Symbolic Dynamics

Author: Chris Barrett

Supervisor: Evgenios Kakariadis

School of Mathematics and Statistics

Newcastle University

Introduction

Although originally arising as a way to encode and study dynamical systems, the techniques and ideas of Symbolic Dynamics 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. This is because 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, if two successive 1's are too distant, the clocking circuit measuring the time between them may drift far enough to give an inaccurate reading of how many 0's came in-between. 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. This is Symbolic Dynamics [3].

This project involved research into the field of Symbolic Dynamics, along with the implementation of an algorithm to find graphical representations of shift spaces [1], and has an associated research paper [2], which contains original work and makes use of the implementation.

Shift Spaces and Sliding Block Codes

The fundamental elements of study in Symbolic Dynamics are bi-infinite sequences of symbols, drawn from a finite alphabet such as $\{0, 1, \dots, 9\}$ or $\{a, b, \dots, z\}$. Such a sequence is denoted by $x = (x_i)_{i\in\mathbb{Z}}$, or by

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

A *word* or *block* is a finite sequence of letters.

Our main objects of study are **shift spaces** - defined to be the set of all sequences that do *not* contain any forbidden words, taken over a given alphabet. Different sets of forbidden words can generate distinct shift spaces. An important property of these spaces is that they are invariant when every sequence is shifted one place forward; hence their name.

Where shift spaces are the objects of study, **sliding block codes** are the appropriate morphisms between them. To define such a code $\phi : X \to Y$, first take a fixed block map Φ from words of length N in X to letters of Y. This then induces the sliding block code

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

where *memory* m and *anticipation* n are such that $m + n + 1 = N$. The entry $\phi(x)_i$ depends on a "window" of coordinates around x_i and $y = \phi(x)$ is given by "sliding" this window along one coordinate at a time.

$$\begin{array}{ccc} x & \dots x_{-m-1}\boxed{x_{-m}\dots x_i\dots x_n}x_{n+1}\dots. \\ \phi\big(\big\downarrow & \big\downarrow\Phi \\ y & \dots y_{i-1}\boxed{y_i}y_{i+1}\dots. \end{array}$$

With sliding block codes as our morphisms, it makes sense that sliding block codes with a sliding block inverse are our notion of **conjugacy**.

Shifts of Finite Type

A subset of shift spaces are **shifts of finite type** - those that can be generated by a finite set of forbidden words. Note that in general, a shift of finite type may *also* be generated by an infinite set of words. Shifts of finite type are closed under conjugacy.

We say the **edge shift** is the set of all bi-infinite walks over an unlabeled graph . That is, let G be a graph with the edge set E. Then the corresponding edge shift is the shift space over the alphabet $\mathcal{A} = E$ given by $\{(\xi_i)_{i\in\mathbb{Z}} \in E^{\mathbb{Z}} : t(\xi_i) = i(\xi_{i+1}) \text{ for all } i \in \mathbb{Z}\}$.

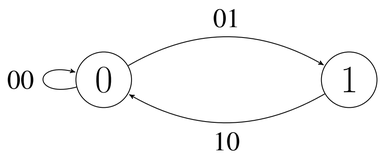

Figure 1: Representation of a conjugacy of the golden mean shift, generated by $\mathcal{F} = \{11\}$.

Theorem. *Every shift of finite type is conjugate to an edge shift.*

This is illustrated above - the golden mean shift is over the binary alphabet, meaning there will be two edges in the graph. We can see by elimination that the original shift has no representation as an edge shift, however the above graph represents a shift conjugate to it. The elements of the alphabet here are the names of the three edges. By considering only the first letter of each edge, we get back the golden mean shift.

Sofic Shifts

The natural question to ask next is what shift spaces can be represented on labeled graphs, and what is their relation to shifts of finite type? Say that shift spaces that can be represented on a labeled graph are called **sofic shifts**. Here, we will take the shift space represented to be the labels assigned to bi-infinite sequences of walks on our graph. The labeling function $\mathcal{L}$ from the edge set of the underlying graph to the new alphabet, induces a sliding block code $\mathcal{L}_\infty$ from the edge shift of the underlying graph to the sofic shift on the labeled graph. Since $\mathcal{L}_\infty$ is surjective by definition, we see that sofic shifts can be written as the image of a shift of finite type under a surjective sliding block code. In fact, the converse is also true.

Theorem. *Sofic shifts are exactly the closure of shifts of finite type under surjective sliding block codes.*

Follower Set Graphs

In a given shift space, the **follower set** of a word in is defined as the set of all words which follow it in some sequence, i.e. if we let X be a shift space and w be a word appearing in some sequence of X, then the follower set is defined as the set of words

$$F(w) = \{v \text{ appears in } X : wv \text{ appears in } X\}.$$

It turns out that sofic shifts are exactly those with a finite number of follower sets. There can be an infinite number of words allowed in a shift space, however many words can also share the same follower set. A simple example is if we take X to be all possible sequences: $F(w)$ is the same for every w.

We will see that if we know the follower sets of a sofic shift, we can construct the **follower set graph** - a labeled presentation of the shift space. It is defined as follows: suppose X is a shift space over $\mathcal{A}$. Set the vertices to be the distinct follower sets of X - representing the equal follower sets of many different words. To draw the edges, take each vertex in turn. Call it v_1 and take $F_X(w)$ to be a follower set it represents. For each $a \in \mathcal{A}$, check if wa is forbidden. If so, do nothing - otherwise, there exists a follower set $F_X(wa)$ which corresponds to another vertex - call it v_2. In this case, construct an edge from v_1 to v_2, labeled as a.

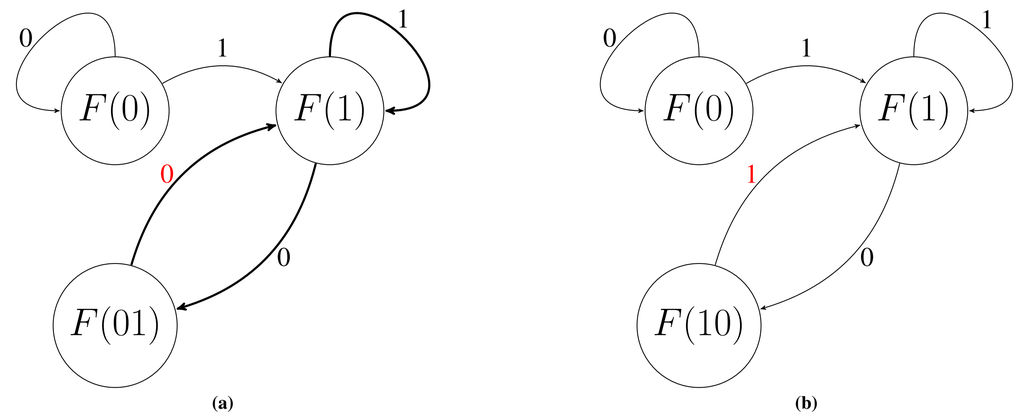

Figure 2: Follower set graphs - an example of labeled graphs presenting shift spaces

The above figure presents (a), the even shift - a binary sofic shift (not of finite type) given by the set of forbidden words $\{10^{2n+1} : n \in \mathbb{N}\}$. That is, each consecutive 1 is separated by an even number of 0's.

Figure (b) presents the binary shift of finite type generated by the set $\mathcal{F} = \{100\}$. This graph was generated by the algorithm that was implemented in this project. Note the two figures have isomorphic underlying graphs, although the red label is different so they are not labeled isomorphic.

Algorithm

During this project, we constructed and implemented an algorithm which, given a finite set of forbidden words, generates the follower set graph of the corresponding shift of finite type.

The reason this algorithm terminates is due to the following result. Let X be a shift of finite type, generated by a set of forbidden words of length at most M. For every word w and μ that appears in X, we have that μw appears in X if and only if $\mu w_1 \dots w_M$ appears in X.

Hence, we only need to check words of length up to M to determine the follower sets fully, which is the main challenge. In [2], we give a specific algorithm for constructing such graphs, along with both a conceptual (in pseudocode) and a concrete (in the programming language Haskell) implementation of the algorithm.

```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
```

Minimal Presentations and Main Results

We give two definitions. First, a labeled graph is **right resolving** if every edge coming out of a given vertex has a distinct label; for example, follower set graphs. Second, for a sofic shift space to be **irreducible**, it must have a presentation on an irreducible graph - i.e. any vertex can be reached from any other vertex.

Theorem. *For irreducible sofic shift spaces, there exists a unique (up to labeled isomorphism)* ***minimal right resolving presentation****, with respect to the number of vertices.*

The figure below (originally found by Jonoska [4]) shows that the property of irreducibility is necessary for this theorem to hold. Note there are no presentations of the shift space presented below using fewer than 4 vertices.

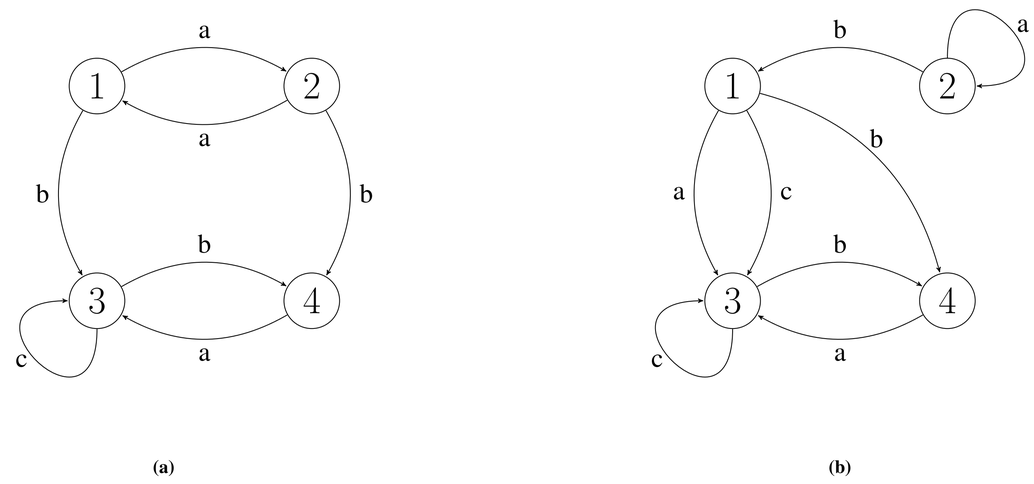

Figure 3: Two reducible, minimal right resolving graphs presenting the same shift space

A central result of our work is that the minimal presentation of an irreducible sofic shift space can be found as a subgraph of its follower set graph. We can see that in figure 2(a), the minimal presentation is given by the bolded edges and attached vertices. Figure 2(b) presents a reducible shift and hence this theorem does not apply.

Assuming an irreducible shift space X, a word v is **intrinsically synchronizing** in X if whenever uv is allowed and vw is allowed, then uvw is also allowed. The relevent subgraph is that consisting only of the vertices corresponding to follower sets of intrinsically synchronizing words, and the edges between them. The following is an original result taken from our associated paper.

Theorem. *Let two irreducible sofic shifts be such that their follower set graphs share an unlabeled isomorphism. Then the isomorphism preserves their minimal presentation.*

The two graphs in figure 2 show that irreducibility is also necessary for the above theorem, as the subgraph of (b) given by the vertices labeled $F(1)$ and $F(10)$ no longer presents the same shift. During our work on this project we found a number of useful counterexamples like this. The same two graphs also show that unlabeled isomorphism of the follower set graph does not preserve the property of being a shift of finite type.

While the statement of these two theorems is in some ways the culmination of this research project, it should be remembered that this is simply an introduction to the large topic of Symbolic Dynamics.

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

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