $\mathcal{L}_1 = \mathcal{B}(X_{\mathcal{F}_1})$ and $\mathcal{L}_2 = \mathcal{B}(X_{\mathcal{F}_2}) = \{1^n : n \in \mathbb{N}\}$. Their intersection $\mathcal{L}_1 \cap \mathcal{L}_2 = \{1\}$ is clearly not a language as it does not satisfy property 1(b) of the language proposition.

2.3 Sliding Block Codes

Definition 2.13 (Sliding block code)**.** Let X be a shift space over $\mathcal{A}_1$. Fix $n, m \in \mathbb{Z}$ with $-m \leq n$ and let $\Phi : \mathcal{B}_{m+n+1}(X) \to \mathcal{A}_2$ be a fixed block map from allowed $(m+n+1)$-blocks in X to symbols in $\mathcal{A}_2$. Then the map $\phi : X \to \mathcal{A}_2^{\mathbb{Z}}$ given by

$$\phi(x)_i = \Phi(x_{[i-m,i+n]})$$

is called a *sliding block code* with *memory* m and *anticipation* n. We can see that $\phi(x)_i$ depends on a "window" of coordinates around x_i and that we form y by "sliding" this window along one coordinate at a time. We say ϕ is *induced* by Φ, with the formation denoted by $\phi = \Phi_\infty^{[-m,n]}$ or simply $\phi = \Phi_\infty$ if m and n are understood. If Y is a shift space over $\mathcal{A}_2$ and $\phi(X) \subseteq Y$, we can write $\phi : X \to Y$.

It will be useful to note that we can extend the "window size" of Φ to be arbitrarily large. For $M \geq m$ and $N \geq n$ define $\hat{\Phi} : \mathcal{B}_{M+N+1}(X) \to \mathcal{A}_2$ by:

$$\hat{\Phi}(x_{[-M,N]}) = \Phi(x_{[-m,n]}).$$

Similarly it will prove useful to allow Φ to act on larger sections of a sequence at once. We can define a different $\hat{\Phi} : \mathcal{B}_{m+n+k}(X) \to \mathcal{B}_k(\mathcal{A}_X^{[N]})$ as follows:

$$\hat{\Phi}(x_{[-m,n+k-1]}) = \Phi(x_{[-m,n]})\Phi(x_{[-m+1,n+1]}) \dots \Phi(x_{[-m+k-1,n+k-1]}).$$

If our sliding block code ϕ is induced by a block map Φ with no memory and anticipation then it acts on a single letter at a time and we call it a *1-block code*. In general we will not always use the hat symbol to denote an alteration of the block map in such a way, but assume Φ can be implicitly altered when needed.

Definition 2.14 (Higher block shifts)**.** Let X be a shift space over the alphabet $\mathcal{A}$ and $\mathcal{A}_X^{[N]} = \mathcal{B}_N(X)$ be the collection of all allowed N-blocks in X. We can consider $\mathcal{A}_X^{[N]}$ to be an alphabet in its own right, and form the full shift $(\mathcal{A}_X^{[N]})^{\mathbb{Z}}$. Define the Nth *higher block code* $\beta_N : X \to (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$ by

$$(\beta_N(x))_{[i]} = x_{[i,i+N-1]}.$$

Thus β_N replaces the ith coordinate of x with the N-block of coordinates in x starting at position i. Imagine the symbols in $\mathcal{A}_X^{[N]}$ to be written vertically, read from bottom to top. Then the image of $x = (x_i)_{i \in \mathbb{Z}}$ under β_3 has the form

$$\beta_3(x) = \dots \begin{bmatrix} x_0 \\ x_{-1} \\ x_{-2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_0 \\ x_{-1} \end{bmatrix} . \begin{bmatrix} x_2 \\ x_1 \\ x_0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_2 \\ x_1 \end{bmatrix} \dots \in (\mathcal{A}_X^{[N]})^{\mathbb{Z}}.$$

Remark 2.15. In fact, higher block codes are examples of sliding block codes. Let X be a shift space over $\mathcal{A}$ and take $m = 0$, $n = N - 1$. It becomes clear that if we define $\Phi : \mathcal{B}_N(X) \to \mathcal{A}_X^{[N]}$ with

$$\Phi(a_0 a_1 \dots a_{N-1}) = a_0 a_1 \dots a_{N-1}$$

then the induced sliding block code $\Phi_\infty^{[0,N-1]} : X \to \mathcal{A}_X^{[N]}$ is the Nth higher block shift. Note that while this may look like the identity function, we are taking N-words from our domain and assigning them letters in our alphabet $\mathcal{A}_X^{[N]}$. It just so happens that the letters in the output correspond in a sensible way to our input.

Proposition 2.16. *Let X and Y be shift spaces. If $\phi : X \to Y$ is a sliding block code, then $\phi \circ \sigma_X = \sigma_Y \circ \phi$; i.e. the function is shift commuting.*

Proof. Let ϕ be induced by the block map $\Phi : \mathcal{B}_{m+n+1}(X) \to \mathcal{A}$ with memory m and anticipation n. Then for any $x \in X$,

$$(\sigma_Y \circ \phi)(x)_{[i]} = \phi(x)_{[i+1]} = \Phi(x_{[i+1-m,i+1+n]}),$$

while

$$(\phi \circ \sigma_X)(x)_{[i]} = \phi(\sigma_X(x))_{[i]} = \Phi(\sigma_X(x)_{[i-m,i+n]}) = \Phi(x_{[i+1-m,i+1+n]}).$$

We have shown the ith coordinates of the images agree for each i, so the images are equal. ∎

We proceed to give a full characterization of sliding block codes. This will become useful in the next section, when we come to give another characterization in terms of topological properties.

Proposition 2.17. *Let X and Y be shift spaces. A map $\phi : X \to Y$ is a sliding block code if and only if it is shift commuting and there exists $N \geq 0$ such that $\phi(x)_{[0]}$ is a function of $x_{[-N,N]}$, for every $x \in X$.*

Proof. Let ϕ be a sliding block code induced by block map Φ with memory m and anticipation n. From the previous proposition, we have that ϕ commutes with the shift map, and directly from the definition of such a code we can see that $\phi(x)_{[0]} = \Phi(x_{[-m,n]}) = \hat{\Phi}(x_{[-N,N]})$ if we expand the window size of Φ appropriately.

For the converse, we need to show there exists a block map Φ that induces the map $\phi : X \to Y$. By hypothesis, $\phi(x)_0 = f(x_{[-N,N]})$ for some $N \geq 0$ and some f. Set $\hat{\Phi}(x_{[-N,N]}) = f(x_{[-N,N]})$. This is clearly well defined, but it is necessary to show that Φ is defined for every $(2N+1)$-block allowed in x. Take some $w \in \mathcal{B}_{(2N+1)}(X)$. Then $w = x_{[i,i+2N]}$ for some $x \in X$ and some $i \in \mathbb{Z}$. Now see $y = \sigma^{(i+N)}(x) \in X$ also, due to shift invariance of shift spaces. We have $w = y_{[-N,N]}$ and so we have defined $\hat{\Phi}(w) = f(y_{[-N,N]})$.

Finally we must show that $\hat{\Phi}$ induces our map ϕ; that is, that $\phi(x)_i = \hat{\Phi}(x_{[i-N,i+N]})$. We use the shift commuting property of ϕ to see

$$\begin{aligned}\phi(x)_{[i]} &= (\sigma_Y^i \circ \phi(x))_{[0]} = (\phi \circ \sigma_X^i(x))_{[0]} \\ &= \hat{\phi}(\sigma_X^i(x))_{[0]} = \hat{\Phi}(\sigma_X^i(x)_{[-N,N]}) = \hat{\Phi}(x_{[i-N,i+N]}).\end{aligned}$$

as required. ∎

2.4 Shifts as Metric Spaces

Shift spaces admit a topology once we have applied a suitable metric to the space. The metric we will use in the following text captures the idea that points are closer together when they share a larger central block.

Definition 2.18 (The metric on $\mathcal{A}^{\mathbb{Z}}$)**.** For $x, y \in \mathcal{A}^{\mathbb{Z}}$, let k be the maximal integer such that $x_{[-k,k]} = y_{[-k,k]}$. Then

$$\rho(x,y) = \begin{cases} 2^{-k} & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases}$$

defines a metric. In other words, we find the largest k such that the central $(2k+1)$-blocks of x and y agree, and take 2^{-k} as the distance (with the convention that if $x_0 \neq y_0$ then $k = -1$). This allows us to see $\mathcal{A}^{\mathbb{Z}}$ as a metric space.

Remark 2.19. The function ρ is indeed a metric. It is trivial to show that $\rho(x,y) = 0$ if and only if $x = y$ and that $\rho(x,y) = \rho(y,x)$. It remains to show that the triangle inequality holds, i.e. $\rho(x,z) \leq \rho(x,y) + \rho(y,z)$.

If $\rho(x,y) = 2^{-n}$ then $x_{[-n,n]} = y_{[-n,n]}$ and similarly if $\rho(y,z) = 2^{-m}$ then $y_{[-m,m]} = z_{[-m,m]}$. Set $k = \min(\{n,m\})$. Then $x_{[-k,k]} = z_{[-k,k]}$ at least and so

$$\rho(x,y) \leq 2^{-k} \leq 2^{-n} + 2^{-m} = \rho(x,y) + \rho(y,z).$$

as required.

Definition 2.20 (Cylinder set)**.** Let X be a shift space and fix some word $u \in \mathcal{B}(X)$ and $k \in \mathbb{Z}$. Then the *cylinder set* is defined by

$$C_k(u) = \{x \in X : x_{[k,k+|u|-1]} = u\};$$

that is $C_k(u)$ is the set of sequences in X in which the word u occurs starting at position k.

Proposition 2.21. *Cylinder sets are open.*

Proof. Let $C_k(u)$ be a cylinder set for fixed integer k and fixed word u. Set $N = \max(\{|k|, |k + |u| - 1|\})$ and $\delta = 2^{-N}$. Then for every point $x \in C_k(u)$, we have that the open ball $B_\delta(x)$ is the set of all points which share at least their central $(2N+1)$-block with x. This shared central $(2N+1)$-block is constructed to contain the word u, and so $B_\delta(x) \subseteq C_k(u)$. ∎

Remark 2.22. Notice that for fixed integer k and fixed word $u = u_1 u_2 \dots u_n$, we have

$$C_k(u) = C_k(u_1) \cap C_{k+1}(u_2) \cap \cdots \cap C_{k+n-1}(u_n).$$

Proposition 2.23. *Cylinder sets are closed.*

Proof. Fix an integer k. First, observe that for a given letter i, the cylinder set

$$C_k(i)^c = \{x : x_k = i\}^c = \{x : x_k \neq i\} = \bigcup_{j \neq i} C_k(j),$$

is the union of open sets, and thus open.

Now we must show that $[C_k(u)]^c$ is open for a word $u = u_1 u_2 \dots u_n$. We directly verify that

$$\begin{aligned} [C_k(u)]^c &= [C_k(u_1) \cap C_{k+1}(u_2) \cap \cdots \cap C_{k+n-1}(u_n)]^c \\ &= C_k(u_1)^c \cup C_{k+1}(u_1)^c \cup \cdots \cup C_{k+n-1}(u_n)^c. \end{aligned}$$

Each u_i is a letter and so $[C_k(u)]^c$ is open as the union of open sets. ∎

Remark 2.24. The set $\mathcal{A}^{\mathbb{Z}}$ is totally disconnected. Assume there exists a non-trivial connected subset W. Then it contains two distinct points x and y. Points x and y must therefore differ in at least one co-ordinate, say i. We can thus separate W by the two open, non-empty and non-intersecting sets $C_i(x_i)$ which contains x and $[C_i(x_i)]^c$ which contains y.

Remark 2.25. It will be useful to observe that open balls are equivalent to certain 'centred' cylinder sets. See that for some given point $x \in X$ and a given $\epsilon > 0$, we have

$$B_\epsilon(x) = \{y \in X : d(x, y) < \epsilon\}.$$

In fact, for $2^{-n} \leq \epsilon < 2^{-(n-1)}$, with n an integer

$$B_\epsilon(x) = B_{2^{-n}}(x).$$

due to the discrete nature of the definition of our metric. Such an open ball contains all sequences which share a central $(2k+1)$-block with x. Thus

$$B_\epsilon(x) = C_{-k}(x_{[-k,k]})$$

As such, the centred cylinder sets form a (countable) neighbourhood basis for x:

$$\{C_k(x_{[-k,k]}) : k \in \mathbb{Z}\}.$$

Similarly, a base for the topology is formed from the set of all such cylinder sets:

$$\{C_k(x_{[-k,k]}) : k \in \mathbb{Z},\ x \in X\}.$$

In fact, we can also find a sub-base for $\mathcal{A}^{\mathbb{Z}}$. Indeed, consider the set:

$$\{C_k(i) : i \in \mathcal{A},\ k \in \mathbb{Z}\}.$$

Then finite intersections of these elements can generate any of the elements of our cylinder base, as illustrated by Remark 2.22. Since each cylinder set $C_k(i)$ can be regarded as an open ball around some element of X by using the property of shift invariance, we can see that this generates the same topology as the metric space.

Let us denote the projection map from bi-infinite sequences onto their symbol set as $\pi_k : \mathcal{A}^{\mathbb{Z}} \to \mathcal{A}$. This returns the kth element of a sequence. Then it is interesting to note that we can regard $C_k(i)$ as the inverse projection map π_k^{-1}. The product topology $\Pi_{i\in\mathbb{Z}} X_i$ has a sub-base consisting of sets that are the inverse projections of an open set in each X_i. If we take each X_i to be our symbol set $\mathcal{A}$ endowed with the discrete topology, we can see that $\mathcal{A}^{\mathbb{Z}}$ and $\Pi_{i\in\mathbb{Z}}\mathcal{A}$ are the same set sharing the same sub-base, and as such are equivalent topological spaces.

We proceed to give a topological characterization of shift spaces and of sliding block codes.

Definition 2.26 (Compactness in metric spaces)**.** In a metric space M, *compactness* can be characterized as follows: for every sequence $\{x^{(n)}\}$ in M, there exists a subsequence $\{x^{(k_n)}\}$ with $k_n > k_{n-1}$ and an x in M for which $x^{(k_n)} \to x$ as $n \to \infty$.

Theorem 2.27. *Shift spaces are compact.*

Proof. Let X be a shift space with the metric ρ and a sequence $\{x^{(n)}\}$ in X. We will find a subsequence converging in X. Since we have an infinite sequence $\{x^{(n)}\}$ and only finitely many symbols in $\mathcal{A}$, there must be an infinite set of integers S_0 such that $x^{(n)}_{[0]}$ are equal for all $n \in S_0$.

Similarly, there are only a finite number of 3-blocks over $\mathcal{A}$. And so, looking at the subset of points we already have from S_0, we can find another infinite set $S_1 \subseteq S_0$ such that $x^{(n)}_{[-1,1]}$ are equal for all $n \in S_1$. Continuing in the same way, for each $k \geq 1$ we can find an infinite set $S_k \subseteq S_{k-1}$ such that $x^{n}_{[-k,k]}$ are equal for all $n \in S_k$.

Define x to be the point with $x_{[-k,k]} = x^{(n)}_{[-k,k]}$ for some $n \in S_k$. This suffices as a definition because we can determine every coordinate in the sequence. It is also well defined, as for all $n \in S_k$, the words $x^{(n)}_{[-k,k]}$ are equal by construction, and for any two values of k, the two definitions of x will agree where defined.

We inductively define k_n as the smallest element of S_k which exceeds k_{n-1}. This guarantees increasing values of k_n and so we have a true subsequence of n. By this definition, $x^{(k_n)}$ will share a central $(2k+1)$-block with x, and so as $k \to \infty$, $x^{(k_n)} \to x$ under our metric.

It remains to show that x is a member of X. We can see that every subblock of x is taken from a point in X and so is in $\mathcal{B}(X)$. Then by Corollary 1.11, x is in X. ■

Corollary 2.28. *The full shift is compact.*

Remark 2.29. For the next theorem, recall that in a compact metric space M, a subset $E \subseteq M$ is comapct if and only if it is closed.

Theorem 2.30. *A subset of $\mathcal{A}^{\mathbb{Z}}$ is a shift space if and only if it is shift-invariant and compact.*

Proof. We have already seen that shift spaces are shift-invariant and compact. Take a subset $X \subseteq \mathcal{A}^{\mathbb{Z}}$ that is shift-invariant and compact. It is a compact subset of a metric space and therefore closed. Thus $\mathcal{A}^{\mathbb{Z}} \setminus X$ is open. Hence, for each $y \in \mathcal{A}^{\mathbb{Z}} \setminus X$ there is some open ball containing y inside this set. Remembering open balls are equivalent to certain cylinder sets, see that there exists some $k = k(y)$ such that the cylinder set $C_{-k}(y_{[-k,k]}) \subseteq \mathcal{A}^{\mathbb{Z}} \setminus X$. Take $\mathcal{F} = \{y_{[-k,k]} : y \in \mathcal{A}^{\mathbb{Z}} \setminus X\}$. It is clear that this forbids every sequence outside of X and so we have $X_{\mathcal{F}} \subseteq X$. It remains to show that $X \subseteq X_{\mathcal{F}}$. We will show the contrapositive. To this end, let $x \notin X_{\mathcal{F}}$. Then there is some $y_{[-k,k]} \in x$, which we can take to start at coordinate n. So $x_{[n,n+2k]} = y_{[-k,k]}$. Thus $\sigma^{n+k}(x)_{[-k,k]} = y_{[-k,k]}$ and so $\sigma^{n+k}(x) \notin X$. But by shift invariance we also have $x \notin X$. Thus $X = X_{\mathcal{F}}$ is a shift space. ■

Lemma 2.31. *Let E and F be compact, disjoint subspaces of a metric space M. Then there exists a $\delta > 0$ such that $\rho(x,y) \geq \delta$ whenever $x \in E$ and $y \in F$.*

Proof. Assume the statement is false. Then for every $\delta > 0$ there exist points $x \in E$ and $y \in F$ such that $\rho(x,y) < \delta$. Taking $\delta_n = \frac{1}{n}$ for $n \in \mathbb{N}$, construct sequences $\{x_n\} \subseteq E$ and $\{y_n\} \subseteq F$ of points such that $\rho(x_n, y_n) < \delta_n$. By compactness of E and F, we can find convergent subsequences $\{x_{n_k}\}$ and $\{y_{n_k}\}$ with $x_{n_k} \to x \in E$ and $y_{n_k} \to y \in F$ as $k \to \infty$. Clearly $\rho(x_{n_k}, y_{n_k}) < \delta_{n_k} \to 0$ and $\rho(x_{n_k}, y_{n_k}) \to \rho(x,y)$. Thus by the uniqueness of limits in metric spaces, $\rho(x,y) = 0$ and thus $x = y$. However x and y are in disjoint metric spaces and so this is a contradiction. ■

Theorem 2.32. *Let X and Y be shift spaces and a function $\theta : X \to Y$ to map between them. Then θ is a sliding block code if and only if it is continuous and shift commuting.*

Proof. We have already seen that sliding block codes are shift commuting. Here we show that they are also continuous. Let $\theta : X \to Y$ be a sliding block code, expanding the window so that it has memory and anticipation k. Let $x, x' \in X$ share a central $(2N+1)$-block. Their images under ϕ will agree so long as the window is over only the shared co-ordinates. For $i \in [-N+k, N-k]$, $\phi(x)_{[i]}$ and $\phi(x')_{[i]}$ will depend only on coordinates in the shared block and thus be equal. So their images agree on $2N+1-2k$ coordinates. Thus any points that share a central $(2M+1+2k)$-block will have images that share a central $(2M+1)$-block - for any $M \in \mathbb{N}$. This is equivalent to the definition of continuity in metric spaces - statements about shared central blocks correspond to statements about distance under our metric.

For the converse, we take a continuous and shift commuting $\theta : X \to Y$ and wish to show that it is a sliding block code. By Proposition 2.17 we have only have to show that, for every $x \in X$, $\theta(x)_0$ is a function of $x_{[-N,N]}$ for some $N \geq 0$. Let $\mathcal{A}_X$ be the alphabet of X and $\mathcal{A}_Y$ that of Y. For each $b \in \mathcal{A}_Y$ let $C_0(b)$ denote the cylinder set $\{y \in Y : y_0 = b\}$. The sets $C_0(b)$ are disjoint and closed, so their inverse images $E_b = \theta^{-1}(C_0(b))$ are disjoint and closed. Closed subsets of compact metric spaces are themselves compact, so each E_b is compact. Thus we can use lemma 2.31 to find a $\delta > 0$ such that points in different sets E_b are at least δ apart. Choose $n \in \mathbb{N}$ such that $2^{-n} < \delta$. Then any pair of points within 2^{-n} of each other share a central $(2n+1)$-block and so must lie in the same set E_b. Thus we have that $\theta(x)_0 = b = \theta(x')_0$ for every $x, x' \in X$ such that $x_{[-n,n]} = x'_{[-n,n]}$. Hence the central coordinate of $\theta(x)$ depends only on the central $(2n+1)$-block of x. ∎

Definition 2.33 (Factor map)**.** A surjective sliding block code from X to Y is called a *factor map*. We say that Y is a factor of X.

Definition 2.34 (Conjugacy)**.** A map between two shift spaces is a *conjugacy* if it is a sliding block code with a sliding block code inverse. Two shift spaces are *conjugate* if there exists a conjugacy between them.

Since sliding block codes are continuous, they preserve topological properties, and since they are shift commuting, they preserve the shift invariance which is core to shift spaces. So it seems that sliding block codes are indeed the appropriate *homomorphisms* here. Similarly, the definition of conjugacy here is equivalent to a shift commuting *homeomorphism* from topology.

Remark 2.35. The higher block shift $\beta_N(X) : X \to (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$ is a conjugacy, since it has a sliding block inverse given by

$$(\beta_N^{-1}(y))_{[i]} = f(y_i),$$

with $f : \mathcal{A}_X^{[N]} \to \mathcal{A}$ given by

$$f(a_0 a_1 \dots a_N) = a_0.$$

Remark 2.36. Let $\theta : X \to Y$ be a map between two compact metric spaces X and Y. Recall that if θ is continuous and bijective, θ has continuous inverse.

Theorem 2.37. *A map $\phi : X \to Y$ between two shift spaces is a conjugacy if and only if it is a bijective sliding block code.*

Proof. For the implication, note that ϕ has an inverse and so it is trivial to show it is bijective. For the converse, note that sliding block codes are continuous maps between compact metric spaces, and when they are bijective they have a (continuous) inverse. It only remains to be shown that ϕ^{-1} is shift-commuting. We can see that

$$\phi^{-1} \circ \sigma_Y(y) = \phi^{-1} \circ \sigma_Y \circ \phi \circ \phi^{-1}(y) = \phi^{-1} \circ \phi \circ \sigma_X \circ \phi^{-1}(y) = \sigma_X \circ \phi^{-1}(y),$$

as required. ∎

Theorem 2.38. *The image of a shift space X under a sliding block code ϕ is itself a shift space.*

Proof. We aim to show the image of X is compact and shift invariant. Continuous maps between metric spaces preserve the property of compactness, so the image of X under ϕ is compact. Sliding block codes are continuous and so preserve compactness - thus $\phi(X)$ is compact. To understand that shift invariance is preserved, see

$$\sigma_Y \circ \phi(X) = \phi \circ \sigma_X(X) = \phi(X),$$

by the shift commuting property of ϕ and the shift invariance of X. Thus $\phi(X)$ is also shift invariant - and hence is itself a shift space. ∎

Let $\phi : X \to Y$ be a sliding block code. Then it is possible to find a conjugate shift $\hat{X}$ such that the corresponding map $\hat{\phi} : \hat{X} \to Y$ is a 1-block code. Recoding ϕ in such a way can make it easier to reason about - however, the cost is that the alphabet of the domain becomes more complicated.

Proposition 2.39. *Let $\phi : X \to Y$ be a sliding block code with memory m and anticipation n. Then there exists a higher block shift $X^{[m+n+1]}$, conjugate to X by $\psi = \sigma^m \circ \beta_{m+n+1}$, and a 1-block code $\hat{\phi} : X^{[m+n+1]} \to Y$ such that $\hat{\phi} \circ \psi = \phi$. That is, the diagram below commutes.*

$$\begin{tikzcd} X \arrow[r, "\psi"] \arrow[dr, "\phi"'] & X^{[m+n+1]} \arrow[d, "\hat{\phi}"] \\ & Y \end{tikzcd}$$

Proof. We have already seen that the higher block shift is a conjugacy and since σ^m has the continuous inverse σ^{-m}. Thus ψ is the composition of two conjugacies and as such is itself a conjugacy. We can also see that

$$\psi(x)_i = \begin{bmatrix} x_{i+n} \\ \dots \\ x_{i-m} \end{bmatrix}$$

where the output is considered a single letter in the alphabet of $X^{[m+n+1]}$. Define

$$\hat{\Phi}(\begin{bmatrix} a_{m+n} \\ \dots \\ a_0 \end{bmatrix}) = \Phi(a_0 \dots a_{m+n})$$

noting that $\hat{\Phi}$ has no memory or anticipation and so induces a 1-block code. We will call this code $\hat{\phi}$ and check that it satisfies $\hat{\phi} \circ \psi = \phi$. To see this, note that:

$$(\hat{\phi} \circ \psi(x))_i = \hat{\phi}(\dots \begin{bmatrix} x_{n-1} \\ \dots \\ x_{-m-1} \end{bmatrix} . \begin{bmatrix} x_n \\ \dots \\ x_{-m} \end{bmatrix} \dots)_i = \hat{\Phi}(\begin{bmatrix} x_{i+n} \\ \dots \\ x_{i-m} \end{bmatrix}) = \Phi(x_{i-m} \dots x_{i+n}) = \phi(x)_i,$$

as required. ∎

3 Shifts of Finite Type

3.1 Basic Properties and Characterization

Shifts of finite type are those that can be described by a finite set of forbidden blocks. We have already encountered a number of these, for example the run-length limited shift or the full shift. Note that these shift spaces may in general also be described by an infinite set of forbidden blocks - in fact, the complement of the language is infinite for any proper subset of the full shift. We will later see that these shifts have a simple representation as a finite directed graph.

Definition 3.1 (Shift of finite type)**.** A shift space X is a *shift of finite type* if $X = X_{\mathcal{F}}$ for some finite set $\mathcal{F}$ of blocks.

Suppose we have a finite set of forbidden words $\mathcal{F}$ such that the longest word in the set is of length n. Then we can form the set $\mathcal{F}_n$ of all blocks of length n which contain some word in $\mathcal{F}$, and we will have $X_{\mathcal{F}} = X_{\mathcal{F}_n}$. This is because if a word is forbidden in $\mathcal{F}_n$, then it is forbidden in $\mathcal{F}$. On the other hand, if a word is forbidden in $\mathcal{F}$ then it will occur in a bi-infinite sequence inside some block of length n - which is a forbidden word in $\mathcal{F}_n$. For example, if $\mathcal{A} = \{0, 1\}$ and $\mathcal{F} = \{11, 000\}$, then $\mathcal{F}_3 = \{110, 111, 011, 000\}$. It will sometimes be convenient to assume this procedure has been carried out, and all forbidden words have the same length.

If we take $X_{\mathcal{F}}$ to be generated by such a set, with the words in $\mathcal{F}$ of length $M + 1$, then the sequence $x \in \mathcal{A}^{\mathbb{Z}}$ is in $X_{\mathcal{F}}$ exactly when $x_{[i,i+M]} \notin \mathcal{F}$ for every $i \in \mathbb{Z}$. This is because for x to contain a word in $\mathcal{F}$, and therefore be outside of $X_{\mathcal{F}}$, there must exist some $i \in \mathbb{Z}$ such that $x_{[i,i+M]} \in \mathcal{F}$. Thus to detect whether or not x is in $X_{\mathcal{F}}$, we need only scan the coordinates of x with a "window" of width N, and check if each block seen through this window is in the allowed collection $\mathcal{B}_N(X)$.

Definition 3.2 (M-step)**.** A shift of finite type is *M-step* (or has *memory M*) if it can be described by a collection of forbidden blocks, all of which have length $M + 1$.

Clearly if a shift is M-step, this also implies that it is of finite type. To motivate the above definition, see that to determine if a sequence is in an M-step shift, you must scan with a window size of $M + 1$, "remembering" the previous M coordinates. Note that an M-step shift is also K-step for all $K \geq M$. A 0-step shift is the full shift.

Example 3.3. The even shift, defined in Example 2.7, can be generated by the set of forbidden words $\mathcal{F} = \{10^m 1 : m \text{ odd}\}$. This is *not* a shift of finite type. To see this, suppose that it were - then there exists a finite set $\mathcal{F}'$ that generates the even shift, and so an integer $N \geq 1$ that is the length of the longest word of $\mathcal{F}'$. Then consider the point $x = 0^\infty 1 0^{2N+1} 1 0^\infty$. Every N-block of this sequence is in $\mathcal{B}_N(X_{\mathcal{F}})$ and hence in $\mathcal{B}_N(X_{\mathcal{F}'})$. Hence by the opening discussion, this should be enough to determine that x is in the even shift. But we can clearly see an odd number of 0s separate the two 1s and so we have reached a contradiction.

We proceed to give a characterization of shifts of finite type and show that the class is closed under conjugacy.

Theorem 3.4. *A shift space X is an M-step shift of finite type if and only if whenever $uv, vw \in \mathcal{B}(X)$ and $|v| \geq M$, then $uvw \in \mathcal{B}(X)$.*

Proof. First, suppose that X is M-step; meaning that $X = X_{\mathcal{F}}$ for finite set of blocks $\mathcal{F}$, with each block having length $M+1$. Then we take points $uv, vw \in \mathcal{B}(X)$ such that $|v| \geq M$ and aim to show $uvw \in \mathcal{B}(X)$. There must exist sequences $x, y \in X$ such that $uv \in x$ and $vw \in y$. Since X is shift invariant, we can take these sequences to be such that v lies in the same position in both; that is, $x_{[0,n]} = v = y_{[0,n]}$. We claim the point $z = x_{(-\infty,0)}vy_{(n,\infty)}$ is in X. Since X is M-step, if a forbidden word were to occur, it would occur in either $x_{(-\infty,0)}v$ or in $vy_{(n,\infty)}$, which would produce a contradiction. Thus $uvw \in z \in X$ and so $uvw \in \mathcal{B}(X)$.

For the converse, suppose that X is a shift space over $\mathcal{A}$ and M is an integer such that if $uv, vw \in \mathcal{B}(X)$ with $|v| \geq M$, then $uvw \in \mathcal{B}(X)$. We produce a candidate $\mathcal{F}$ which is the set of all $(M+1)$-blocks over $\mathcal{A}$ not in $\mathcal{B}_{M+1}(X)$. Showing $X = X_{\mathcal{F}}$ will prove X is an M-step shift of finite type.

If $x \in X$ then every $(M+1)$-block of x will be in $\mathcal{B}_{M+1}(X)$ and thus no words in $\mathcal{F}$ will occur in x - so x is also in $X_{\mathcal{F}}$. This shows $X \subseteq X_{\mathcal{F}}$ and so it remains to show $X_{\mathcal{F}} \subseteq X$. Again we take $x \in X_{\mathcal{F}}$ and aim to show it is in X. Such an x will have subblocks $x_{[0,M]}, x_{[1,M+1]} \in \mathcal{B}(X)$ since they are $(M+1)$ length blocks not in $\mathcal{F}$ - i.e. they *are* in $\mathcal{B}(X)$. Since they overlap in M letters, we can see that $x_{[0,M+1]} \in \mathcal{B}(X)$ too, by hypothesis (taking $v = x_{[1,M]}$). Now see that $x_{[2,M+2]}$ is also in $\mathcal{B}(X)$ and overlaps with $x_{[0,M+1]}$ in M letters. Thus $x_{[0,M+2]} \in \mathcal{B}(X)$ also. Repeated application of this argument in both directions - along with the basic properties of languages - shows that every subblock of x is in $\mathcal{B}(X)$, and thus $x \in X$ by Proposition 2.11. ∎

Theorem 3.5. *A shift space that is conjugate to a shift of finite type is itself a shift of finite type.*

Proof. Suppose that X and Y are shift spaces, conjugate via $\phi : X \to Y$ and with Y an M-step shift of finite type. Then we aim to show that X is an N-step shift of finite type - and we will do this using the above theorem. That is, we will take some arbitrary $uv, vw \in \mathcal{B}(X)$ such that $|v| \geq N$ and show that $uvw \in \mathcal{B}(X)$, thus proving X is N-step. Essentially, the aim is to map these uv, vw to Y where, if their images overlap enough, we can glue their images together. Then, we wish to apply the inverse map to this block to find the uvw we seek.

Take ϕ to be induced by Φ and its inverse by Ψ. The technical difficulty here is that the block maps Φ and Ψ produce shorter blocks each time they are applied. We can take each of Φ and Ψ to have both memory and anticipation equal to k, and so we will lose $2k$ symbols each time one of these maps is applied. Note that here we will allow their input to be longer than $2k+1$, in which case we use the 'extended' versions of the block maps. This is allowable as long as every $(2k+1)$-sub-block of the input is an allowed word.

Take $uv, vw \in \mathcal{B}(X)$. Then by Proposition 2.10, we can find $s, t \in \mathcal{B}_{2k}(X)$ such that $suv, vwt \in \mathcal{B}(X)$. Then observe that $\Phi(suv) = u'\Phi(v)$ and $\Phi(vwt) = \Phi(v)w'$ are in $\mathcal{B}(Y)$. This is simply because for example suv must exist in some sequence $x \in X$ and thus the sequence $\phi(x) \in Y$ must contain its image. If we have $|\Phi(v)| \geq M$, i.e. $|v| - 2k \geq M$, then these images overlap enough that we can use Theorem 3.4 to see that $u'\Phi(v)v' \in \mathcal{B}(Y)$. For $|v| \geq 2k$, we know that every $(2k+1)$-block of $suvwt$ is allowed in $\mathcal{B}(X)$ (although we do not yet know that the word is allowable as a whole). This tells us that our extended Φ can be applied to $suvwt$. In fact $u'\Phi(v)v'$ is nothing other than $\Phi(suvwt)$. Now we must carefully apply the inverse map. Observe that

$$\phi^{-1} \circ \phi(x)_i = x_i$$

which implies

$$\Psi(\phi(x)_{[i-k,i+k]}) = \Psi(\Phi(x_{[i-2k,i+2k]})) = x_i$$

and thus the composition of these two functions merely selects the central block - minus $2k$ symbols from either side. Since we started by choosing s, t to be of length $2k$, then we can see $\Psi(\Phi(suvwt)) = uvw$. We have already seen that $\Phi(suvwt) \in \mathcal{B}(Y)$ and thus this is a valid input for Ψ so long as its length is greater than $2k+1$. Thus the output uvw is allowable in X - so long as we take $|v| \geq M + 4k$. ∎

3.2 Graphical Representations

It turns out that many shift spaces admit a graphical representation. In this section, we introduce the different ways one can encode shift spaces into graphs, and how graphs can generate shift spaces. First, we begin with the basic definitions of graph theory:

Definition 3.6 (Graph)**.** A *graph* G consists of a finite set of *vertices* V, and a finite set of *edges* E. Each edge $e \in E$ has an initial and terminal vertex, denoted $i(e)$ and $t(e)$ respectively. We allow multiple edges between a given pair of vertices, and we allow *loops* - that is, edges which have the same initial and terminal vertex. A *subgraph* H of G is a graph given by a subset of vertices and edges of G, with the initial and terminal vertices of each edge defined as before. By the definition of a graph, we cannot inherit an edge from G without also inheriting its initial and terminal vertices from the vertex set.

Definition 3.7 (Path)**.** A path $\pi = e_1 e_2 \dots e_n$ on a graph G is a finite sequence of edges $e_i \in E$ such that each edge starts at the vertex the previous edge terminates on - that is, $t(e_i) = i(e_{i+1})$ for $1 \leq i \leq n-1$. We say the path π begins at $i(e_1)$ and terminates at $t(e_n)$. The *length* of the path is denoted $|\pi| = n$. A path that begins and ends at the same vertex is called a *cycle*.

Definition 3.8 (Graph homomorphism)**.** Let G and H be graphs. A *graph homomorphism* from G to H is a mapping which preserves initial and terminal states of the edges. It consists of a pair of maps $\Psi : V_G \to V_H$ and $\Phi : E_G \to E_H$ such that $i(\Phi(e)) = \Psi(i(e))$ and $t(\Phi(e)) = \Psi(t(e))$ for all edges $e \in E$. Thus if there is an edge e from v_1 to v_2 in G, then the edge $\Phi(e)$ will be from $\Psi(v_1)$ to $\Psi(v_2)$.

Naturally we say that graphs are *isomorphic* when there is a bijective homomorphism between them - that is, that we obtain H from G simply by relabelling vertices and edges, while preserving initial and terminal points.

Definition 3.9 (Adjacency Matrix)**.** Let G be a graph with vertex set V. For vertices $v, w \in V$, let A_{vw} denote the number of edges in G with initial vertex v and terminal vertex w. Then the *adjacency matrix* of G is $A = [A_{vw}]$. Given an adjacency matrix A, we can reconstruct the graph up to isomorphism - we denote such a reconstruction G_A.

It turns out that the ordering of the vertices in the matrix is hardly important - any selection of ordering will allow us to reconstruct the original graph up to isomorphism.

Proposition 3.10. *Let G be a graph with adjacency matrix A, and let $m \geq 0$.*

1. *The number of paths of length m from v to w is $(A^m)_{vw}$, the (v,w)-th entry of A^m.*

2. *The number of cycles of length m in G is given by the trace of A^m, i.e. the sum of the diagonal elements, and this number equals the number of points in X_G with period m (that is, sequences which repeat themselves every m symbols).*

Proof. (1) We can see this is true for $m = 0$ since the only paths of length 0 are the empty paths at each vertex. It is also trivially true for $m = 1$ since the adjacency matrix is defined using the number of paths between each vertex. We proceed by induction. Assume the statement is true for paths of length m, and we aim to prove for paths of length $m + 1$. Each path of length $m+1$ from v to w takes the form of a path of length m from v to k followed by a path of length 1 from k to v. To count all these paths we take the sum

$$\sum_{i=1}^{r} A^m_{vk_i} A_{k_i w}$$

which is exactly the matrix sum for the coordinate $(A^m A)_{vw} = A^{m+1}_{vw}$ as required. By induction, we achieve item (1).

(2) The first part follows from item (1) and the definition of a cycle. For the second part, note that if π is a cycle of length m in G, then π^∞ is a point of period m in X_G, while if $x \in X_G$ is a point of period m, then $x_{[1,m]}$ must be a cycle of length m in G. Thus there is a one-to-one correspondence between cycles of length m in G and points of period m in X_G. ∎

Definition 3.11 (Edge Shift)**.** Let G be a graph with the edge set E and adjacency matrix A. Then the *edge shift* X_G is the shift space over the alphabet $\mathcal{A} = E$ given by

$$X_G = X_A = \{(\xi_i)_{i\in\mathbb{Z}} \in E^{\mathbb{Z}} : t(\xi_i) = i(\xi_{i+1}) \text{ for all } i \in \mathbb{Z}\}.$$

That is, a bi-infinite sequence of edges is in X_G exactly when the terminal state of each edge is the initial state of the next - i.e. the sequence describes a bi-infinite *walk* on G. Note that a word in $\mathcal{B}(X_G)$ corresponds to a path in G. We now look at a similar construction based on the vertices of a graph.

Proposition 3.12. *For a graph G, the associated edge shift X_G is a 1-step shift of finite type.*

Proof. Let the alphabet $\mathcal{A}$ of X be the edge set E of G. Consider the finite collection

$$\mathcal{F} = \{ef : e, f \in \mathcal{A}, t(e) \neq i(f)\}$$

of 2-blocks over $\mathcal{A}$. We can see from the definition of an edge shift that a sequence $\xi \in \mathcal{A}^{\mathbb{Z}}$ is in X_G exactly when no block of $\mathcal{F}$ occurs in ξ. The proposition is then immediate. ∎

It may be the case that certain edges of a graph will never appear in an edge shift. Such a situation will arise when, for example, an edge goes to a vertex from which no edges leave. Such a vertex cannot be a part of a bi-infinite walk, and thus neither can its associated edges.

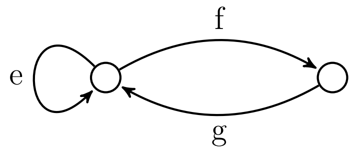


Figure 1: A shift of finite type generated by $\mathcal{F} = \{ee, eg, ff, fe, gg\}$ over $\mathcal{A} = \{e, f, g\}$.

Definition 3.13 (Essential)**.** We call a vertex v *stranded* when either no edges start at v or no edges end at v. A graph is *essential* if no vertex is stranded.

Proposition 3.14. *If G is a graph, then there is a unique subgraph H of G such that H is essential and $X_G = X_H$.*

Proof. The subgraph H is given by the set of edges $e \in E_G$ contained in some bi-infinite walk on G and the set of vertices visited on such walks. By definition, H is a subgraph and the set of bi-infinite walks on G and H are the same - thus $X_G = X_H$.

To see that H is unique, take H' to be another essential subgraph of G. Note that if we removed any vertex or edge from H to create H', we would be removing a bi-infinite walk in H and hence a sequence in X_H; thus $X_{H'} \neq X_H$. If we added any vertices to H from G, we would necessarily be adding stranded vertices. This is because we know they are not involved in any bi-infinite walks, which must be because at least one of them is stranded - else, every vertex would have at least one path coming into and leaving it, and thus a bi-infinite walk can be found including any given vertex. ∎

Definition 3.15 (Irreducibility in Graphs)**.** A graph G is *irreducible* if for every pair of vertices $v, w \in V$, there exists a path beginning at v and terminating at w.

Definition 3.16 (Irreducibility in Shift Spaces)**.** A shift space is said to be irreducible if for every pair $u, v \in \mathcal{B}(X)$ there exists a $w \in \mathcal{B}(X)$ such that $uwv \in \mathcal{B}(X)$.

Proposition 3.17. *An essential graph is irreducible if and only if its edge shift is irreducible.*

Proof. Let G be an irreducible graph, and $\pi, \tau \in \mathcal{B}(X_G)$. Recall that paths in G correspond to words in X_G. Suppose that π terminates at vertex v_1 and τ begins at vertex v_2. Then by the irreducibility of G, we can find a path ω from v_1 to v_2 and thus $\pi\omega\tau$ is a path in G and hence a word in $\mathcal{B}(X_G)$.

Conversely, suppose that G is essential and X_G is an irreducible shift. Since G is essential, every vertex has at least one path leaving and another path entering it. Thus for a given pair of vertices v_1, v_2, we can take edges f, g entering v_1 and leaving v_2 respectively to be paths of length 1. These paths are then words in $\mathcal{B}(X_G)$ and thus by the irreducibility of X_G, we can find a word $\omega \in \mathcal{B}(X_G)$ such that $f\omega g \in \mathcal{B}(X_G)$. Then ω corresponds to a path in G from v_1 to v_2 and thus G is irreducible. ∎

While it may seem that edge shifts are special cases of shifts of finite type, we will see that any such shift space can be recoded, using a higher block presentation, to an edge shift.

Theorem 3.18. *If X is an M-step shift of finite type, then there is a graph G such that $X^{[M+1]} = X_G$.*

Proof. First, we deal with the case $M = 0$. Here, X is the full shift and we simply define G to be as below - a single node with a loop corresponding to each letter of the alphabet.

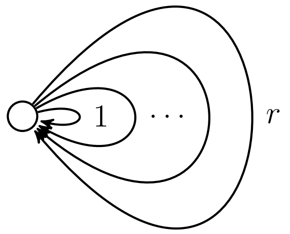


Figure 2: Graphical representation of $\{1, \ldots, r\}^{\mathbb{Z}}$.

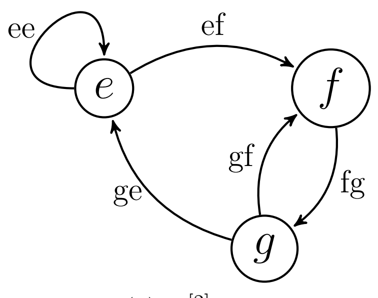
For the cases where $M \geq 1$, we define the vertex set to be $V = \mathcal{B}_M(X)$, the allowed M-blocks in X. And we define the edge set E as follows: suppose $v_1 = a_1a_2\dots a_M$ and $v_2 = b_1b_2\dots b_M$ are two vertices of G. If $a_2\dots a_M = b_1\dots b_{M-1}$ and if $a_1\dots a_Mb_M = a_1b_1\dots b_M$ is in $\mathcal{B}(X)$ then call this $(M+1)$-word w and draw exactly one edge from v_1 to v_2 labeled w. Otherwise, there is no edge from v_1 to v_2.

From this construction, it is clear that any progressively overlapping sequence of $(M+1)$-blocks can be found as a walk in G. Since $X^{[M+1]}$ is exactly the set of these sequences, we have that $X_G = X^{[M+1]}$. ∎

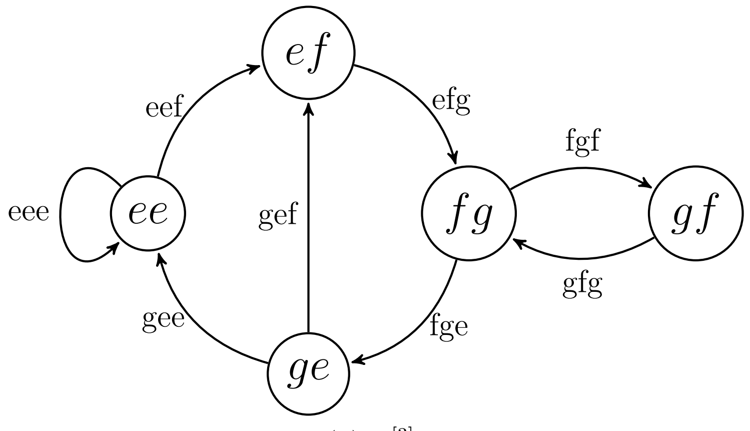
Definition 3.19 (Higher Edge Graph)**.** Let G be a graph. For $N = 1$, define $G^{[1]} = G$. For $N \geq 2$ we define the *higher edge graph* $G^{[N]}$ of G to have a vertex set equal to the collection of all paths of lengths $N-1$ in G, with the edge set as follows: suppose $v_1 = e_1e_2\dots e_{N-1}$ and $v_2 = f_1f_2\dots f_{N-1}$, are two vertices of G. Then draw exactly one edge from v_1 to v_2 whenever $e_2\dots e_{N-1} = f_1\dots f_{N-2}$, labeled as $e_1e_2\dots e_{N-1}f_{N-1} = e_1f_1\dots f_{N-1}$, and none otherwise.

Proposition 3.20. *Let G be a graph. Then $(X_G)^{[N]} = X_{G^{[N]}}$.*

Proof. The symbols for $(X_G)^{[N]}$ are the N-blocks from X_G, which are the paths of length N in G. But these are also the symbols for $X_{G^{[N]}}$. A bi-infinite sequence of these symbols is in either shift precisely when the N-blocks overlap progressively. ∎



(a) $G^{[2]}$



(b) $G^{[3]}$

Figure 3: Examples of higher edge graph, where G is taken to be the graph in Figure 1.

Definition 3.21 (Vertex Shift)**.** Let G be a graph such that from one vertex to another, there is at most one edge. Let A be its adjacency matrix. The *vertex shift* is the shift space $\hat{X}_G$ over the alphabet $\mathcal{A} = \{v_1, v_2 \dots v_n\}$, where v_i is a vertex in G, and is defined by

$$\hat{X}_G = \hat{X}_A = \{(v_i)_{i\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}} : A_{v_iv_{i+1}} = 1 \text{ for all } i \in \mathbb{Z}\}.$$

The following result establishes the relationship between edge shifts, vertex shifts and 1-step shifts of finite type.

Proposition 3.22.

1. *A shift of finite type is 1-step if and only if it is the vertex shift of some graph.*

2. If a shift space is the edge shift of some graph, then it is also the vertex shift of some other graph.

3. If X is an M-step shift of finite type, then $X^{[M]}$ is a 1-step shift of finite type, equivalently a vertex shift. In fact, there is a graph G such that $X^{[M]} = \hat{X}_G$ and $X^{[M+1]} = X_G$.

Proof. (1) Let X be some 1-step shift such that $X = X_{\mathcal{F}}$, where we can take $\mathcal{F}$ to be a set of 2-blocks. Then $X = \hat{X}_G$ for the graph G with vertex set given by the alphabet of X, and a single edge from vertices v_1 to v_2 if $v_1v_2 \in \mathcal{B}(X)$; otherwise no edge exists. For the converse, we will see that vertex shifts are 1-step shifts. Let $\hat{X}_G$ be the vertex shift of a graph G, and let A denote its adjacency matrix. Then X is the shift space defined by the forbidden set

$$\mathcal{F} = \{vw : A_{vw} = 0\},$$

which is a set of 2-blocks.

(2) We saw in Proposition 3.12 that edge shifts are 1-step shifts. Thus by the previous proof, we can find a graph such that its vertex shift is the same as a given edge shift.

(3) To justify the first statement, see that forbidden words in X can be taken to be of length $M+1$, which will be regarded as a 2-block in $X^{[M]}$. The graph referenced in the second statement is the graph from Theorem 3.18; call it G. It remains only to see that $\hat{X}_G = X^{[M]}$. It is clear that the symbols of both these shifts are the allowed M-blocks X and that words (correspondingly, the vertices of a walk on G) in $\hat{X}_G$ occur exactly when the symbols progressively overlap. This is precisely when words are allowable in $X^{[M]}$ also, thus we can see the shift spaces are the same. ∎

Although every 1-step shift can be represented as a vertex shift, we tend to use edge shifts more often. The reasoning is that the same information can be presented much more concisely in an edge shift - by having multiple edges between vertices, without the limitation of having only a single edge from one vertex to another - and a vertex for every letter of the alphabet. For example, the edge shift X_A is the vertex shift $\hat{X}_B$ for the following adjacency matrices:

$$A = \begin{bmatrix} 3 & 2 \\ 3 & 1 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

It is important to note that although every edge shift is a shift of finite type (hence a vertex shift), the reverse is *not* true.

Example 3.23. The binary shift space called the *golden mean* shift is given by $\mathcal{F} = \{11\}$ and hence is a shift of finite type. However, it cannot be represented by an edge shift. If there were a graph G such that $X_{\mathcal{F}} = X_G$, then we can assume that it is essential. Such a graph would clearly have two edges - one for each letter of the alphabet. This leaves few possibilities: either G has one vertex, in which case its graph must correspond to the full shift, or G has two vertices and X_G is simply the two points $(10)^\infty$ and $(01)^\infty$. Any other configurations leaves some vertex stranded.

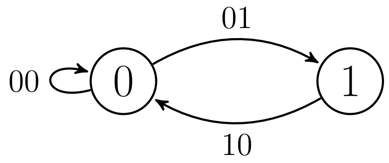


Figure 4: Representation of a conjugacy of the golden mean shift.

However, we can recode the golden mean shift to a conjugate higher block shift, which can then be represented on a graph - as illustrated above.

4 Sofic Shifts

4.1 Basic Properties and Characterization

We saw in the last section that shifts of finite type are conjugate to their higher block shifts, which can be represented as either vertex or edge shifts on unlabeled graphs. Here, we will see what types of shift space can be represented by *labeled* graphs.

Definition 4.1 (Labeled Graph)**.** A *labeled graph* $\mathcal{G}$ is a pair $(G, \mathcal{L})$, where G is a graph with edge set E, each element of which the labeling function $\mathcal{L} : E \to \mathcal{A}$ assigns a letter from the finite alphabet $\mathcal{A}$. We call G the *underlying graph* of $\mathcal{G}$.

For a labeled graph $\mathcal{G} = (G, \mathcal{L})$, we define the label of a walk $\pi = e_1 \dots e_n$ to be

$$\mathcal{L}(\pi) = \mathcal{L}(e_1) \dots \mathcal{L}(e_n),$$

and for a bi-infinite walk $\xi = \dots e_{-1}.e_0 e_1 \dots$,

$$\mathcal{L}_\infty(\xi) = \dots \mathcal{L}(e_{-1}).\mathcal{L}(e_0)\mathcal{L}(e_1) \dots .$$

Definition 4.2 (Labeled Graph Homomorphism)**.** Let $\mathcal{G} = (G, \mathcal{L}_G)$ and $\mathcal{H} = (H, \mathcal{L}_H)$ be labeled graphs. A *labeled graph homomorphism* from $\mathcal{G}$ to $\mathcal{H}$ is a graph homomorphism $(\Psi, \Phi) : G \to H$ such that it preserves the labeling of edges - that is, $\mathcal{L}_H(\Phi(e)) = \mathcal{L}_G(e)$ for all edges of G.

Naturally a labeled graph homomorphism which is also a graph isomorphism, is known as a *labeled graph isomorphism*. In a clear context, this is often just referred to as an isomorphism.

Definition 4.3 (Labeled Edge Shift)**.** For a labeled graph $\mathcal{G} = (G, \mathcal{L})$, we define the labeled edge shift $X_\mathcal{G}$ to be

$$X_\mathcal{G} = \{\mathcal{L}_\infty(\xi) : \xi \in X_G\} = \mathcal{L}_\infty(X_G)$$

Note that $\mathcal{L} : E(G) \to \mathcal{A}$ is a block map from 1-words in the edge shift of G. Thus $\mathcal{L}_\infty : X_G \to X_\mathcal{G}$ is actually the induced sliding block code. Hence it is immediate that the set of bi-infinite labeled walks on a graph do constitute a shift space - they are exactly the image of the edge shift under $\mathcal{L}_\infty$.

Definition 4.4 (Presentation)**.** A labeled graph $\mathcal{G}$ is said to be a *presentation* of a shift space X if $X = X_\mathcal{G}$. A walk π on $\mathcal{G}$ is said to be a presentation of $\mathcal{L}(\pi)$ if $\mathcal{L}(\pi) \in \mathcal{B}(X)$.

Note that there are multiple presentations of any given shift space. A walk π on $\mathcal{G}$ will also be a presentation of its label exactly when it can be extended to a bi-infinite walk on $\mathcal{G}$. This naturally occurs if the graph is irreducible.

Definition 4.5 (Sofic Shift)**.** A shift space X is said to be *sofic* if there exists some labeled graph $\mathcal{G}$ which is a presentation of X - that is, $X = X_{\mathcal{G}}$.

Theorem 4.6. *Every shift of finite type is sofic.*

Proof. Suppose that X is a shift of finite type. Then X is M-step, for some $M \geq 0$. The proof of 3.18 constructs a graph G such that $X^{[M+1]} = X_G$. For the case $M = 0$, we have that X is the full shift and that it is presented by the unlabeled graph given in the proof. This can be considered a labeled graph where the labeling function is simply the identity.

When $M > 0$, recall that the vertices of G are the M-blocks in X, and there exists an edge from $a_1a_2\dots a_M$ to $b_1b_2\dots b_M$ if and only if $a_2\dots a_M = b_1\dots b_{M-1}$. In this case, the edge is uniquely called $a_1\dots a_Mb_M = a_1b_1\dots b_M \in \mathcal{B}(X)$. We choose a labeling function $\mathcal{L}(a_1a_2\dots b_M) = a_1$ and show that the associated labeled graph $\mathcal{G} = (G, \mathcal{L})$ is a presentation of X. Let $\beta_{M+1} : X \to X^{[M+1]} = X_G$ be the higher block code given by

$$\beta_{M+1}(x)_i = x_{[i,i+M]}.$$

We can see that $\mathcal{L}_\infty(\beta_{M+1}(x))_i = \mathcal{L}(x_{[i,i+M]}) = x_i$ and thus $\mathcal{L}_\infty$ is inverse to β_{M+1}. Hence,

$$X_{\mathcal{G}} = \mathcal{L}_\infty(X_G) = \mathcal{L}_\infty(\beta_{M+1}(X)) = X$$

as required. ∎

Corollary 4.7. *A sofic shift is a shift of finite type if and only if it has a presentation $(G, \mathcal{L})$ such that $\mathcal{L}_\infty$ is a conjugacy.*

Proof. If X is a shift of finite type, then we have seen by the previous theorem that there exists a labeled graph $\mathcal{G} = (G, \mathcal{L})$ such that $X = X_{\mathcal{G}}$. Further, in the graph that was constructed, $\mathcal{L}_\infty$ had a sliding block inverse, and hence is a conjugacy.

Conversely, assume X is a sofic shift presented by $(G, \mathcal{L})$ such that $\mathcal{L}_\infty$ is a conjugacy. Then $X = X_{\mathcal{G}}$ is conjugate to X_G via $\mathcal{L}_\infty$, which is a shift of finite type. We have already seen that shifts conjugate to shifts of finite type are themselves shifts of finite type, so we are done. ∎

Theorem 4.8. *A shift space is sofic if and only if it is a factor of a shift of finite type.*

Proof. First, suppose that X is sofic with a presentation given by $(G, \mathcal{L})$. Then $\mathcal{L}_\infty : X_G \to X_{\mathcal{G}}$ is surjective by definition of $X_{\mathcal{G}}$. Note that $X = X_{\mathcal{G}} = \mathcal{L}_\infty(X_G)$ to see that X is a factor of the shift of finite type X_G.

For the converse, suppose we have a shift space X for which there is a factor code $\phi : Y \to X$ from an M-step shift of finite type Y. Let ϕ be induced by Φ with memory m and anticipation n. Remembering Y is N-step for all $N \geq M$, and that we can increase m if necessary, we can consider Y to be $(m+n)$-step. Thus there exists a graph G such that $Y^{[m+n+1]} = X_G$.

Now we will make use of Proposition 2.39, which says we can find a conjugacy ψ and a 1-block code $\hat{\phi}$ such that the following diagram commutes:

$$\begin{tikzcd}
Y \arrow[r, "\psi"] \arrow[dr, "\phi"'] & Y^{[m+n+1]} = Y_G \arrow[d, "\hat{\phi}"] \\
& X
\end{tikzcd}$$

We can take $\hat{\phi}$ to be induced by the map $\hat{\Phi} : \mathcal{B}_{m+n+1}(X) \to \mathcal{A}$ where $\mathcal{A}$ is the alphabet of X. Noting that the edges of G are $(n + m + 1)$-words we see we can label G with $\mathcal{L} = \hat{\Phi}$ to create a labeled graph $\mathcal{G}$. These maps induce the same sliding block codes $\mathcal{L}_\infty = \hat{\phi}$. Noting that $\mathcal{L}_\infty = \hat{\phi} = \phi \circ \psi^{-1}$, we can see that it is surjective as the composition of surjective functions. Hence the image of Y_G under $\mathcal{L}_\infty$ is all of X and thus $X = \mathcal{L}_\infty(X_G) = X_{\mathcal{G}}$, showing that X is sofic. ∎

Immediately from this, we can see that sofic shifts are actually the closure of shifts of finite type under factor maps. This is because if we have a sofic shift Y, it is the image under some factor ϕ of a shift of finite type X. Now take a factor map ψ from Y to some shift space Z. Then Z can be written as the image of X under $\psi \circ \phi$, a factor map itself, and hence it is in fact sofic.

4.2 Follower Set Graphs

Definition 4.9 (Follower Set)**.** Let X be a shift space and w be a word in $\mathcal{B}(X)$. The *follower set* $F_X(w)$ of w in X is intuitively defined as the set of all words which can follow w in X, i.e.

$$F_X(w) = \{v \in \mathcal{B}(X) : wv \in \mathcal{B}(X)\}.$$

The collection of all follower sets in X is denoted by

$$C_X = \{F_X(w) : w \in \mathcal{B}(X)\}.$$

Each follower set is an infinite set of words, since every word in the language of X can be extended indefinitely to the right. There can also be an infinite number of words in the language of a shift space. However many words can also share the same follower set. A simple example is if we take X to be the full shift: any word can be followed by any other word, thus $F_X(w) = \mathcal{B}(X)$, for every $w \in \mathcal{B}(X)$; hence C_X is a singleton set.

Definition 4.10 (Follower Set Graph)**.** Suppose X is a shift space over $\mathcal{A}$ such that C_X is finite. Then we can construct a labeled graph $\mathcal{G} = (G, \mathcal{L})$, called the *follower set graph* of X. Let the vertices be the elements of C_X - so each vertex represents a follower set (likely this is the follower set of many different words). Let $v_1 = F_X(w) \in C_X$ and $a \in \mathcal{A}$. If $wa \in \mathcal{B}(X)$, then there exists a follower set $F_X(wa)$ which corresponds to a vertex - call it v_2. In this case, construct an edge from v_1 to v_2, labeled as a. Else if wa is not an allowed word, do nothing. Repeating this process for every vertex and every $a \in \mathcal{A}$ yields a labeled graph $\mathcal{G}$.

Note that this graph is indeed well defined - the construction is independent of the choice of the representative follower set for a vertex. To see this, note that when $F_X(w) = F_X(w')$, then wa is allowed if and only if $w'a$ is allowed, as both w and w' share the same follower set. Similarly $F_X(wa) = F_X(w'a)$ since each set contains exactly the words in $F_X(w) = F_X(w')$ which begin with a.

Example 4.11. In Figure 5, the vertices labeled C_0, C_1 and C_2 are associated with the three distinct follower sets $F_X(0)$, $F_X(1)$ and $F_X(01)$ of the even shift respectively. In fact, it is easy to check that

$$F_X(w) = \begin{cases} C_0 & \textit{if } w \textit{ contains no 1's,} \\ C_1 & \textit{if } w \textit{ ends in } 10^{2k} \textit{ for some } k \geq 0, \\ C_2 & \textit{if } w \textit{ ends in } 10^{2k+1} \textit{ for some } k \geq 0, \end{cases}$$

and so the above follower sets are in fact the only ones.

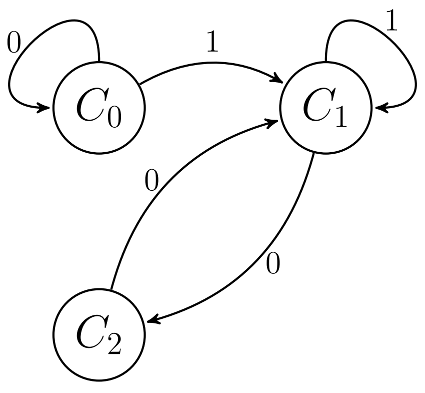


Figure 5: Follower set graph for the even shift (defined in Example 2.7).

Proposition 4.12. *If $\mathcal{G}$ is the follower set graph of some shift space X, then $\mathcal{G}$ is a presentation of X, and thus X must be sofic.*

Proof. To show that $X = X_{\mathcal{G}}$, it suffices by Proposition 2.10(3) to show that they have the same langauge. First, suppose that $u = a_1a_2 \dots a_n \in \mathcal{B}(X_{\mathcal{G}})$. This is equivalent to saying there is a path π in $\mathcal{G}$ labeled as u. Assume without loss of generality that π starts at vertex $F_X(w)$. The definition of $\mathcal{G}$ implies that since there is an edge labeled a_1 from $F_X(w)$ to some other vertex, and that $wa_1 \in \mathcal{B}(X)$. Further the next vertex has as a representative $F_X(wa_1)$. Continuing this argument, we see that $wa_1a_2 \in \mathcal{B}(X)$ and inductively that $wa_1wa_2 \dots wa_n = wu \in \mathcal{B}(X)$. Hence $u \in \mathcal{B}(X)$.

To prove the reverse inclusion, let $u \in \mathcal{B}(X)$. Again by Proposition 2.10(1)(b), we can find some w, v, t such that $wvut \in \mathcal{B}(X)$ and $|v|, |t| > |C_X| = |V(G)|$. Take $v = v_1 \dots v_n$. Since $w \in \mathcal{B}(X)$ there exists a follower set $F_X(w)$, which is a representative of some vertex of $\mathcal{G}$. Similar to the idea above, we see that there must exist an edge labeled v_1 from this vertex to some vertex representing $F_X(wv_1)$. Inductively, there is a path in the underlying graph labeled vut in $\mathcal{G}$. Since each of v is of length greater than the number of vertices in $\mathcal{G}$, it must contain a cycle. Hence, we can write v as $\alpha\beta\gamma$ where β is a cycle. And similarly for t, which we will write as $\delta\omega\eta$, with ω as a cycle. Finally, let π be the underlying path labeled u, which we have already seen to exist. Hence we know there exists the bi-infinite path $\beta^\infty\gamma\pi\delta\omega^\infty$ in G, so its labeling is a point of $X_{\mathcal{G}}$. Its labeling contains u and thus $u \in \mathcal{B}(X_{\mathcal{G}})$ as required. ∎

Theorem 4.13. *A shift space is sofic if and only if it has a finite number of follower sets.*

Proof. If a shift space X has a finite number of follower sets, we can construct the follower set graph - which is a labeled graph presentation of X. Hence X is sofic.

For the forwards implication, let X be a sofic shift and so we can take labeled graph $\mathcal{G} = (G, \mathcal{L})$ to be a presentation of X. For a word $w \in \mathcal{B}(X)$, we can find one or more presentations of w in $\mathcal{G}$. Let T_w denote the set of vertices at which these presentations terminate. Thus, the labeled walks that can follow these presentations in $\mathcal{G}$ are exactly those that start at a vertex in T_w, and these are exactly the words in X that can follow w. If the presentations of two words v and w are such that $T_v = T_w$, then their follower sets will be equal. Since there is only a finite number of subsets of the vertex set of $\mathcal{G}$, there can only be a finite number of unique follower sets. ∎

4.3 Minimal Right Resolving Presentations

This section leads up to one of the main results of the paper. We will show that an irreducible sofic shift has a unique minimal right resolving presentation, hence giving a canonical way to present such a shift graphically. Further, we are able to give a characterization of this graph - it is exactly the right resolving graph which is both irreducible and follower separated. This will lead on to a method of finding this presentation given a follower set graph, then a more subtle link between the two presentations given in the next section.

Definition 4.14 (Right Resolving)**.** A labeled graph is called *right resolving* if, for each vertex v, the edges starting at v carry different labels.

Right resolving presentations are important, because they help eliminate the ambiguity around different presentations of a given word. With a right resolving graph, any word has at most one presentation starting at each vertex. Another characterization of this is that the restriction of the labeling function to edges coming from a given vertex is injective.

Corollary 4.15. *Every sofic shift has a right-resolving presentation.*

Proof. The follower set graph constructed in the previous section is a right resolving presentation by construction. ∎

Definition 4.16 (Minimal Right Resolving Presentation)**.** A *minimal right resolving presentation* of a sofic shift X is a labeled graph $\mathcal{G}$ such that $X = X_{\mathcal{G}}$ and $\mathcal{G}$ has the fewest vertices among all right resolving presentations of X.

Definition 4.17 (Follower Set of a Vertex)**.** Let $\mathcal{G}$ be a labeled graph and v a vertex in $\mathcal{G}$. Then the *follower set* of v, $F_{\mathcal{G}}(v)$ is the set of labels of paths beginning at v. That is,

$$F_{\mathcal{G}}(v) = \{\mathcal{L}(\pi) : \pi \in \mathcal{B}(X_G),\ i(\pi) = v\}.$$

We say that $\mathcal{G}$ is *follower separated* if distinct vertices have distinct follower sets.

Definition 4.18 (Merged Graph)**.** Let $\mathcal{G}$ be a labeled graph. Call two vertices v, w of $\mathcal{G}$ equivalent if $F_{\mathcal{G}}(v) = F_{\mathcal{G}}(w)$. This is an equivalence relation and so partitions the set of vertices. We will now construct $\mathcal{H}$, the *merged graph* of $\mathcal{G}$. Let $\mathcal{H}$ have the set of vertices given by the above equivalence classes; that is, each vertex of $\mathcal{H}$ is a set of vertices in $\mathcal{G}$ which share the same follower set. There is an edge in $\mathcal{H}$ labeled a from I to J exactly when there is an edge labeled a in $\mathcal{G}$ from some vertex in I to some vertex in J.

Lemma 4.19. *Let $\mathcal{G}$ be a labeled graph and $\mathcal{H}$ the corresponding merged graph. Then $\mathcal{H}$ is follower separated and $X_{\mathcal{G}} = X_{\mathcal{H}}$.*

Proof. Let a vertex v in $\mathcal{G}$ be a representative of the equivalence class I - a vertex in $\mathcal{H}$. We first show that $F_{\mathcal{G}}(v) = F_{\mathcal{H}}(I)$. A path in $\mathcal{G}$ gives rise to a path in $\mathcal{H}$ labeled similarly, by passing to equivalence classes. Thus $F_{\mathcal{G}}(v) \subseteq F_{\mathcal{H}}(I)$.

Next assume that $F_{\mathcal{G}}(v) \neq F_{\mathcal{H}}(I)$ for at least one vertex v and corresponding I. Among all such v and I, there are words in $F_{\mathcal{H}}(I)$ but not $F_{\mathcal{G}}(v)$. Select $w = w_1 w_2 \dots w_n$ to be a word of minimal length in this set. Then we have that $F_{\mathcal{G}}(v) \cap \mathcal{A}^k = F_{\mathcal{H}} \cap \mathcal{A}^k$ for all $k < n$ and all v, I, by minimality of w. Note that there is an edge labeled w_1 from I in $\mathcal{H}$ to some vertex J and hence an edge in $\mathcal{G}$ labeled similarly from $v' \in I$ to $w \in J$. Now $w_2 \dots w_n \in F_{\mathcal{H}}(J) \cap \mathcal{A}^{n-1} = F_{\mathcal{G}}(w) \cap \mathcal{A}^{n-1}$. And so we also have a path in $\mathcal{G}$ starting at v' and labeled w i.e. $w \in F_{\mathcal{G}}(v') = F_{\mathcal{G}}(v)$, contradicting our initial choice of w. Hence $F_{\mathcal{G}}(v) = F_{\mathcal{G}}(I)$.

It is now clear that $\mathcal{H}$ is follower separated, since if two vertices in $\mathcal{H}$ share the same follower set, we also have that their equivalence class representatives in $\mathcal{G}$ share the same follower set - hence, by the definition of the vertices of $\mathcal{H}$, these two vertices must be the same.

Since $\mathcal{B}(X_{\mathcal{G}})$ is the union of the follower sets in $\mathcal{G}$ over each vertex, and similarly for $\mathcal{H}$, we immediately obtain $\mathcal{B}(X_{\mathcal{G}}) = \mathcal{B}(X_{\mathcal{H}})$ and hence that $X_{\mathcal{G}} = X_{\mathcal{H}}$. ∎

Lemma 4.20. *Let $\mathcal{G}$ be a labeled graph and $\mathcal{H}$ the corresponding merged graph. If $\mathcal{G}$ is irreducible then $\mathcal{H}$ is irreducible. Further, if $\mathcal{G}$ is right resolving then $\mathcal{H}$ is right resolving.*

Proof. First, assume $\mathcal{G}$ is irreducible. Let I, J be distinct vertices of $\mathcal{H}$. Then $v \in I$ and $w \in J$ are vertices of $\mathcal{G}$. By the irreducibility of $\mathcal{G}$, there is a path π from v to w and passing to equivalence classes gives a path in $\mathcal{H}$ from I to J.

Next, assume that $\mathcal{G}$ is right resolving. Whenever there is an edge labeled a from some I to J in $\mathcal{H}$, there must be some similarly labeled edge from some $v \in I$ to some $w \in J$ in $\mathcal{G}$. Since $\mathcal{G}$ is right resolving, this is the only edge from v labeled a. Note that paths that start at w are exactly the subset of paths that start at v which begin by moving from v to w over this edge. Hence

$$F_{\mathcal{G}}(w) = \{w_1 \dots w_n : aw_1 \dots w_n \in F_{\mathcal{G}}(v)\}.$$

Now assume there is a second edge labeled a from I to some other vertex J'. Then, as above, there is another edge in $\mathcal{G}$ labeled a from some $v' \in I$ to some $w' \in J'$. Again, we have that

$$F_{\mathcal{G}}(w') = \{w_1 \dots w_n : aw_1 \dots w_n \in F_{\mathcal{G}}(v')\}.$$

Since $F_{\mathcal{G}}(v) = F_{\mathcal{G}}(v')$ we have that $F_{\mathcal{G}}(w) = F_{\mathcal{G}}(w')$ and hence

$$F_{\mathcal{H}}(J) = F_{\mathcal{G}}(w) = F_{\mathcal{G}}(w') = F_{\mathcal{H}}(J')$$

proving that $J = J'$, by the property of $\mathcal{H}$ being follower separated. Thus, any edged labeled a coming from I must terminate at the same vertex J. But by the definition of $\mathcal{H}$, there is at most one edge labeled a from I to J. Hence $\mathcal{H}$ is right resolving - there is at most one edge with a given label from each vertex. ∎

Proposition 4.21. *A minimal right resolving presentation of a sofic shift is follower separated.*

Proof. Let $\mathcal{G}$ be a minimal right resolving presentation of a sofic shift X. If $\mathcal{G}$ is not follower separated, then the merged graph $\mathcal{H}$ would be a right resolving presentation of X with fewer vertices, contradicting minimality. ∎

Lemma 4.22. *Suppose that X is an irreducible sofic shift, and that $\mathcal{G}$ is a minimal right resolving presentation of X. Then $\mathcal{G}$ is an irreducible graph.*

Proof. Let $\mathcal{G} = (G, \mathcal{L})$ be a minimal right resolving presentation of X. We first show that for every vertex $I \in V(\mathcal{G})$, there is a word $u_I \in \mathcal{B}(X_{\mathcal{G}})$ such that every path in $\mathcal{G}$ presenting u_I contains I. Suppose this is false for some vertex I. Then every word is presented by a path that does not pass through I, and hence I and all edges incident to it can be removed from $\mathcal{G}$

to create a presentation $\mathcal{H}$ of X. The graph $\mathcal{H}$ is right resolving and has fewer vertices than $\mathcal{G}$, contradicting minimality.

Now let I and J be distinct vertices of $\mathcal{G}$ and u_I and u_J words satisfying the above property. By the irreducibility of X, there exists a word w such that $u_I w u_J \in \mathcal{B}(X)$. So there exists a path π in G presenting $u_I w u_J$. But the subpath of π presenting u_I and u_J contains vertices I and J, and thus there exists some subpath of π from I to J. ∎

Proposition 4.23. *A sofic shift is irreducible if and only if it has an irreducible presentation.*

Proof. Let X have an irreducible presentation $\mathcal{G} = (G, \mathcal{L})$. Then words $u, w \in \mathcal{B}(X_{\mathcal{G}}) = \mathcal{B}(X)$ have presentations in $X_{\mathcal{G}}$ and thus underlying paths π, τ in G. By the irreducibility of $\mathcal{G}$, there is a path ω from $t(\pi)$ to $i(\tau)$. Let $v = \mathcal{L}(\omega)$. Then

$$\mathcal{L}(\pi\omega\tau) = uvw \in \mathcal{B}(X_{\mathcal{G}})$$

since in an irreducible graph, every path is part of a bi-infinite walk. Hence $X_{\mathcal{G}}$ is irreducible.

For the converse, assume $X_{\mathcal{G}}$ is irreducible. We know that a finite right resolving presentation exists for $X_{\mathcal{G}}$ and hence a minimal right resolving presentation also exists. By the above lemma, such a graph is also irreducible. ∎

Definition 4.24 (Synchronizing Word)**.** Let $\mathcal{G} = (G, \mathcal{L})$ be a labeled graph. A word $w \in \mathcal{B}(X_{\mathcal{G}})$ is a *synchronizing word* for $\mathcal{G}$ if all paths in G presenting w terminate at the same vertex. We say that w *focuses* to this vertex.

Lemma 4.25. *Suppose that $\mathcal{G}$ is a right resolving labeled graph, and that w is a synchronizing word for $\mathcal{G}$. Then any word of the form wu in $\mathcal{B}(X_{\mathcal{G}})$ is also synchronizing for $\mathcal{G}$. If w focuses to I, then $F_{X_{\mathcal{G}}}(w) = F_{\mathcal{G}}(I)$.*

Proof. For the first statement, let w focus to I. Any path presenting wu has the from $\pi\tau$, where π terminates at I. But by the right resolving property, there is only one path starting at I labeled u, thus wu focuses to $t(\tau)$ and so is synchronizing.

For the second statement, note that any word following w is presented by some path starting at I. And the converse also holds: any path starting at I presents some word which can follow w. ∎

Proposition 4.26. *Suppose that $\mathcal{G}$ is a right resolving graph that is follower separated. Then every word $u \in \mathcal{B}(X_{\mathcal{G}})$ can be extended on the right to a synchronizing word $uw \in \mathcal{B}(X_{\mathcal{G}})$.*

Proof. For any word $w \in \mathcal{B}(X_{\mathcal{G}})$, let $T(w)$ denote the set of terminal vertices of all paths presenting w. If $T(u)$ is just a single vertex then u is synchronizing and w can be any word such that $uw \in \mathcal{B}(X_{\mathcal{G}})$.

Suppose then that $T(u)$ has more than one vertex, and let I and J be distinct vertices in $T(u)$. Then $F_{\mathcal{G}}(I) \neq F_{\mathcal{G}}(J)$, as $\mathcal{G}$ is follower separated, and so there exists some word in one set but not the other. Without loss of generality, we can assume $v_1 \in F_{\mathcal{G}}(I)$ but $v_1 \notin F_{\mathcal{G}}(J)$. Since $\mathcal{G}$ is right resolving, there is at most one labeled path labeled v_1 starting at each vertex of $T(u)$. These are all the paths presenting uv_1. Each vertex in $T(u)$ corresponds to at most one vertex in $T(uv_1)$ via these paths. Since $v_1 \notin F_{\mathcal{G}}(J)$, it follows that $T(uv_1)$ does not contain J, and hence has fewer elements than $T(u)$. We can continue inductively, and since $T(u)$ was a finite set to begin with, eventually we will reach some $T(uv_1 \dots v_n)$ with only one element. Then we have found our $w = v_1 \dots v_n$. ∎

Proposition 4.27. *If $\mathcal{G}$ and $\mathcal{G}'$ are irreducible, follower separated, right resolving presentations of a sofic shift, then $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic as labeled graphs.*

Proof. First we find a word that is synchronizing in both graphs. By applying Proposition 4.26 to our first graph we can find a word u_1 that is synchronizing for $\mathcal{G}$. Next, we can apply the Proposition again - this time to our second graph - to find an extension $w = u_1u_2$ that is synchronizing in $\mathcal{G}'$. By Lemma 4.25, w remains synchronizing in $\mathcal{G}$ also.

We now use this synchronizing word to identify vertices and inductively construct a labeled graph isomorphism from $\mathcal{G}$ to $\mathcal{G}'$. Note that w focuses to some $I_0 \in V(\mathcal{G})$ and to some $I'_0 \in V(\mathcal{G}')$. Now let $\mathcal{G}_k$ and $\mathcal{G}'_k$ be the subgraphs consisting of all vertices at most k edges away from these I_0 and I'_0 respectively, and all edges contained in paths of length at most k starting from these base vertices.

The basis of our induction is to note that $\mathcal{G}_0$ and $\mathcal{G}'_0$ are simply the lone vertices I_0 and I'_0 and hence are trivially isomorphic. Next we assume that the subgraphs $\mathcal{G}_k$ and $\mathcal{G}'_k$ are isomorphic via (Ψ_k, Φ_k) and aim to prove that there exists an isomorphism between graphs $\mathcal{G}_{k+1}$ and $\mathcal{G}'_{k+1}$ as well. Since our graphs are irreducible and have a finite number of vertices, this will suffice to show that the entire graphs $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic.

Take a vertex $I_k \in V(\mathcal{G}_k)$. Then by irreducibility, there exists some path π from I_0 to I_k in $\mathcal{G}_k$ labeled as, say, v. By our assumption of isomorphism, we can find a corresponding $I'_k \in V(\mathcal{G}'_k)$ and corresponding path π' from I'_0 to I'_k also labeled v. Note that by Lemma 4.25,

$$F_{\mathcal{G}}(I_k) = F_{X_{\mathcal{G}}}(wv) = F_{X_{\mathcal{G}'}}(wv) = F_{\mathcal{G}}(I'_k).$$

This holds regardless of our choice of paths π and π' or their labellings. Hence if we take some edge e out of I_k labeled a, there must exist an edge e' out of I'_k also labeled a. By the right resolving property, both of these edges are unique. Take edges e and e' to terminate at vertices J and J' respectively. Then we set $\Psi_{k+1}(e) = e'$ and $\Phi_{k+1}(J) = J'$. Repeat for each letter in the alphabet and for each vertex in $\mathcal{G}_k$. In this way, we can account for all vertices in $\mathcal{G}_{k+1}$ and each will map to some vertex in $\mathcal{G}'_{k+1}$. It is clear this mapping respects the initial and terminating vertices of its edges, as well as the labeling, and hence is indeed a graph homomorphism from $\mathcal{G}_{k+1}$ to $\mathcal{G}'_{k+1}$.

Also note that since there exists a path labeled va from I_0 to J and I'_0 to J', we have that

$$F_{\mathcal{G}}(J) = F_{X_{\mathcal{G}}}(wva) = F_{X_{\mathcal{G}'}}(wva) = F_{\mathcal{G}}(J').$$

This forces the mapping to be well defined, since if a vertex were mapped to two different places, it would contradict the follower separation of $\mathcal{G}'$. By a similar argument, the mapping is injective - else a contradiction of the follower separated property of $\mathcal{G}$ arises.

The mapping is also surjective. Take some vertex J' in $\mathcal{G}'_{k+1}$. Then there exists some I'_k in $\mathcal{G}'_k$ and an edge e' labeled as, say, a, from I'_k to J'. This I'_k has a corresponding I_k in $\mathcal{G}$ by the assumption of isomorphism. Since $F_{\mathcal{G}}(I_k) = F_{\mathcal{G}'}(I'_k)$, there exists an edge e labeled a from I_k, which terminates at some J in $\mathcal{G}_{k+1}$. This J will be mapped to J'. Thus our map is indeed an isomorphism, and the induction completes our argument. ∎

Theorem 4.28. *Any two minimal right resolving presentations of an irreducible sofic shift are isomorphic as labeled graphs.*

Proof. If $\mathcal{G}$ and $\mathcal{G}'$ are both minimal right resolving presentations of an irreducible sofic shift, then by Lemma 4.22 they are both irreducible graphs. By Proposition 4.21, minimal right resolving graphs are follower separated, hence $\mathcal{G}$ and $\mathcal{G}'$ have this property also. Thus by Proposition 4.27, they are both isomorphic as labeled graphs. ∎

Corollary 4.29. *Let X be an irreducible sofic shift, and $\mathcal{G}$ a right resolving presentation of X. Then $\mathcal{G}$ is the graph that is minimal among all right resolving presentations of X if and only if $\mathcal{G}$ is irreducible and follower separated.*

Proof. If $\mathcal{G}$ is the minimal right resolving presentation of an irreducible sofic shift, by Lemma 4.22 it is an irreducible graph. Proposition 4.21 says that right resolving minimal graphs are follower separated, so $\mathcal{G}$ has this property also. For the converse, take $\mathcal{G}'$ to be an irreducible, follower separated and right resolving presentation of X, and $\mathcal{G}$ the minimal right resolving presentation described above. Then by Proposition 4.27, the two graphs are isomorphic. ∎

Corollary 4.30. *Let X be an irreducible sofic shift and let $\mathcal{G}$ be an irreducible, right resolving presentation of X. Then the merged graph of $\mathcal{G}$ is the minimal right resolving presentation of X.*

Proof. Apply Lemma 4.19, Lemma 4.20 and Corollary 4.29. ∎

Example 4.31. The following counterexample is due to N. Jonoska [5], and shows that Theorem 4.28 can fail for reducible shifts. Let $\mathcal{G}$ be the graph (a) in Figure 6, below, and $\mathcal{H}$ be graph (b). It is simple enough to check by inspection that the graphs are right resolving. Further, we can see they are not isomorphic since $\mathcal{H}$ has a self-loop labeled a and $\mathcal{G}$ does not. However, we will also show that they are both presentations of the same shift space, and that no graph with fewer vertices can be a right resolving presentation of this shift.

First, we show that $\mathcal{B}(X_\mathcal{G}) = \mathcal{B}(X_\mathcal{H})$, hence showing that $X_\mathcal{G} = X_\mathcal{H}$. Any path in $\mathcal{G}$ can be extended to one which cycles through vertices 1 and 2, later dropping down to vertices 3 or 4 before continuing. Hence $\mathcal{B}(X_\mathcal{G})$ consists of all *subwords* of the following words:

$$v_1 = a^k b c^{m_1} b a c^{m_2} b a c^{m_3} b a \ldots,$$
$$v_2 = a^k b a c^{n_1} b a c^{n_2} b a c^{n_3} b a \ldots,$$

where exponents k, m_i, n_j are integers greater than or equal to 0. Similarly, there are three types of paths on $\mathcal{H}$ which contain all words - which path depends on which edge exits vertex 1. Thus $\mathcal{B}(X_\mathcal{H})$ consists of all subwords of the three types of words:

$$w_1 = a^k b a c^{p_1} b a c^{p_2} b a c^{p_3} b a \ldots,$$
$$w_2 = a^k b c c^{q_1} b a c^{q_2} b a c^{q_3} b a \ldots,$$
$$w_3 = a^k b b a c^{r_1} b a c^{r_2} b a c^{r_3} b a \ldots,$$

where again the exponents are greater than or equal to 0. Words of type w_1 correspond to those of type v_2. Type w_2 corresponds to type v_1 where $m_1 = 1 + q_1$ and w_3 corresponds to type v_1 with $m_1 = 0$. This proves the languages are equal.

Next we show that no right resolving graph with fewer vertices can present this shift space. It is trivial to see that a graph with zero or one vertices cannot present it. For the less simple cases, observe that there are three distinct follower sets, $F_X(aa)$, $F_X(c)$ and $F_X(cb)$, each containing a word which is in neither of the other two. For example, we have

$$aab \in F_X(aa)\ \{F_X(c) \cup F_X(cb)\}, c \in F_X(c)\ \{F_X(aa) \cup F_X(cb),\}ac \in F_X(cb)\ \{F_X(aa) \cup F_X(c)\}.$$

Assuming we can find a minimal right resolving presentation with fewer than 4 vertices, we know there must exist some presentation within it of the word $aaaab$. Hence there is a vertex where a presentation of aa terminates and where a presentation of aab begins. Such a vertex cannot also be the terminating point of presentations of c or cb, since doing so would mean $aab \in F_X(c)$ or $aab \in F_X(cb)$ respectively. A similar argument carries for the remaining two follower sets, hence showing that such a graph cannot have only two vertices, and further that the follower set of each one of the three vertices of the graph is equal to one of the distinct follower sets above. It also follows that aab can only be presented with a path starting at the vertex with follower set $F_X(aa)$. However, since we can also see that $F_X(aab) = F_X(c) \cup F_X(cb)$, there must be paths from vertex $F_X(aa)$ to both vertices $F_X(c)$ and $F_X(cb)$, contradicting the right resolving property.

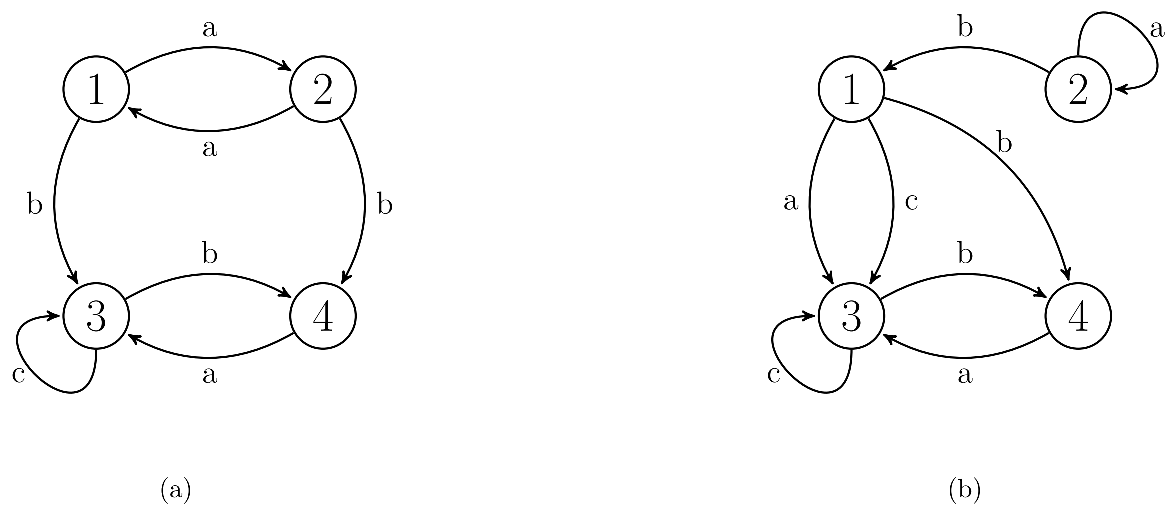


Figure 6: Two reducible, minimal right resolving graphs presenting the same shift

In fact, there is a way to obtain the minimal right resolving presentation of an irreducible sofic shift from its follower graph [7].

Definition 4.32 (Intrinsically synchronizing)**.** Given an irreducible shift space X, a word $v \in \mathcal{B}(X)$ is *intrinsically synchronizing* if whenever $uv \in \mathcal{B}(X)$ and $vw \in \mathcal{B}(X)$, then $uvw \in \mathcal{B}(X)$.

Lemma 4.33. *Let X be an irreducible sofic shift and $\mathcal{G} = (G, \mathcal{L})$ be its minimal right resolving presentation. Then $v \in \mathcal{B}(X)$ is synchronizing in $\mathcal{G}$ if and only if v is intrinsically synchronizing.*

Proof. If $uv \in \mathcal{B}(X)$ and $vw \in \mathcal{B}(X)$ then there are presentations of uv and vw - say, τ and $\eta\mu$ respectively, where $\mathcal{L}(\tau) = uv$, $\mathcal{L}(\eta) = v$ and $\mathcal{L}(\mu) = w$ and each is part of some bi-infinite walk. If v is synchronizing, then every presentation of v terminates at some vertex I, so τ and η both terminate at I. Hence μ begins at I and $\tau\mu$ is a path labeled uvw. Further, this path can be extended to become part of a bi-infinite walk on $\mathcal{G}$ and so its label $uvw \in \mathcal{B}(X)$.

For the converse, we aim to show that an intrinsically synchronizing word is synchronizing in $\mathcal{G}$. In fact we prove the contrapositive - that if a word is not synchronizing then it is not intrinsically synchronizing. Suppose a word $v \in \mathcal{B}(X)$ is not synchronizing. Then it has two presentations τ and ω which terminate at distinct vertices I and I' respectively. The presentation $\mathcal{G}$ is follower separated by Proposition 4.21 and so $F(I) \neq F(I')$. Without loss of generality, there exists some $w \in F(I) \setminus F(I')$. Note $vw \in \mathcal{B}(X)$.

Since $\mathcal{G}$ is right resolving and τ and ω present the same word, $i(\tau) \neq i(\omega)$. By Proposition 4.26, we can find some synchronizing word u' in $\mathcal{G}$, and by the irreducibility of $\mathcal{G}$ we can extend this word to $u'u''$ terminating at $i(\omega)$. By Lemma 4.25, $u'u''$ remains synchronizing in $\mathcal{G}$. Setting $u = u'u''$, it is clear that $uv \in \mathcal{B}(X)$. Since uw is also synchronizing, all its presentations terminate at I'. However we chose $v \notin F(I')$ and so $uvw \notin \mathcal{B}(X)$. ∎

Theorem 4.34. *The minimal right resolving presentation of an irreducible shift X is exactly the labeled subgraph of the follower set graph of X formed by using only the follower sets of intrinsically synchronizing words, and the edges between them.*

Proof. Denote with $\mathcal{G}$ the follower set graph of X, and $\mathcal{H}$ the subgraph of $\mathcal{G}$ formed by taking only the vertices representing follower sets of intrinsically synchronizing words, and the edges between these vertices. First we show that $\mathcal{H}$ actually presents X. Since $\mathcal{H} \subseteq \mathcal{G}$, we automatically have that $X_{\mathcal{H}} \subseteq X_{\mathcal{G}} = X$. To show the reverse inclusion, we show that $\mathcal{B}(X) \subseteq \mathcal{B}(X_{\mathcal{H}})$. Consider any $w \in \mathcal{B}(X)$. Since $\mathcal{G}$ is right resolving, we can find some synchronizing word u for it. By the irreducibility of X, there exists a word $v \in \mathcal{B}(X)$ such that $uvw \in \mathcal{B}(X)$, In fact, each of $uv, uvw_1, uvw_1w_2, \ldots, uvw$ are also synchronizing in $\mathcal{G}$. The first half of the argument from the previous lemma applies to any presentation of X, so each of the above words is also intrinsically synchronizing for X. Hence $F(uv)$ is a vertex in $\mathcal{H}$, as is $F(uvw_1), \ldots F(uvw)$, and by the definition of $\mathcal{H}$ there exists a path visiting each of these vertices in order, labeled w. Hence if we can show $\mathcal{H}$ to be irreducible, we also have that $X_{\mathcal{H}} = X$.

In fact, by Corollary 4.29, showing $\mathcal{H}$ is right resolving, irreducible and follower separated also shows it is the minimal right resolving graph of X. The right resolving property is inherited immediately from $\mathcal{G}$.

To see that $\mathcal{H}$ is irreducible, take any two distinct vertices $F(u)$ and $F(w)$ of $\mathcal{H}$, where u and v are intrinsically synchronizing words. Note that the follower sets of these vertices may no longer be the follower sets they represented in $\mathcal{G}$. Since X is irreducible, there exists some $v \in \mathcal{B}(X)$ such that $uvw \in \mathcal{B}(X)$. Since u is intrinsically synchronizing for X, the above lemma implies that it is synchronizing in the minimal right resolving presentation (there is no need to assume here that $\mathcal{H}$ is this graph). Hence any extension, and specifically uvw, is also synchronizing in this graph, and so again by the above lemma, we see that uvw remains intrinsically synchronizing in $\mathcal{H}$. Thus there is a vertex in $\mathcal{H}$ labeled $F(uvw)$, and so a path from $F(u)$ to $F(uvw)$ labeled w. Further, since w and uvw are synchronizing in the minimal right resolving presentation of X, they both focus to the same vertex, and so by Lemma 4.25, $F(w) = F(uvw)$. Thus we have found a path between two arbitrary vertices of $\mathcal{H}$.

Finally, we show that $\mathcal{H}$ is follower separated. Consider two distinct vertices of $\mathcal{H}$, labeled $F(u)$ and $F(v)$, for intrinsically synchronizing words u and v. Without loss of generality then, there exists some $w \in F(u) \setminus F(v)$. Hence no path with label w leaves the vertex labeled $F(v)$ in $\mathcal{G}$ and hence in $\mathcal{H}$. However there is a path labeled w leaving $F(u)$ in $\mathcal{G}$, and further uw is intrinsically synchronizing, so $F(uw)$ is a vertex in $\mathcal{H}$. This implies that there exists a path from $F(u)$ to $F(uw)$ in $\mathcal{H}$ labeled w. So there is a path labeled w from $F(u)$ but there cannot be a path from $F(v)$ with the same label. Hence the follower sets of our arbitrary vertices $F(u)$ and $F(v)$ are different, and $\mathcal{H}$ is follower separated. ∎

4.4 Entropy

In this section, we introduce entropy, an invariant under unlabeled graph isomorphism. It may be viewed as a measurement of "information capacity" of a shift space - or of ability to transmit messages. We present only the results needed to prove our main theorem relating to unlabeled isomorphism of follower set graphs.

Definition 4.35 (Entropy)**.** Let X be a shift space. The *entropy* of X is defined by

$$h(X) = \lim_{n\to\infty} \frac{1}{n} \log_2 |\mathcal{B}_n(X)|.$$

In the special case that $X = \emptyset$, the definition gives us that $h(\emptyset) = -\infty$.

The number of n-blocks appearing in sequences from X gives us some measure of complexity. The definition of entropy given above roughly summarizes the growth rate of comlexity as n increases. In what follows, we will assume base 2 for all logarithms.

To know that we are working with a well defined quantity, we must show that this limit always exists.

Lemma 4.36. *Let $(a_n)_{n\in\mathbb{N}}$ be a sequence of non-negative numbers such that $a_{m+n} \leq a_m + a_n$ for all $m, n \in \mathbb{N}$ Then $\lim_{n\to\infty} a_n/n$ exists and equals $\inf_{n\in\mathbb{N}} a_n/n$.*

Proof. Let $\alpha = \inf_{n\geq 1} a_n/n$. By definition, $a_n/n \geq \alpha$ for all n. Fix $\epsilon > 0$. We aim to show that $a_n/n < \alpha + \epsilon$ for all large enough n, hence proving the lemma. We can find some k for which $a_k/k < \alpha + \frac{1}{2}\epsilon$, for if we could not then $\alpha + \frac{1}{2}\epsilon$ would be a greater upper bound than α. Then for $0 \leq j < k$ and $m \geq 1$ we have that

$$\frac{a_{mk+j}}{mk+j} \leq \frac{a_{mk} + a_j}{mk+j} \leq \frac{a_{mk}}{mk} + \frac{a_j}{mk}$$

By the hypothesis, we inductively get that $a_{mk} \leq m a_k$, and hence can obtain

$$a_{(m+1)k} = a_{mk+k} \leq a_{mk} + a_k \leq m a_k + a_k = (m+1)a_k.$$

The above holds for all k. Hence we can continue our chain of inequalities with

$$\frac{a_{mk}}{mk} + \frac{a_j}{mk} \leq \frac{m a_k}{mk} + \frac{j a_1}{mk} \leq \frac{a_k}{k} + \frac{a_1}{m} < \alpha + \frac{1}{2}\epsilon + \frac{a_1}{m}.$$

Hence if $n = mk + j$ is large enough that $a_1/m < \epsilon/2$, then $a_n/n < \alpha + \epsilon$ as required. ∎

Proposition 4.37. *If X is a shift space, then $h(X)$ exists as a limit.*

Proof. For integers $m, n \geq 1$, a block in $\mathcal{B}_{m+n}(X)$ is uniquely determined by its initial m-block and the subsequent n-block, so that

$$|\mathcal{B}_{m+n}(X)| \leq |\mathcal{B}_m(X)| \cdot |\mathcal{B}_n(X)|.$$

We may not get equality, as some blocks in $\mathcal{B}_{m+n}(X)$ may be forbidden even if their initial m letters and final n letters considered separately are allowable. Hence, applying logarithms we see that

$$\log |\mathcal{B}_{m+n}(X)| \leq \log |\mathcal{B}_m(X)| + \log |\mathcal{B}_n(X)|.$$

Applying the previous lemma to the sequence $a_k = \log |\mathcal{B}_k(X)|$ yields the result. ∎

Example 4.38. Let X be the full r-shift. Then $|\mathcal{B}_n(X)| = r^n$, so $h(X) = \log r$.

Proposition 4.39. *If Y is a factor of X, then $h(Y) \leq h(X)$.*

Proof. Let $\phi : X \to Y$ be a factor code induced by $\Phi_\infty^{[-m,k]}$, for a block map Φ and integers m and k. Then every block in $\mathcal{B}_n(Y)$ is the image under the extension of Φ of a block in $\mathcal{B}_{n+m+k}(X)$, since ϕ is surjective. Hence $|\mathcal{B}_n(Y)| \leq |\mathcal{B}_{n+m+k}(X)|$ and

$$\begin{aligned} h(Y) &= \lim_{n\to\infty} \frac{1}{n}|\mathcal{B}_n(Y)| \leq \lim_{n\to\infty} \frac{1}{n}|\mathcal{B}_{n+m+k}(X)| \\ &= \lim_{n\to\infty} \left(\frac{n+m+k}{n}\right) \frac{1}{n+m+k}|\mathcal{B}_{n+m+k}(X)| = h(X), \end{aligned}$$

since the limit of products is the product of limits. ∎

Corollary 4.40. *If X is conjugate to Y, then $h(X) = h(Y)$.*

Proof. Let $\phi : X \to Y$ be a conjugacy. Then Y is a factor of X via ϕ and X is a factor of Y via ϕ^{-1}. Hence the above proposition tells us that $h(Y) \leq h(X)$ and $h(X) \leq h(Y)$. ∎

Example 4.41. The full 2-shift has a different entropy to that of the full 3-shift, hence the two shift spaces cannot be conjugate.

Proposition 4.42. *Let $\mathcal{G} = (G, \mathcal{L})$ be a right resolving labeled graph. Then $h(X_\mathcal{G}) = h(X_G)$.*

Proof. The sliding block code induced by the labeling function, $\mathcal{L}_\infty$, is a 1-block factor code from X_G to $X_\mathcal{G}$, so by Proposition 4.39 $h(X_\mathcal{G}) \leq h(X_G)$. Since $\mathcal{G}$ is right resolving, there can be at most one presentation of a given word starting at each vertex. Hence if $\mathcal{G}$ has k states, then any block in $\mathcal{B}_n(X_\mathcal{G})$ has at most k presentations. Equivalently, there are at most k paths in G and hence at most k words in $\mathcal{B}_n(X_G)$, thus $k|\mathcal{B}_n(X_\mathcal{G})| \geq |\mathcal{B}_n(X_G)|$. Rearranging appropriately we achieve

$$\lim_{n\to\infty} \frac{1}{n} \log |\mathcal{B}_n(X_\mathcal{G})| \geq \frac{1}{k} \lim_{n\to\infty} \frac{1}{n} \log |\mathcal{B}_n(X_\mathcal{G})|,$$

and since $k \geq 1$ we have shown that $h(X_\mathcal{G}) \geq h(X_G)$. ∎

Corollary 4.43. *Let $\mathcal{G} = (G, \mathcal{L}_1)$ and $\mathcal{H} = (H, \mathcal{L}_2)$ be two right resolving labeled graphs such that their underlying graphs G and H are isomorphic. Then $h(X_\mathcal{G}) = h(X_\mathcal{H})$.*

Proof. The number of n paths on G and H are the same, hence $|\mathcal{B}_n(X_G)| = |\mathcal{B}_n(X_H)|$. By the definition of entropy, we thus get that $h(X_G) = h(X_H)$, but since Proposition 4.42 tells us that $h(X_\mathcal{G}) = h(X_G)$ and $h(X_\mathcal{H}) = h(X_H)$, we are done. ∎

Remark 4.44. In the following theorem, we use, but do not prove the result that for an irreducible shift space X and proper subshift Y of X, $h(Y) < h(X)$.

Theorem 4.45. *Let X and Y be irreducible sofic shift spaces, with $\mathcal{G}$ and $\mathcal{H}$ their respective follower set graphs and $\mathcal{G}^{min} \subseteq \mathcal{G}$ and $\mathcal{H}^{min} \subseteq \mathcal{H}$ their respective minimal right resolving graphs, found as in Theorem 4.34. If $\mathcal{G}$ and $\mathcal{H}$ are isomorphic as unlabeled graphs, then $\mathcal{G}^{min}$ and $\mathcal{H}^{min}$ are isomorphic as unlabeled graphs.*

Proof. Let G, H and G^{min}, H^{min} be the respective underlying graphs of $\mathcal{G}$, $\mathcal{H}$, $\mathcal{G}^{min}$ and $\mathcal{H}^{min}$. Now let Ω be the unlabeled isomorphism from G to H. Define H' to be $\Omega(G^{min})$, and $\mathcal{H}'$ to be the corresponding labeled subgraph of $\mathcal{H}$. We wish to show that H' is isomorphic to H^{min}.

First, we shall assume that $\mathcal{H}'$ does not present Y, and hence that $\mathcal{H}'$ presents a proper subshift of Y. Hence, using the above Corollary and stated result, we can achieve the following contradiction:

$$h(Y) = h(X_{\mathcal{H}}) = h(X_{\mathcal{G}}) = h(X_{\mathcal{G}^{min}}) = h(X_{\mathcal{H}'}) < h(Y).$$

So we have that $\mathcal{H}'$ is presentation of Y.

Moreover, $\mathcal{H}'$ is a right resolving presentation of Y. Hence showing that it is minimal will prove that it is isomorphic to $\mathcal{H}^{min}$, and the result follows. Assume that this is false, and hence $\mathcal{H}'$ has strictly more vertices than $\mathcal{H}^{min}$. Then define G' to be $\Omega^{-1}(H^{min})$ and $\mathcal{G}'$ to be the corresponding labeled subgraph of $\mathcal{G}$. By the above argument, $\mathcal{G}'$ will thus present X, but then

$$|V(\mathcal{G}')| = |V(\mathcal{H}^{min})| < |V(\mathcal{H}')| = |V(\mathcal{G}^{min})|$$

shows that $\mathcal{G}'$ is a smaller right resolving presentation of X than is $\mathcal{G}^{min}$, which is a contradiction.

■

Note that the uniqueness of the minimal presentation only gets us labeled isomorphism of $\mathcal{H}'$ and $\mathcal{H}^{min}$. However, it is entirely possible that these are two different, but isomorphic, subgraphs of $\mathcal{H}$. The following result is in fact an original result, found by the author of this paper, dealing with this issue.

Theorem 4.46. *In the above setup, we actually achieve that $\mathcal{H}' = \mathcal{H}^{min}$. Hence, since there are sets of intrinsically synchronizing words associated with each vertex of $\mathcal{G}^{min}$ and $\mathcal{H}^{min}$ via the follower set labels, Ω thus induces a canonical bijection between these sets.*

Proof. Let I be a vertex of $\mathcal{H}'$. Considering I as a vertex inside $\mathcal{H}$, we wish to show that it is labeled with the follower set of an intrinsically synchronizing word. Thus we show $\mathcal{H}' \subseteq \mathcal{H}^{min}$, but since they are of the same size, this suffices.

We can find some word $u_1 \in \mathcal{B}(Y)$ inside $\mathcal{H}'$, if $\mathcal{H}'$ is non-empty (else the result is already trivial). By Proposition 4.26, we can extend this to $u_1u_2 \in \mathcal{B}(X_{\mathcal{H}'})$ such that this word is synchronizing in $\mathcal{H}'$. By the irreducibility of $\mathcal{H}'$, we can extend this to $u_1u_2u_3 \in \mathcal{B}(X_{\mathcal{H}'})$ such that this word terminates at I and remains synchronizing in $\mathcal{H}'$ (by Lemma 4.25).

Since $\mathcal{H}'$ is a minimal right resolving presentation of $X_{\mathcal{H}}$, Lemma 4.33 says that $u_1u_2u_3$ is intrinsically synchronizing in $X_{\mathcal{H}}$. Further, he same path labeled $u_1u_2u_3$ can also be found in $\mathcal{H}$, we have that I must be labeled with the follower set of $u_1u_2u_3$. ■

5 Follower Set Graph Algorithm

5.1 Preliminaries

Here we will present an algorithm for constructing the follower set graph of a shift of finite type, given a finite set of forbidden words, under certain circumstances. First we will present the results that allow for us to determine the follower sets of a given shift within a finite number of steps.

Proposition 5.1. *Let X be an M-step shift of finite type. Then for every word $w \in \mathcal{B}(X)$ such that $|w| \geq M$, we have that $\mu w \in \mathcal{B}(X)$ if and only if $\mu w_1 \dots w_M \in \mathcal{B}(X)$, for every $\mu \in \mathcal{B}(X)$.*

Proof. The implication is trivial as $\mu w_1 \dots w_M$ is a subword of $\mu w \in \mathcal{B}(X)$. The converse is taken from the fact that if a forbidden word exists in μw, it must be of length less than or equal to M. We know the forbidden word is not in $\mu w_1 \dots w_M$, hence it must be entirely within the word w - but this would be a contradiction, since w is an allowed word by assumption. ■

The above proposition has the immediate implication that $F(\mu)$ is entirely determined by the set we will denote

$$F_M(\mu) = \{w \in \mathcal{B}_m(X) : \mu w \in \mathcal{B}(X),\ m \leq M\}.$$

That is, the follower set of μ restricted to words of length M or less. Therefore, calculating this finite set is sufficient to distinguish between the follower sets of different words. However, we still have the problem that there are potentially an infinite number of words for which we must calculate this finite set. A similar argument will solve this problem too. Note that in the following proposition, we index letters of μ in reverse order for clarity.

Proposition 5.2. *Let X be an M-step shift of finite type. Then for every word $\mu \in \mathcal{B}(X)$ of length N such that $N \geq M$, we have that $F(\mu) = F(\mu_M \dots \mu_1)$.*

Proof. A word w is in $F(\mu)$ exactly when $\mu w \in \mathcal{B}(X)$. By an argument symmetric to that of the previous proposition, this occurs if and only if $\mu_M \dots \mu_1 w \in \mathcal{B}(X)$. Again, this occurs exactly when $w \in F(\mu_M \dots \mu_1)$. ■

Hence to determine all distinct follower sets of an M-step shift X, we need only find the follower sets of the words in $\mathcal{B}(X)$ of length M or less.

5.2 Algorithm

Fix an M-step shift of finite type X, generated by a finite set of forbidden words $\mathcal{F}$ over the alphabet $\mathcal{A}$. Further, fix $\mu = \{\mu_1, \dots, \mu_n\}$ to be the set of words length at most M that do not contain a forbidden word (including the empty word). Let X^* be the set of all words not containing a forbidden word. Note that this may not necessarily be the language of X. The follower set graph of X^* is defined in the same way as the follower set graph of X, although the follower sets are defined over words that do not contain a forbidden word, as opposed to words in the language of X. The algorithm given below constructs the follower set graph of X^*, which is a presentation of X. The resulting graph coincides with the follower set graph of X when it is essential. This is because in this case we achieve $\mu \in \mathcal{B}(X)$ if and only if μ does not contain a forbidden word. The forward implication is trivial, and the reverse implication comes from the fact that we can find a path labeled μ terminating at the vertex labeled $F(\mu)$, and since the graph is essential, we can extend this to be part of a bi-infinite walk - hence μ is indeed in the language of X.

1. Form a table, indexing both rows and columns by elements of μ. For the (μ_i, μ_j)-th entry of the table, if $\mu_i\mu_j$: does not contain a forbidden word, we enter the boolean value 'True' into the table. If it does, we enter 'False'. The i-th row of this table constitutes a representations of $F_M(\mu_i)$ which is enough to determine the entire follower set of μ_i. Further, by Proposition 5.2, there are enough rows to determine all distinct follower sets of X.

2. From this table, we determine the vertices of the graph. Where two rows are equal, the two words indexing them share a follower set. Each distinct row of the table thus represents a distinct follower set - but each may have multiple indexing words. For each follower set, find the set of all indexing words and create a vertex multi-labeled with this set.

3. Although it suffices to compute $F(\mu)$ for $|\mu| \leq M$, we will also compute $F(\mu)$ for $|\mu| = M+1$ as these will become useful when drawing the edges of the graph. For each such μ, if it is forbidden do nothing. Else find the vertex with $\mu_M \dots \mu_1$ among its labels, and add μ to the set.

4. Finally, we draw the edges. Repeat the following procedure for each vertex I and each letter $a \in \mathcal{A}$. Take any word labeling the vertex with length less than or equal to M. Call it w. For each vertex J now check to see if it is labeled with wa. If it is, draw an edge from I to J labeled a and terminate the search through the remaining vertices. If no such vertex exists, then do nothing.

Example 5.3. We now illustrate how a follower set graph is constructed using the above algorithm. Let $\mathcal{A} = \{0, 1\}$ and $\mathcal{F} = \{100\}$ define a shift space X. First we form the follower table indexed by all words of length at most two.

	$\varnothing$	0	1	00	01	10	11
$\varnothing$	$\varnothing$	0	1	00	01	10	11
0	0	00	01	000	001	010	011
1	1	10	11	100	101	110	111
00	00	000	001	0000	0001	0010	0011
10	10	100	101	1000	1001	1010	1011
01	01	010	011	0100	0101	0110	0111
11	11	110	111	1100	1101	1110	1111

Next, we determine which of these words are forbidden, and indicate them with a black square.

	$\varnothing$	0	1	00	01	10	11
$\varnothing$							
0							
1				■			
00							
10		■		■	■		
01				■			
11				■			

This gives the following vertices:

$$\begin{aligned}
&F(\varnothing) = F(0) = F(00) = F(000), \\
&F(1) = F(01) = F(11) = F(001) = F(011) = F(101) = F(111), \\
&F(10) = F(010) = F(110).
\end{aligned}$$

Note that $F(100)$ is meaningless here as the word 100 is forbidden. Knowledge of these sets allows us to construct the follower set graph, displayed in Figure 7. We can see that the graph is essential and hence is indeed the follower set graph of X.

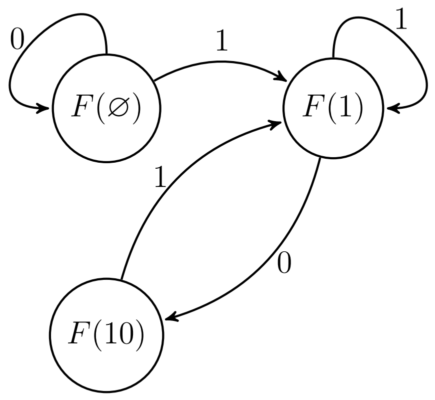


Figure 7: The constructed follower set graph for X.

Remark 5.4. In fact, this graph provides an original counterexample. Compare it to the follower set graph of the even shift, given in Example 4.11. We can see that the two graphs share an unlabeled isomorphism, but the two shifts they represent are not conjugate. In fact, from Example 3.3, we see that the even shift is not even a shift of finite type (although it is sofic) whereas our shift space X is of finite type. In fact, we also see that unlabeled isomorphism of follower set graphs does not preserve the property of being a shift of finite type. This counterexample was obtained from the implementation of the follower set graph algorithm.

Remark 5.5. We can extend this algorithm to find the minimal right resolving presentation if given an irreducible shift of finite type. The idea is as follows: take the follower set graph output by the previous algorithm. Each vertex is multi-labeled with words of length less than or equal to M whose follower sets coincide. By Proposition 5.2, words of length greater than M will only be associated with a vertex if the final M letters are also associated - hence, failure to find associated M-length words implies there are no words of length greater than M associated either.

We inspect each vertex in turn to determine if it is to be kept in the subgraph $\mathcal{H}$ described in Theorem 4.34. First, check if there exist any M-length words associated with the vertex. If there are, then they are intrinsically synchronizing by Theorem 3.4, and so must be kept in $\mathcal{H}$. Otherwise, no words of length M or greater are associated with this vertex. Next, take each remaining word v of length less than M that the vertex is labeled with. By Proposition 5.1, $uv, vw \in \mathcal{B}(X)$ implies $uvw \in \mathcal{B}(X)$ exactly when $u_1 \dots u_M v, vw_1 \dots w_M \in \mathcal{B}(X)$ implies $u_1 \dots u_M v w_1 \dots w_M \in \mathcal{B}(X)$ for $|u|, |w| \geq M$, with each statement being equivalent to v being intrinsically synchronizing. Hence, allow u and w to be every allowable word of length less than or equal to M in turn. Check whether the condition $uv, vw \in \mathcal{B}(X)$ holds, and if so we check whether $uvw \in \mathcal{B}(X)$. If this latter condition holds for every pair u and w for which the former condition holds, we have that v is intrinsically synchronizing and that the vertex should be included in $\mathcal{H}$. Else v is not intrinsically synchronizing. If we find every associated word v is not intrinsically synchronizing, then the vertex can be discarded.

6 Discussion

We began by introducing our primary objects of study, shift spaces - sets of bi-infinite sequences generated by a set of forbidden words. The appropriate morphisms between such objects turn out to be sliding block codes, induced from a fixed block map. Under our chosen metric, it turns out that shift spaces are exactly the compact and shift invariant sets of sequences, and sliding block codes are exactly the continuous and shift commuting maps. Thus bijective sliding block codes are considered the appropriate notion of equivalence, here termed conjugacy. The author believes that the topological point of view leads to clearer proofs and understanding than the initial combinatorial nature of the definitions would lead to if used directly.

Shifts of finite type are the first specific subset of shift spaces we studied - spaces generated by a finite set of forbidden words. We went on to describe graphical representations of shift spaces, where sequences are considered as bi-infinite walks on a graph. Any given graph represents some edge shift. We achieve the result that shifts of finite type are exactly the shift spaces conjugate to those that can be represented like this on an unlabeled graph.

The natural question to ask then, is what types of shift spaces can be represented by labeled graphs - where sequences are given by the labeled bi-infinite walks? There is an obvious relation here. The labeling function on the graph in fact induces a simple sliding block code from an edge shift (a shift of finite type) to this new class of shift spaces. This map is surjective by definition as every labeling of a bi-infinite walk on a graph is mapped to by the labeling function from the underlying walk. Hence we show that this class of shift spaces can be written as factor maps of shifts of finite type. We call such shifts spaces sofic shifts. In fact, we also achieve the converse - given a shift of finite type and a factor map, we can find a labeled graph presenting the image. This graph presents the factor of the shifts of finite type, hence showing that the factor is sofic. Immediately from this, we can see that the class of sofic shifts is actually the closure of the class of shifts of finite type under factor maps.

Next we explore different graphical representations of sofic shifts. One particularly interesting construction is that of the follower set graph, which can be constructed when a shift space has a finite number of distinct follower sets. It is a labeled, right resolving and "follower separated" presentation of the given shift space, constructed so that each vertex corresponds to a distinct follower set. It turns out that the shift spaces with finite follower number of distinct follower sets are exactly the sofic shifts, and thus we can find their presentation as a follower set graph.

In fact, for the subset of shifts of finite type, we give a specific algorithm for constructing such graphs and a concrete (in the programming language Haskell) implementation of the algorithm. Using this, we undertake original investigations into what properties the unlabeled ambient graph of the follower set presentation may preserve. Using the code found in the appendix, we discover that non-conjugate shifts may have follower set graphs which share unlabeled isomorphism.

Another main result of this report is that sofic shifts that can be represented on an irreducible graph have a unique (up to labeled isomorphism) minimal presentation among all right resolving presentations. In fact this is exactly the same presentation as any that is both irreducible and follower separated. We go on to detail the theory for finding this minimal presentation when given a follower set graph. Next, we introduce a measure of entropy in order to prove our final result: for irreducible sofic shifts, unlabeled isomorphism of follower set graphs implies unlabeled isomorphism of minimal representations.

While the statement of these two theorems is in many ways the culmination of this report, it should be noted that this is just an introduction to the exciting topic of Symbolic Dynamics.

Appendix A Pseudocode

Algorithm 1: Generate FollowerSetGraph(F, symbolSet)

```
Input  : $F$ = the list of forbidden words generating the language.
         $symbolSet = \{1, \ldots, d\}$.
Output: $Graph = (Nodes, Edges)$ where:
         – each node in $Nodes$ is the set of words of length less than or equal to $k+1$ which
         have the same follower set; and
         – each edge in $Edges$ is of the form (source, destination, label).
// Deal with the case that $F$ is the empty set.
if $F$ is empty then
    initialize $node$ as a list containing $symbolSet$ and the empty word
    initialize $nodes$ as a list containing only $node$
    initialize $edges$ as empty list
    foreach $letter \in symbolSet$ do
        insert $(node, node, letter)$ into $edges$
    end
    return$(nodes, edges)$
end
// If $F$ is non-empty find value of $k$ from $F$.
initialize $maxLength$ as 1
foreach $word \in F$ do
    if length of $word > maxLength$ then
        $maxLength \longleftarrow$ length of $word$
    end
end
initialize $k$ as $maxLength - 1$
// Generate $mu$, the set of allowed words of length at most $k$.
initialize $mu$ as a list containing only the empty word
foreach $n \in \{1, 2, \ldots, k\}$ do
    foreach $word \in$ set of all possible words over $symbolSet$ of length $n$ do
        if $word$ contains no element of $F$ as a subword then
            insert $word$ into $mu$
        end
    end
end
// Form tables.  Indexed by base word and concatenated word.
foreach word $a \in mu$ do
    foreach word $b \in mu$ do
        $followerTable[a][b] \longleftarrow$ concatenate $b$ with $a$
        if $followerTable[a][b]$ contains no element of $F$ as a subword then
            $truthTable[a][b] \longleftarrow True$
        else
            $truthTable[a][b] \longleftarrow False$
        end
    end
end
```

Algorithm 2: Generate FollowerSetGraph(F, symbolSet), continued

```
// Each node of the graph is a set of words sharing the same follower set.
initialize $Nodes$ as empty list;
foreach $row \in$ unique rows of $truthTable$ do
    initialize $node$ as an empty list;
    foreach word $a \in mu$ do
        if $truthTable[a] = row$ then
            insert $a$ into $node$;
        end
    end
    insert $node$ into $Nodes$;
end
// Add words of length $k+1$ to the appropriate node.
foreach $node \in Nodes$ do
    foreach $word \in node$ do
        if length of $word = k$ then
            foreach $letter \in symbolSet$ do
                $newWord \longleftarrow$ concatenate $word$ with $letter$;
                if $newWord$ contains no element of $F$ as a subword then
                    insert $newWord$ into $node$;
                end
            end
        end
    end
end
// Create edges.
initialize $Edges$ as empty list;
foreach $node \in Nodes$ do
    foreach $letter \in symbolSet$ do
        $newWord \longleftarrow$ concatenate $letter$ with any word in $node$ of length less than $k+1$;
        // Find which node $newWord$ belongs to, if any
        foreach $node' \in Nodes$ do
            if $newWord \in node'$ then
                $edge \longleftarrow (node, node', letter)$;
                insert $edge$ into $Edges$;
                break;
            end
        end
    end
end
return$(Nodes, Edges)$;
```

Appendix B Code

The following code is an implementation of the follower set graph algorithm described in section 5.2 for X^*. As noted above, if the graph is essential then it coincides with the follower set graph of X. It is written in the language Haskell. A link to the working program can be found in the references section. There are also included some extra functions which can be used to search for shifts of finite type whose follower set graphs are isomorphic when considered without labels. These makes use of the Graph Automorphism [3] library to quickly determine if two graphs are isomorphic. These functions were used in the associated paper [2].

```haskell
-- FILE: Main.hs
import FollowerSetGraph

validInput :: String -> Bool
validInput letters = all (\l -> l == '0'|| l == '1') letters

prettyPrint :: ([[String]], [(Int, Int, String)]) -> IO ()
prettyPrint (vertices, edges) = do
  putStrLn "Vertices (index, words): "
  printVertices vertices 1
  putStrLn "Edges (from, to, label): "
  printEdges edges
  putStr "\n"

printVertices :: [[String]] -> Int -> IO ()
printVertices [] iter = return ()
printVertices (vertex:vs) iter = do
  putStrLn $ (show iter) ++ " " ++ (show vertex)
  printVertices vs (iter+1)

printEdges :: [(Int, Int, String)] -> IO ()
printEdges [] = return ()
printEdges ((i,j,char):es) = do
  putStrLn $ show (i+1,j+1,char)
  printEdges es

main = do
  putStrLn "Enter forbidden words in sequence, or a blank line to generate the
    follower set graph:"
  loop []

loop :: [String] -> IO ()
loop forbiddenWords = do
  word <- getLine
  if not . validInput $ word then do
    putStrLn "Invalid input. Start over"
    loop []
  else if null word then do
    putStrLn "The graph generated is:"
    prettyPrint $ constructGraph $ forbiddenWords
    main
  else loop (word:forbiddenWords)
```

```haskell
-- FILE: FollowerSetGraph.hs
module FollowerSetGraph (
  constructGraph,
  testGraphs
)   where

import Control.Monad
import Data.List
import Data.Function (on)
import Data.Ord (comparing)
import Data.Tuple
import qualified Data.Graph as G1 (buildG)
import qualified Data.Graph.Automorphism as G2 (isIsomorphic)

-- Returns all possible words up to length n
getWords :: Int -> [String]
getWords n = concat $ map (flip replicateM "01") [0..n]

-- returns only words of length N
getWordsN :: Int -> [String]
getWordsN n = replicateM n "01"

-- Needed because [a,b,c] /= [b,c,a]
listsSameElts :: Eq a => [a] -> [a] -> Bool
listsSameElts xs ys = null (xs \\ ys) && null (ys \\ xs)

permuteZeroesAndOnes' :: String -> String
permuteZeroesAndOnes' [] = []
permuteZeroesAndOnes' (x:xs) =
  if x == '0'
    then '1' : permuteZeroesAndOnes' xs
    else '0' : permuteZeroesAndOnes' xs

permuteZeroesAndOnes :: [String] -> [String]
permuteZeroesAndOnes words = map permuteZeroesAndOnes' words

removePermutations :: [[String]] -> [[String]]
removePermutations [] = []
removePermutations (fs:fWords) =
  fs : removePermutations (filter
    (not . listsSameElts (permuteZeroesAndOnes fs)) fWords)

-- Using "tail" because we dont want the empty word included. Returns only words
   of length n
getSetsOfForbiddenWords n =
  removePermutations . tail . subsequences $ getWordsN n

isAllowed :: [String] -> String -> Bool
isAllowed forbiddenWords word = not $ any (flip isInfixOf word) forbiddenWords

getAllowableWords :: [String] -> Int -> [String]
getAllowableWords forbiddenWords n =
  filter (isAllowed forbiddenWords) $ getWords n

followerVector :: [String] -> String -> [String]
followerVector words word = map (word ++) words

type FollowerTable = [(String, [String])]
type TruthTable = [(String, [Bool])]

followerTable :: [String] -> FollowerTable
followerTable words = map (\x -> (x, followerVector words x)) words

-- Takes (key, value) pairs and returns sets of keys which shared the same value
groupSets :: (Ord b) => [(a, b)] -> [[a]]
groupSets pairs =
  map (map fst) $ groupBy ((==) `on` snd) . sortBy (comparing snd) $ pairs

testForbidden :: [String] -> FollowerTable -> TruthTable
testForbidden forbiddenWords table =
  map (\(a,b) -> (a, map (isAllowed forbiddenWords) $ b)) table

addExtraLetterToFollowerSet :: Int -> [String] -> [String] -> Char -> [String]
addExtraLetterToFollowerSet k forbiddenWords mus letter =
  (filter (isAllowed forbiddenWords) longerMus)
    where longerMus =
      map ([letter] ++) . filter (\mu -> length mu == k)  $ mus

addExtraLettersToFollowerSets :: Int -> [String] -> [[String]] -> [[String]]
addExtraLettersToFollowerSets k fWords nodes =
  map (\node ->
        node ++ (addExtraLetterToFollowerSet k fWords node '0') ++
        (addExtraLetterToFollowerSet k fWords node '1'))  nodes

-- Node is a list of words which are to be 'followed'. Ie the nodes on the graph
getNodes :: Int -> [String] -> [[String]]
getNodes k forbiddenWords =
  addExtraLettersToFollowerSets k forbiddenWords . groupSets . testForbidden
    forbiddenWords . followerTable . getAllowableWords forbiddenWords $ k

-- Index of node source, index of node dest, label (0 or 1)
type MaybeEdge = (Maybe Int, Maybe Int, String)
type Edge = (Int, Int, String)

createEdge :: String -> [String] -> [[String]] -> MaybeEdge
createEdge add node nodes = (elemIndex node nodes, findIndex (elem addedLetter)
    nodes, add)
  where addedLetter = (head node) ++ add

eliminate :: [MaybeEdge] -> [Edge]
eliminate ((Just x, Just y, z):list) = (x, y, z):eliminate list
eliminate ((Just x, Nothing, z):list) = eliminate list
eliminate [] = []

getEdges :: [[String]] -> [Edge]
getEdges nodes =
  eliminate $ foldr
    (\node acc -> createEdge "0" node nodes : createEdge "1" node nodes : acc) []
    nodes

```

```haskell
-- Pairs of (node, edge) where node is a list of follower sets associated with
   that node
type Graph = ([[String]], [Edge])

constructGraph :: [String] -> Graph
constructGraph forbiddenWords =
  (nodes, edges)
  where
    k = (maximum $ map length forbiddenWords) - 1
    nodes = getNodes k forbiddenWords
    edges = getEdges nodes

-- Returns list of labels coming from each node. Nodes come in already sorted by
   first element
getLabelledEdgeSetSrc :: Graph -> [String]
getLabelledEdgeSetSrc g =
  sort . map
    (\edges -> concatMap tripleTrd edges) . groupBy ((==) `on` tripleFst) $
    (snd g)

checkLabelledEdgeSetsSrc :: Graph -> Graph -> Bool
checkLabelledEdgeSetsSrc g1 g2 =
  getLabelledEdgeSetSrc g1 == getLabelledEdgeSetSrc g2

-- Returns list of labels going to each node
getLabelledEdgeSetDest :: Graph -> [String]
getLabelledEdgeSetDest g =
  sort . map
    (\edges -> concatMap tripleTrd edges) . groupBy ((==) `on` tripleSnd) .
    sortBy (comparing tripleSnd) $ (snd g)

checkLabelledEdgeSetsDest :: Graph -> Graph -> Bool
checkLabelledEdgeSetsDest g1 g2 =
  getLabelledEdgeSetDest g1 == getLabelledEdgeSetDest g2

-- returns (index, #emmitting)
getVertexSrc :: Graph -> [(Int, Int)]
getVertexSrc g =
  map (\edges -> (tripleFst . head $ edges, length edges)) .
  groupBy ((==) `on` tripleFst) . snd $ g

-- returns (index, #edges going to index)
getVertexDest :: Graph -> [(Int, Int)]
getVertexDest g =
  sortBy (comparing fst) .
  map (\edges -> (tripleSnd . head $ edges, length edges)) .
  groupBy ((==) `on` tripleSnd) .
  sortBy (comparing tripleSnd) . snd $ g

-- Takes (index, #emitting) and (index, #edges going to index) and returns the
   sorted list of number of edges at each node (ignoring direction)
sumEdges :: [(Int, Int)] -> [(Int, Int)] -> [(Int, Int)]
sumEdges [] ys = ys
sumEdges xs [] = xs
sumEdges (x:xs) (y:ys) =
  if fst x == fst y
    then (fst x, snd x + snd y) : sumEdges xs ys
    else (if fst x > fst y
      then y : sumEdges (x:xs) ys
      else x : sumEdges xs (y:ys))

-- Returns sorted number of edges attached to each vertex
getUnlabelledEdgeSet :: Graph -> [Int]
getUnlabelledEdgeSet g =
  sort . map snd $ sumEdges (getVertexSrc g) (getVertexDest g)

checkUnlabelledEdgeSet g1 g2 = getUnlabelledEdgeSet g1 == getUnlabelledEdgeSet g2

getNumberCycles :: Graph -> Int
getNumberCycles g = length . filter (\(src, dest, lab) -> src == dest) $ snd g

checkCycles :: Graph -> Graph -> Bool
checkCycles g1 g2 = getNumberCycles g1 == getNumberCycles g2

checkNumberVertices g1 g2 = length (fst g1) == length (fst g2)

graphsCouldBeTheSame :: Graph -> Graph -> Bool
graphsCouldBeTheSame g1 g2 =
  (checkNumberVertices g1 g2) && (checkCycles g1 g2) &&
  (checkUnlabelledEdgeSet g1 g2) && (checkLabelledEdgeSetsSrc g1 g2) &&
  (checkLabelledEdgeSetsDest g1 g2)

-- Returns (fWords, graph) pairs
createGraphsOfKValue :: Int -> [([String], Graph)]
createGraphsOfKValue k =
  map (\set -> (set, constructGraph set)) $ getSetsOfForbiddenWords (k+1)

-- Double check this is for digraphs and not just graphs?
myIsIsomorphic :: Graph -> Graph -> Bool
myIsIsomorphic g1 g2 = G2.isIsomorphic g1' g2'
  where
    e1 = map (\(a,b,c) -> (a,b)) (snd g1)
    e2 = map (\(a,b,c) -> (a,b)) (snd g2)
    bounds1 = (0, length . fst $ g1)
    bounds2 = (0, length . fst $ g2)
    g1' = G1.buildG bounds1 e1
    g2' = G1.buildG bounds2 e2

-- takes x = (forbidden word set, graph) pair and compares against another list
   of graphs. Returns the sets of forbidden words that give isomorphic graphs to
   that inside x - including the forbidden words generating x itself
testOneGraphAgainst :: ([String], Graph) -> [([String], Graph)] -> [[String]]
testOneGraphAgainst x [] = [fst x]
testOneGraphAgainst x (y:ys) =
  if graphsCouldBeTheSame (snd x) (snd y) && myIsIsomorphic (snd x) (snd y)
    then (fst y) : testOneGraphAgainst x ys
    else testOneGraphAgainst x ys

-- Takes a list of (fWords, graph) pairs and reutrns a list of sets of fWords
   that generate similar looking graphs
```

```haskell
test :: [([String], Graph)] -> [[[String]]]
test [] = []

-- Dont test elements already noted to generate similar graphs to previous ones
test (x:xs) =
  similarForbiddenSets :
  (test $ filter (\(fs,g) -> not . flip elem similarForbiddenSets $ fs) xs)
    where similarForbiddenSets = testOneGraphAgainst x xs

testGraphs k =
  filter (\fs -> length fs > 1) . test $ (concatMap createGraphsOfKValue [1..k])

-- Helper functions
tripleFst :: (a,b,c) -> a
tripleFst (a,b,c) = a

tripleSnd :: (a,b,c) -> b
tripleSnd (a,b,c) = b

tripleTrd :: (a,b,c) -> c
tripleTrd (a,b,c) = c
```

References

[1] C. Barrett, *Haskell code and implementation of follower set graph algorithm*, http://www.mas.ncl.ac.uk/∼nek29/papers.html (item 21a).

[2] C. Barrett and E. T. A. Kakariadis, *On the quantized dynamics of factorial languages*, preprint (arXiv: 1611.06844).

[3] J. Bernardy, *Haskell graph automorphism library*, https://hackage.haskell.org/package/hgal-2.0.0.2/docs/Data-Graph-Automorphism.html (2008).

[4] R. Fischer, *Graphs and symbolic dynamics*, Colloq. Math. Soc. Janos Bolyai: Topics in Information Theory (1975).

[5] N. Jonoska, *Sofic shifts with synchronizing presentations*, Theor. Computer Sci. (1995).

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

[7] R. Pavlov, *Symbolic Dynamics course webpage*, http://web.cs.du.edu/∼rpavlov/3705/math3705.html, University of Denver B.

[8] B. Weiss, *Subshifts of finite type and sofic systems*, Monatsh. Math. (1973), 462–474.