Towards algorithms for reducible presentations of sofic shifts

by Justin Cai

April 2020

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in the Department of Computer Science
at the University of Colorado Boulder

Abstract

Given an irreducible presentation of a sofic shift, it is well known that there is a procedure that yields a presentation with the fewest vertices among all presentations of the shift in polynomial time. However, given a reducible presentation, the previous procedure does not necessarily yield such a minimal presentation. If the reducible presentation presents an irreducible shift, then an irreducible presentation that presents the whole shift itself can be found within this presentation, so one could minimize these by these presentations by finding this component within the presentation. However, without knowing in advance if presentation presented an irreducible shift or not, this procedure would not work, so an algorithm for testing that is desirable. In this thesis, we present progress towards such an algorithm.

Acknowledgements

Raf, if I didn't take your theory of computation course, I probably would've never gotten to the the level of appreciation and joy for mathematics and theoretical computer science I have now (or at least taken a lot longer), so thank you for your class, introducing me to symbolic dynamics, being willing to advise me, and being a great advisor overall.

Jem and Josh, thank you for volunteering your time to be on my committee. It is much appreciated!

STARBURTS[1], thanks for being a great bunch of friends this semester (and for listening to me talk about theory of computation for three hours).

[1] **S**uper **T**errific **A**wesome **R**adical **B**rilliant **U**rban **R**eally **S**emantic **T**echnical **S**uperstars

Contents

Chapter 1

Introduction

In an effort to to sharpen the dividing line between classical differential analysis and abstract symbolic analysis used frequently in the study of recurrence and transitivity in dynamical systems, Morse and Hedlund named the field of symbolic dynamics in their eponymous 1938 paper [MH38]. Shift spaces (or simply shifts), the main object of study in symbolic dynamics, are sets of unending sequences over a finite set of symbols in which certain forbidden finite sequences (know as forbidden words) are not allowed to appear in the unending sequences. If a shift space has a finite set of forbidden words that describe it, then the shift space is known as a shift of finite type (SFT). In addition to being characterized by a finite list of forbidden words, SFTs also have a representation in a graphical form, in the sense that all the unending sequences of SFTs can be described by reading off the vertices in an infinite walk around a graph. Sofic shifts were introduced by [Wei73] as the smallest class of shift spaces containing every SFT and which were closed under factor codes. However, it was clear that sofic shifts were exactly the shifts described by infinite walks around a labeled graph.[1]

The labeled graphs that described sofic shifts, referred to hereafter as presentations of a sofic shift, are not unique, in the sense that many different presentations can present the same sofic shift. Fischer [Fis75] was interested in irreducible sofic shifts (what he called transitive), and shows that such sofic shifts admit a unique minimal deterministic presentation (with respect to number of vertices in the presentation). [JM94] further characterized these minimal presentations. Sofic shifts

[1]The word sofic is derived from the Hebrew word for finite [Lin+95], which an apt name as sofic shifts generalize shifts of *finite* type while still having a finite representation though a labeled graph. Additionally, a shift space is sofic iff it has a finite number of follower sets, which is closely related to the Myhill-Nerode theorem from automata theory.

that are reducible (i.e. not irreducible) do not necessarily admit a unique minimal presentation, which can be seen in [Jon96], where two presentations of a SFT were given that had the smallest number of vertices among all presentation of the SFT, but the presentations were distinct. In the same paper, Jonoska characterized sofic shifts that had synchronizing deterministic presentations in terms of the syntactic monoid of the language of the shift, and showed that any sofic shift having a synchronizing deterministic presentation necessarily had a unique minimal one.

The only algorithms that exist for minimizing presentations of sofic shift seem to be a well-known procedure that is essentially based on DFA minimization [HMU01; Lin+95]. However, this algorithm is only guaranteed to yield the minimal presentation when given an irreducible presentation (where irreducibility of a presentation is a property of the presentation being "connected"). Irreducible presentations present irreducible shifts, but reducible presentations do not necessarily present reducible shifts (i.e. a reducible presentation can present an irreducible shift). It has been essentially noted in [Jon96; MR91; JM94] that presentations that present irreducible sofic shifts contain an irreducible subgraph which present the whole shift itself. Therefore, if you you knew that a presentation presented an irreducible shift, a procedure to yield the minimal presentation would be to select the irreducible subgraph that presents the shift, and then further minimizing that (if necessary). However, an algorithm for deciding whether any presentation presents an irreducible shift seems to be mentioned nowhere in the literature. In this thesis, we work towards such an algorithm.

Chapter 2

Preliminaries

In this section we introduce basic concepts from symbolic dynamics. Definitions and notation follow [Lin+95] closely.

Definition 2.1. Let $\mathcal{A}$ be a finite set. The *full $\mathcal{A}$-shift* is the set $\mathcal{A}^{\mathbb{Z}}$ of all bi-infinite sequences over $\mathcal{A}$ (i.e. functions from $\mathbb{Z}$ to $\mathcal{A}$, hence the usual notation for the set of all functions from $\mathbb{Z}$ to $\mathcal{A}$). Elements in $\mathcal{A}^{\mathbb{Z}}$ are called *points*.

A *word* is a finite sequence of letters over some alphabet $\mathcal{A}$. We use ϵ to denote the empty word. The set of all words over $\mathcal{A}$ is denoted $\mathcal{A}^*$. Note that $\epsilon \in \mathcal{A}^*$; the set of all nonempty words over $\mathcal{A}$ is denoted $\mathcal{A}^+$ (so $\epsilon \notin \mathcal{A}^+$). Let $x = (x_i)_{i\in\mathbb{Z}}$ be a bi-infinite sequence. For $i \leq j$, the word from the ith coordinate to the jth coordinate is denoted

$$x_{[i,j]} \triangleq x_i x_{i+1} \dots x_j.$$

Definition 2.2. Let $\mathcal{F}$ be a set of words over an alphabet $\mathcal{A}$, called the *forbidden words*. A *shift space* (or simply *shift*) is a subset $\mathsf{X}_{\mathcal{F}}$ of some full shift $\mathcal{A}^{\mathbb{Z}}$ such that none of the forbidden words appear in any point of the shift space. That is,

$$\mathsf{X}_{\mathcal{F}} \triangleq \left\{ (x_i)_{i\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}} \,:\, \forall i, j \in \mathbb{Z}, i < j \quad x_{[i,j]} \notin \mathcal{F} \right\}.$$

If X and Y are shift spaces and $X \subseteq Y$, then we say X is a *subshift* of Y. We can see that $\mathcal{A}^{\mathbb{Z}}$ is a shift space by taking $\mathcal{F} = \varnothing$, so as every shift space is a subset of a full shift, so shift spaces are sometimes synonymously referred to as subshifts.

Definition 2.3. Let X be a shift space. The *language of X* is the set

$$\mathcal{B}(X) = \{x_{[i,j]} : x \in X, i, j \in \mathbb{Z}, i < j\}$$

of nonempty words that appear in some point in X.

Instead of specifying what words are forbidden in a shift space, we can characterize shift spaces by their languages. If $L \subseteq \mathcal{A}^+$ is a set of nonempty words, then we say L is *factorial* if for every word $w \in L$, then every nonempty subword is in L. We say L is *prolongable* if there for every word $w \in L$, there are nonempty words $u, v \in L$ such that $uwv \in L$.

Theorem 2.4 ([Lin+95])**.** If $L \subseteq \mathcal{A}^+$ is a set of nonempty words, then $L = \mathcal{B}(X)$ for some shift space X if and only if L is factorial and prolongable. Furthermore, for any shift space, $X = \mathsf{X}_{\mathcal{A}^+ \setminus \mathcal{B}(X)}$, so two shift spaces are equal if and only if their languages are equal.

The following theorem will be useful for our discussion and is well known, but no reference was found for it, so we provide a proof.

Theorem 2.5. If X and Y are shift spaces, then $X \subseteq Y$ if and only if $\mathcal{B}(X) \subseteq \mathcal{B}(Y)$.

Proof. Suppose $X \subseteq Y$. If $w \in \mathcal{B}(X)$, then w occurs in some point $x \in X$. But $x \in Y$ as $X \subseteq Y$, so w occurs in some point in Y. Therefore, $w \in \mathcal{B}(Y)$.

Conversely, suppose $\mathcal{B}(X) \subseteq \mathcal{B}(Y)$. If $x \in X$, then every word occurring in x is in $\mathcal{B}(X)$. But $\mathcal{B}(X) \subseteq \mathcal{B}(Y)$, so every word occurring in x is in $\mathcal{B}(Y)$, so $x \in Y$. □

Definition 2.6. Let X be a shift space. If $u, v \in \mathcal{B}(X)$ and there is a word $w \in \mathcal{B}(X)$ such that $uwv \in \mathcal{B}(X)$, then we say w joins u and v. If for every pair of words in $u, v \in \mathcal{B}(X)$ there is a word w joining u and v, then we say X is *irreducible*. Otherwise, we say X is *reducible*.

Example 2.7. A typical shift space we will see is the *even shift*. We define the even shift to be the set of bi-infinite sequences over $\{0, 1\}$ such that between any two 1's, there is an even number of 0's. That is, the forbidden blocks are $\{10^{2k+1}1 : k \geq 0\}$. The even shift is also irreducible, which can be easily seen after we introduce presentations of sofic shifts.

Another shift space is the 010-*shift*, which are the bi-infinite sequences over $\{0, 1\}$ such that 1 only appears at most once. We can describe the shift using the forbidden blocks $\{10^k1 : k \geq 0\}$. This shift space is reducible, as 1 is in the language, but if there were a word joining 1 and 1, the the the resulting word would contain a forbidden word, so there can be no word joining 1 and 1.

Definition 2.8. A *graph*[1] G is a 4-tuple $G = (\mathcal{V}, \mathcal{E}, i, t)$, where $\mathcal{V}$ is a finite set of *vertices*, $\mathcal{E}$ is a finite set of *edges*, and $i : \mathcal{E} \to \mathcal{V}$ and $t : \mathcal{E} \to \mathcal{V}$ are functions assigning an

[1] In our discussion, a graph really means a directed multigraph; multiple edges between the same two vertices and self loops are permitted.

initial and *terminating* vertex for each edge, respectively. For an arbitrary graph G, let $\mathcal{V}_G$, $\mathcal{E}_G$, i_G, and t_G denote the graph's vertices, edges, and initial and terminating vertex functions, respectively.

Definition 2.9. Let G be a graph. A *bi-infinite walk in* G is a bi-infinite sequence x over the edges of G such that $t(x_i) = i(x_{i+1})$ for all $i \in \mathbb{Z}$. The set X_G of all bi-infinite walks on G (called the *edge shift*) is denoted

$$\mathsf{X}_G \triangleq \left\{ (x_i)_{i\in\mathbb{Z}} \in \mathcal{E}^{\mathbb{Z}} \,:\, \forall i \in \mathbb{Z} \quad t(x_i) = i(x_{i+1}) \right\}.$$

Theorem 2.10 ([Lin+95])**.** If G is a graph, then X_G is a shift space.

Let G be a graph. A *path in* G is a nonempty finite sequence of edges $\pi = e_1 \dots e_n$ such that $t(e_i) = i(e_{i+1})$ for all $i < n$. If I is a vertex in G, we say a path *starts at* I if $i(e_1) = I$. Similarly, we say a path *ends at* I if $t(e_n) = I$.

Notice that every word in $\mathcal{B}(\mathsf{X}_G)$ is a path G, but every path in G is not necessarily a word in $\mathcal{B}(\mathsf{X}_G)$. Call a vertex *stranded* if there is no edge starting at I or ending at I. No bi-infinite walk can go through a stranded vertex, because a bi-infinite path must always have a "next" or "previous" vertex, and stranded vertices are exactly the vertices which have none. If G has no stranded vertices, then we call G *essential*. Every non-essential graph can be made essential by removing the stranded vertices, and the resulting graph will have still have the same edge shift. Then, if G is essential, then every path in G is necessarily a word in $\mathcal{B}(\mathsf{X}_G)$. Thus, it simplifies our discussion to work with essential graphs, as it allows us to paths and words in the language of an edge shift synonymously.

If G is a graph and I and J are vertices in G, then we say J *is reachable from* I if there is a path starting at I and ending at J. We say G is *irreducible* if, for every pair of vertices I, J, I is reachable from J. Otherwise, we say G is *reducible*. Every irreducible graph is necessarily essential. If G is irreducible, then it it not too hard to see that X_G is irreducible. However, as we will discuss in the next section X_G is not necessarily reducible if G is reducible.

Definition 2.11. A *labeled graph* $\mathcal{G}$ is a pair $(G, \mathcal{L})$, where G is a graph and $\mathcal{L} : \mathcal{E} \to \mathcal{A}$ is the *labeling function* from the edges of G onto some finite alphabet $\mathcal{A}$. We refer to G as the *underlying graph*. As we did with graphs, if $\mathcal{G}$ is an arbitrary labeled graph, then let $\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, i_\mathcal{G}$, and $t_\mathcal{G}$ denote the vertices, edges, initial and terminating vertex functions of the underlying graph. Additionally, let $\mathcal{L}_\mathcal{G}$ denote the labeling function and $\mathcal{A}_\mathcal{G}$ denote the set of labels appearing in $\mathcal{G}$ (i.e. the image of $\mathcal{L}_\mathcal{G}$).

If $\mathcal{G}$ is a labeled graph, then we will let $\mathcal{G}$ inherit properties of its underlying graph. Specifically, we say $\mathcal{G}$ is essential if its underlying graph is essential, $\mathcal{G}$ is

irreducible if its underlying graph is irreducible, $\mathcal{G}$ is reducible if its underlying graph is reducible, and so on.

Definition 2.12. Let $\mathcal{G} = (G, \mathcal{L})$ be a labeled graph. If x is a bi-infinite walk in G, then the *label of* x is the bi-infinite sequence $(\xi_i)_{i\in\mathbb{Z}}$, where $\xi_i = \mathcal{L}(x_i)$ for all $i \in \mathbb{Z}$. The set of all the labels of bi-infinite walks in G is denoted

$$\mathsf{X}_{\mathcal{G}} \triangleq \left\{ (\xi_i)_{i\in\mathbb{Z}} : \xi_i = \mathcal{L}(x_i), x \in \mathsf{X}_G \right\}.$$

Theorem 2.13 ([Lin+95])**.** If $\mathcal{G}$ is a labeled graph, then $\mathsf{X}_{\mathcal{G}}$ is a shift space.

If X is a shift space and $X = \mathsf{X}_{\mathcal{G}}$ for some labeled graph $\mathcal{G}$, then we say X is a *sofic shift*. We say $\mathcal{G}$ *presents* X, and will also refer to $\mathcal{G}$ as a *presentation of* X.

Let $\mathcal{G}$ be a labeled graph. If $\pi = e_1 \dots e_n$ is a path in $\mathcal{G}$, then we will abuse notation and define the *label of the path* $\mathcal{L}(\pi) \triangleq w$, where $w = \mathcal{L}(e_1)\dots\mathcal{L}(e_n)$, and say that π *presents* w. Similar to graphs, every word in $\mathcal{B}(\mathsf{X}_{\mathcal{G}})$ is presented by some path in $\mathcal{G}$ but every word presented by a path in $\mathcal{G}$ is not necessarily a word in $\mathcal{B}(\mathsf{X}_{\mathcal{G}})$, unless $\mathcal{G}$ is essential.

Example 2.14. The even shift and 010-shift are both examples of sofic shifts, as we can describe their points as bi-infinite walks on labeled graphs. The even shift is presented by Figure 2.1a and the 010-shift is presented by Figure 2.1b. As the underlying graph of Figure 2.1a is irreducible, the even shift itself is irreducible.

Definition 2.15. Let $\mathcal{G} = (G, \mathcal{L})$ be a presentation of a sofic shift X. For a vertex $I \in \mathcal{V}_{\mathcal{G}}$, the *follower set of* I is the set

$$F_{\mathcal{G}}(I) \triangleq \left\{ \mathcal{L}(\pi) : \pi \in \mathcal{B}(\mathsf{X}_G), i(\pi) = I \right\}.$$

For a word $w \in \mathcal{B}(X)$, the *follower set of* w is the set

$$F_X(w) \triangleq \left\{ u \in \mathcal{B}(X) : wu \in \mathcal{B}(X) \right\}.$$

Let G and H be graphs. A *graph isomorphism between* G *and* H is a pair of bijective functions $\partial\Phi : \mathcal{V}_G \to \mathcal{V}_H$ and $\Phi : \mathcal{E}_G \to \mathcal{E}_H$ such that for all $e \in \mathcal{E}_G$, we have $\partial\Phi(i_G(e)) = i_H(\Phi(e))$ and $\partial\Phi(t_G(e)) = t_H(\Phi(e))$; i.e. G and H are the same graphs up to renaming the vertices and edges. Similarly, if $\mathcal{G}$ and $\mathcal{H}$ are labeled graphs, then a *labeled-graph isomorphism from* $\mathcal{G}$ *to* $\mathcal{H}$ is a graph isomorphism $(\partial\Phi, \Phi)$ between the underlying graphs of $\mathcal{G}$ and $\mathcal{H}$ such that $\mathcal{L}_{\mathcal{H}}(\Phi(e)) = \mathcal{L}_{\mathcal{G}}(e)$ for all $e \in \mathcal{E}_{\mathcal{G}}$. We say $\mathcal{G}$ and $\mathcal{H}$ are *isomorphic* if there is an isomorphism between them.

A *minimal presentation* of a sofic shift X is a presentation of X with the smallest number of vertices among all presentations of X. We say a presentation is *deterministic* if for each vertex in the presentation, no two distinct edges starting at

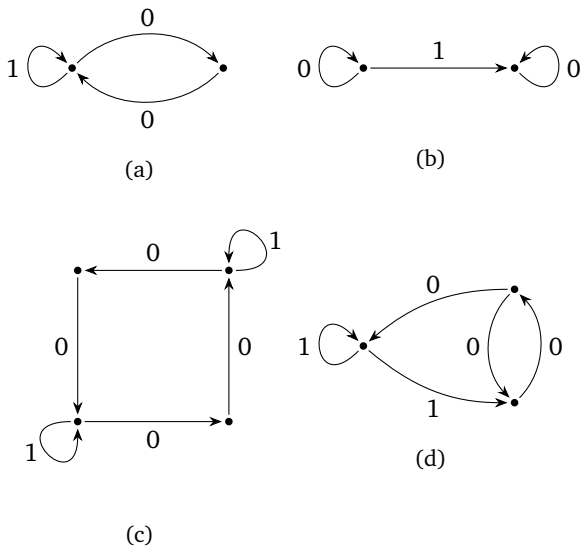

Figure 2.1: Presentations of sofic shifts.

that vertex share the same label (i.e. the edges starting at that vertex are labeled uniquely). A presentation is *follower-separated* no pair of distinct vertices have the same follower sets (i.e. all the follower sets of the presentation are distinct).

Every sofic shift has a deterministic presentation, which borrows the idea of the *subset construction* from automata theory [HMU01]. In fact, [Jon96] showed that the minimal *deterministic finite automata* (DFA) accepting the language of a sofic shift has the minimal presentation for the sofic shift as a subgraph of the DFA.

Additionally, every sofic shift has a follower-separated presentation. Any presentation can be made into a follower-separated by creating a graph from the equivalence class of follower sets (i.e. two vertices in a presentation are equivalent if they have the same follower set), and connecting any two equivalence classes if they have an edge between any of the vertices between the equivalence classes [Lin+95]. We call this process *follower-separation*.

Example 2.16. As the vertex in the upper right of Figure 2.1d is a vertex that has two edges starting at it labeled 1, this presentation is nondeterministic, but still presents the even shift.

Theorem 2.17 ([Lin+95] Fundamental theorem of minimal deterministic presentations of irreducible sofic shifts)**.** Let X be an irreducible sofic shift.

(i) Any minimal deterministic presentation of X is follower-separated and irreducible.

(ii) Any two irreducible deterministic presentations of X that are also follower-separated are isomorphic.

(iii) Therefore, any two minimal deterministic presentations of X are isomorphic.

(iv) Additionally, a deterministic presentation, up to isomorphism, is the unique minimal deterministic presentation of X if and only if it is irreducible and follower-separated.

The idea of follower-separation once again borrows ideas from automata theory, as minimization of DFAs is done by creating a DFA from the equivalence class of states (defined in a similar manner). However, DFA minimization always yields the minimal DFA, but follower-separation does not necessarily yield a minimal presentation. Follower-separation is only guaranteed to give the minimal presentation if the original presentation is irreducible ([Lin+95, Lemma 3.3.8]).

Example 2.18. The presentation in Figure 2.1c is an irreducible presentation of the even shift but is not follower-separated, so by Theorem 2.17, it is not minimal. The presentation Figure 2.1a is an irreducible follower-separated presentation of the even shift, so by Theorem 2.17, it is minimal.

Definition 2.19. Let $\mathcal{G}$ be a presentation of a sofic shift X, I be some vertex in $\mathcal{G}$, and $w \in \mathcal{B}(X)$. If every path π in $\mathcal{G}$ that presents w ends at I, then we say w *synchronizes to* I. We say that w is *synchronizing for* $\mathcal{G}$ if it synchronizes to some vertex in $\mathcal{G}$. If there is a word that synchronizes to I, then we say that the vertex I *is synchronizing*. Finally, we denote $S(\mathcal{G})$ as the set of all synchronizing words for $\mathcal{G}$.

Chapter 3

Irreducibility

Consider the presentation in Figure 3.1. The shift presented by subgraph induced by I and J is the even shift, and every word presented by a path starting from K is presented by some path starting at I or J. Hence, this presentation presents the even shift. Additionally, it is also follower-separated as $01 \in F(K)\backslash F(I)$, $1 \in F(K)\backslash F(J)$, and $1 \in F(I)\backslash F(J)$. When do follower-separated reducible presentations present irreducible shifts? We will first look at a simple class of reducible presentations.

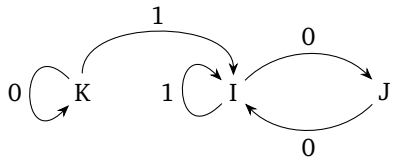

Figure 3.1: A reducible, follower-separated presentation of the even shift

Given a graph and vertices I and J in the graph, we say I *communicates* with J if and only if I is reachable from J and J is reachable from I. By saying a vertex is always reachable from itself, it is routine to check that this defines an equivalence relation. The equivalence classes are called *irreducible components*, as the subgraphs induced by each class are irreducible.

Let $\mathcal{G}^{\rightarrow}\mathcal{H}$ be an essential, deterministic, follower-separated presentation with two irreducible components that induce two subgraphs, namely $\mathcal{G}$ and $\mathcal{H}$, such that there is exactly one edge starting in $\mathcal{G}$ and ending in $\mathcal{H}$. Some properties of $\mathcal{G}^{\rightarrow}\mathcal{H}$ are that the vertices of $\mathcal{G}$ and $\mathcal{H}$ partition the vertices of $\mathcal{G}^{\rightarrow}\mathcal{H}$, both $\mathcal{G}$ and $\mathcal{H}$ are essential, any vertex in $\mathcal{H}$ is reachable from any vertex in $\mathcal{G}^{\rightarrow}\mathcal{H}$, and no vertex in $\mathcal{G}$ is reachable from any vertex in $\mathcal{H}$.

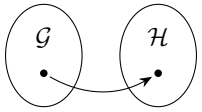

Figure 3.2: Representation of $\mathcal{G}^{\rightarrow}\mathcal{H}$.

Proposition 3.1. Every word $u \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$ can be extended on the right to a word $uw \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$ that synchronizes to a vertex I in $\mathcal{H}$.

Proof. Let $u \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$. As $\mathcal{G}^{\rightarrow}\mathcal{H}$ is deterministic and follower-separated, then by [Lin+95, Proposition 3.3.16], we can extend u on the right to $uw \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$ so that uw synchronizes to some vertex J. As I is a vertex in $\mathcal{H}$, I can be reached from J, so let v be the label of a path from J to I. Then, $uwv \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$ and any path presenting uwv must end at I, so uwv synchronizes to I. □

Corollary 3.2. Every vertex in $\mathcal{H}$ is synchronizing for $\mathcal{G}^{\rightarrow}\mathcal{H}$.

Proof. For a vertex I in $\mathcal{H}$, take any word $u \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$ and use Proposition 3.1 to synchronize it to I. Hence, there is a word that synchronizes to I, so I is synchronizing. □

Lemma 3.3. If there exists a word $u \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$ such that $u \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$, then $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}$ is reducible.

Proof. By Proposition 3.1, we can extend u on the right to a word $uw \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$ so that any path presenting uw synchronizes to a vertex I in $\mathcal{H}$. We want to show that there is no word joining uw and u. As uw synchronizes to I, then by [Lin+95, Lemma 3.3.15], $F(I) = F(uw)$. From the definition of irreducibility and follower sets of words, we can see v is a word joining uw and u exactly when $v \in F(uw)$ and $u \in F(uwv)$. For any word $v \in F(I)$, there is some path labeled v starting at I and ending at J for some vertex J in $\mathcal{G}^{\rightarrow}\mathcal{H}$. Clearly, J is a vertex in $\mathcal{H}$ as no vertex in $\mathcal{G}$ is reachable from I, so $F(J) \subseteq \mathcal{B}(\mathsf{X}_{\mathcal{H}})$. But $u \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$, so $u \notin F(uwv)$. Thus, there is no word joining uw and u, so $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}$ is reducible. □

Theorem 3.4. $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}$ is irreducible if and only if $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}} = \mathsf{X}_{\mathcal{H}}$.

Proof. Suppose $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}$ is irreducible, and let $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$. By Lemma 3.3, as $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}$ is irreducible, then $w \in \mathcal{B}(\mathsf{X}_{\mathcal{H}})$, so $\mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}) \subseteq \mathcal{B}(\mathsf{X}_{\mathcal{H}})$. By construction of $\mathcal{G}^{\rightarrow}\mathcal{H}$, we have $\mathcal{B}(\mathsf{X}_{\mathcal{H}}) \subseteq \mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}})$. Therefore, $\mathcal{B}(\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}) = \mathcal{B}(\mathsf{X}_{\mathcal{H}})$ so $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}} = \mathsf{X}_{\mathcal{H}}$. Conversely, suppose $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}} = \mathsf{X}_{\mathcal{H}}$. As $\mathsf{X}_{\mathcal{H}}$ is irreducible, then so is $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}$. □

Corollary 3.5. If $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}$ is irreducible, then $\mathsf{X}_{\mathcal{G}} \subseteq \mathsf{X}_{\mathcal{H}}$.

Proof. By Theorem 3.4, if $\mathsf{X}_{\mathcal{G}\to\mathcal{H}}$ is irreducible, then $\mathsf{X}_{\mathcal{G}\to\mathcal{H}} = \mathsf{X}_{\mathcal{H}}$. As $\mathsf{X}_{\mathcal{G}} \subseteq \mathsf{X}_{\mathcal{G}\to\mathcal{H}}$, then $\mathsf{X}_{\mathcal{G}} \subseteq \mathsf{X}_{\mathcal{H}}$. □

Theorem 3.6. $\mathsf{X}_{\mathcal{G}} \subseteq \mathsf{X}_{\mathcal{H}}$ if and only if no vertex in $\mathcal{G}$ is synchronizing for $\mathcal{G}^{\to}\mathcal{H}$.

Proof. Suppose $\mathsf{X}_{\mathcal{G}} \nsubseteq \mathsf{X}_{\mathcal{H}}$. Then, there is a word $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$ but $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$. We can extend w to a word $wu \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$ such that wu is synchronizing for $\mathcal{G}$. If every path in $\mathcal{G}^{\to}\mathcal{H}$ presenting wu ends in $\mathcal{G}$, then wu is synchronizing for $\mathcal{G}^{\to}\mathcal{H}$ (as every path presenting wu would be a path in $\mathcal{G}$). Otherwise, let π be any presenting wu starting at some vertex in $\mathcal{G}$ and ending at a vertex in $\mathcal{H}$. As $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$, there is no path in $\mathcal{H}$ presenting w, so the label of any path starting with π does not end with the word w. As π was arbitrary, then any path that starts in $\mathcal{G}$ and ends in $\mathcal{H}$ whose label begins with wu does not end with w. But as $\mathsf{X}_{\mathcal{G}}$ is irreducible, $wu \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$, and $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$, there is a word v such that $wuvw \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$. Therefore, any path in $\mathcal{G}^{\to}\mathcal{H}$ presenting this $wuvw$ cannot end in $\mathcal{H}$, so it must end in $\mathcal{G}$ and thus $wuvw$ is synchronizing for $\mathcal{G}$.

Conversely, suppose there is a word $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}\to\mathcal{H}})$ that synchronizes to a vertex in $\mathcal{G}$. No path presenting this word can start in $\mathcal{H}$ as no vertex in $\mathcal{G}$ is reachable from any vertex in $\mathcal{H}$, so $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$. Clearly, $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$ as there is some path presenting w that starts and ends in $\mathcal{G}$, so $\mathcal{B}(\mathsf{X}_{\mathcal{G}}) \nsubseteq \mathcal{B}(\mathsf{X}_{\mathcal{H}})$. □

Corollary 3.7. If $\mathsf{X}_{\mathcal{G}\to\mathcal{H}}$ is irreducible, then no vertex in $\mathcal{G}$ is synchronizing for $\mathcal{G}^{\to}\mathcal{H}$.

Theorem 3.8. $\mathsf{X}_{\mathcal{G}\to\mathcal{H}} = \mathsf{X}_{\mathcal{H}}$ if and only if $S(\mathcal{G}^{\to}\mathcal{H}) \subseteq S(\mathcal{H})$.

Proof. Suppose $\mathsf{X}_{\mathcal{G}\to\mathcal{H}} = \mathsf{X}_{\mathcal{H}}$. Let $w \in B(\mathsf{X}_{\mathcal{G}\to\mathcal{H}})$ synchronize to some vertex I in $\mathcal{G}^{\to}\mathcal{H}$. If π is some path in $\mathcal{H}$ presenting w, then it must end at I. By Corollary 3.7, I cannot be a vertex in $\mathcal{G}$, so it must be a vertex in $\mathcal{H}$. Therefore, every path in $\mathcal{H}$ presenting w ends at I, so $S(\mathcal{G}^{\to}\mathcal{H}) \subseteq S(\mathcal{H})$ as the above holds for every word w that is synchronizing for $\mathcal{G}^{\to}\mathcal{H}$.

Conversely, suppose $S(\mathcal{G}^{\to}\mathcal{H}) \subseteq S(\mathcal{H})$. Let $u \in \mathcal{B}(\mathsf{X}_{\mathcal{G}\to\mathcal{H}})$. By Proposition 3.1, we can extend u to a word $uw \in \mathcal{B}(\mathsf{X}_{\mathcal{G}\to\mathcal{H}})$ that synchronizes to some vertex I in $\mathcal{G}^{\to}\mathcal{H}$. Hence, uw is synchronizing for $\mathcal{G}^{\to}\mathcal{H}$, so it is also synchronizing for $\mathcal{H}$. But if uw is synchronizing for $\mathcal{H}$, then by definition, $uw \in \mathcal{B}(\mathsf{X}_{\mathcal{H}})$ and thus $u \in \mathcal{B}(\mathsf{X}_{\mathcal{H}})$ (as $\mathcal{B}(\mathsf{X}_{\mathcal{H}})$ is factorial). Therefore, $\mathcal{B}(\mathsf{X}_{\mathcal{G}\to\mathcal{H}}) \subseteq \mathcal{B}(\mathsf{X}_{\mathcal{H}})$, so $\mathsf{X}_{\mathcal{G}\to\mathcal{H}} = \mathsf{X}_{\mathcal{H}}$. □

Proposition 3.9. If $w \in S(\mathcal{G}^{\to}\mathcal{H})$ but $w \notin S(\mathcal{H})$, then $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$.

Proof. As $w \in S(\mathcal{G}^{\to}\mathcal{H})$, there is a vertex I in $\mathcal{G}^{\to}\mathcal{H}$ such that every path in $\mathcal{G}^{\to}\mathcal{H}$ presenting w ends at I. If $w \notin S(\mathcal{H})$, then either $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$ or for every vertex J in $\mathcal{H}$ there is a path in $\mathcal{H}$ that presents w but does not end at J. Hence if the latter

condition were true, then this implies there is a path in $\mathcal{H}$ that presents w and does not end at I. But this path is also in $\mathcal{G}^{\rightarrow}\mathcal{H}$ and presents w, so it should end at I, which is a contradiction. Therefore, $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$. □

Chapter 4

Subshift testing

In the previous section, we posited that if $\mathsf{X}_{\mathcal{G}\to\mathcal{H}}$ is irreducible, then $\mathsf{X}_{\mathcal{G}}$ is a subshift of $\mathsf{X}_{\mathcal{H}}$ (Corollary 3.5). Although this condition is not sufficient for irreducibility (for example, Figure 2.1b, the presentation of the 010-shift), algorithmically, we will show there is a polynomial-time algorithm for deciding this, which gives a direction in looking for a characterization of irreducibility conditional on this subshift property.

Let $\mathcal{G}$ be a deterministic presentation and I be a vertex in $\mathcal{G}$. As every edge starting at I is labeled uniquely, we can see that every path starting at I is also labeled uniquely. That is to say, for paths π and τ both starting at I, if $\mathcal{L}(\pi) = \mathcal{L}(\tau)$, then $\pi = \tau$. We can define a partial *transition function* $\delta_{\mathcal{G}} : \mathcal{V}_{\mathcal{G}} \times \mathcal{A}_{\mathcal{G}}^* \to \mathcal{V}_{\mathcal{G}}$ with $\delta_{\mathcal{G}}(I, w) \triangleq J$ if there is a path π labeled w starting at I and ending at J and $\delta_{\mathcal{G}}(I, \epsilon) = I$ for all I. However, if there is no path labeled w starting at I, then $\delta_{\mathcal{G}}(I, w)$ is not defined, so in this case, we will say $\delta_{\mathcal{G}}(I, w) \triangleq 0$, where 0 is a constant distinct from the vertices of $\mathcal{G}$. Additionally, we will define $\delta_{\mathcal{G}}(0, w) = 0$. Finally, define the *subset transition function* $\Delta_{\mathcal{G}} : \mathcal{P}(\mathcal{V}) \times \mathcal{A}_{\mathcal{G}}^* \to \mathcal{P}(\mathcal{V})$ with $\Delta_{\mathcal{G}}(S, w) \triangleq \{J \in \mathcal{V}_{\mathcal{G}} : J \neq 0 \text{ and } \delta_{\mathcal{G}}(I, w) = J \text{ for some } I \in S\}$ for all subsets S of vertices of $\mathcal{G}$.

Proposition 4.1. Let $\mathcal{G}$ be an essential, deterministic presentation, and I and J be vertices in $\mathcal{G}$. The following properties of the transition function are true:

(i) $w \in F(I)$ if and only if $\delta_{\mathcal{G}}(I, w) \neq 0$

(ii) $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$ if and only if $\Delta_{\mathcal{G}}(\mathcal{V}_{\mathcal{G}}, w) \neq \varnothing$

(iii) $\delta_{\mathcal{G}}(I, uv) = \delta_{\mathcal{G}}(\delta_{\mathcal{G}}(I, u), v)$ for all vertices I in $\mathcal{G}$ and $u, v \in \mathcal{A}_{\mathcal{G}}^*$

(iv) $\Delta_{\mathcal{G}}(S, uv) = \Delta_{\mathcal{G}}(\Delta_{\mathcal{G}}(S, u), v)$ for all subsets of vertices S of $\mathcal{G}$ and $u, v \in \mathcal{A}_{\mathcal{G}}^*$

Proof. Note that $w \in F(I)$ exactly when there is a path labeled w starting at I, so

(i) follows evidently from the definition of $\delta_{\mathcal{G}}$. Similarly, $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$ exactly when $w \in F(I)$ for some vertex I in $\mathcal{G}$, so (ii) follows from the definition of $\Delta_{\mathcal{G}}$.

For (iii), let $u, v \in \mathcal{A}_{\mathcal{G}}^*$. If $\delta_{\mathcal{G}}(I, uv) \neq 0$, then there is a unique path $\pi = e_1 \dots e_n$ labeled uv starting at I and ending at J. As $uv = \mathcal{L}(e_1) \dots \mathcal{L}(e_n)$, then for nonempty v, we can see that there is some i such that $u = \mathcal{L}(e_1) \dots \mathcal{L}(e_i)$ and $v = \mathcal{L}(e_{i+1}) \dots \mathcal{L}(e_n)$ (for $v = \epsilon$, what we are trying to show follows trivially). Let $\rho_1 = e_1 \dots e_i$ and $\rho_2 = e_{i+1} \dots e_n$. As ρ_1 is a path starting at $i(e_1)$ and ending at $t(e_i)$ labeled u, $\delta_{\mathcal{G}}(I, u) = t(e_i)$. Similarly, as ρ_2 is a path starting at $t(e_i)$ and ending at J, $\delta_{\mathcal{G}}(t(e_i), v) = J$. Therefore, $\delta_{\mathcal{G}}(\delta_{\mathcal{G}}(I, u), v) = J$. As there is a path labeled uv starting at I and ending at J, then $\delta_{\mathcal{G}}(I, uv) = J$.

If $\delta_{\mathcal{G}}(I, uv) = 0$, then there is no path labeled uv starting at I. If there is no path labeled u starting at I, then $\delta_{\mathcal{G}}(\delta_{\mathcal{G}}(I, u), v) = \delta_{\mathcal{G}}(0, v) = 0$. Otherwise, if there is a path labeled u starting at I, then there must be no path labeled v starting at $\delta_{\mathcal{G}}(I, u)$, as otherwise, $\delta_{\mathcal{G}}(\delta_{\mathcal{G}}(I, u), v) \neq 0$ and then $\delta_{\mathcal{G}}(\delta_{\mathcal{G}}(I, u), v) = \delta_{\mathcal{G}}(I, uv) \neq 0$. Therefore, $\delta_{\mathcal{G}}(\delta_{\mathcal{G}}(I, u), v) = 0$, which completes the proof for (iii). An similar argument to (iii)'s proof can be done applied to (iv). □

Definition 4.2. Let $\mathcal{G}$ be an essential, labeled graph. The $\mathcal{G}$ *kill state graph with alphabet* $\mathcal{A}$ is the labeled graph $\mathcal{G}^0$ taking the same vertices and edges as $\mathcal{G}$ with an additional vertex 0 and edges such that for each $a \in \mathcal{A}$ and vertex I in $\mathcal{G}$, if there is no edge starting at I labeled a, then there is an edge in $\mathcal{G}^0$ between I and 0 labeled a. Additionally, for each $a \in \mathcal{A}$, there is an edge from 0 to 0.

From the definition of $\mathcal{G}^0$, we can see that for all vertices I in $\mathcal{G}$, $w \in F(I)$ if and only if there is no path in $\mathcal{G}^0$ labeled w starting at I and ending at 0.

Definition 4.3. Let $\mathcal{G}$ and $\mathcal{H}$ be labeled graphs. The *label product* of $\mathcal{G}$ and $\mathcal{H}$ is the labeled graph $\mathcal{G} * \mathcal{H}$ such that if there is an edge e_1 in $\mathcal{G}$ between I_1 and J_1 and an edge e_2 in $\mathcal{H}$ between I_2 and J_2 with $\mathcal{L}_{\mathcal{G}}(e_1) = \mathcal{L}_{\mathcal{H}}(e_2)$, then there is an edge in $\mathcal{G} * \mathcal{H}$ between (I_1, I_2) and (J_1, J_2) labeled $\mathcal{L}_{\mathcal{G}}(e_1) = \mathcal{L}_{\mathcal{H}}(e_2)$.

From the definition of $\mathcal{G} * \mathcal{H}$, we can see that there is a word w and paths π in $\mathcal{G}$ and τ in $\mathcal{H}$ both labeled w with π starting at I_1 and ending at J_1 and τ starting at I_2 and ending at J_2 if and only if there is a path in $\mathcal{G} * \mathcal{H}$ starting at (I_1, J_1) and ending at (I_2, J_2).

With these two graph constructions, we can see that for essential presentations $\mathcal{G}$ and $\mathcal{H}$ and for a vertex I in $\mathcal{G}$ and a vertex J in $\mathcal{H}$, checking $F(I) \subseteq F(J)$ reduces to checking the existence of a path in $\mathcal{G} * \mathcal{H}^0$ from (I, J) to any vertex in the set $\{(K, 0) : K \in \mathcal{V}_{\mathcal{G}}\}$. With this, we can introduce Algorithm 1, which given an irreducible deterministic presentation $\mathcal{G}$ and an essential deterministic presentation

$\mathcal{H}$, tests whether $\mathcal{G}$ presents a subshift of the shift $\mathcal{H}$ presents. From a high level, the algorithm takes a vertex in $\mathcal{G}$ and a vertex in $\mathcal{H}$ and tries to find a word in the follower set of the vertex in $\mathcal{G}$ but not in the follower set of the vertex in $\mathcal{H}$. If it finds this word, the algorithm transitions the vertex in $\mathcal{G}$ forward using the word and transitions a subset of vertices in $\mathcal{H}$ forward using the word. If the size subset of vertices from $\mathcal{H}$ ever reaches 0, then the algorithm has found a word in $\mathcal{B}(\mathsf{X}_{\mathcal{G}})$ not in $\mathcal{B}(\mathsf{X}_{\mathcal{H}})$. Otherwise, if it does not find this word, then every word in $\mathcal{B}(\mathsf{X}_{\mathcal{G}})$ must be in $\mathcal{B}(\mathsf{X}_{\mathcal{H}})$.

Algorithm 1 Subshift testing

Require: $\mathcal{G}$ is an irreducible and deterministic presentation, $\mathcal{H}$ is an essential and deterministic presentation

```
1:  procedure IS_SUBSHIFT(𝒢, ℋ)
2:      I ← any element in V_𝒢
3:      X ← V_ℋ
4:      w ← ϵ
5:      repeat
6:          J ← any element in X
7:          find a word u such that δ_𝒢(I, u) ≠ 0 and δ_ℋ(J, u) = 0
8:          if u exists then
9:              w ← wu
10:             I ← δ_𝒢(I, u)
11:             X ← Δ_ℋ(X, u)
12:             if X = ∅ then
13:                 return false
14:             end if
15:         end if
16:     until u does not exist
17:     return true
18: end procedure
```

Lemma 4.4. If I_0 is the value of I before entering the loop on lines 5-16, the invariants

(i) $\delta_{\mathcal{G}}(I_0, w) = I$

(ii) $I \neq 0$

(iii) $\Delta_{\mathcal{H}}(\mathcal{V}_{\mathcal{H}}, w) = X$

hold before the beginning of each iteration the loop. Additionally, the loop always

terminates.

Proof. Clearly, $I = I_0$ and $w = \epsilon$ before entering the loop, so $\Delta_{\mathcal{G}}(I_0, w) = \delta_{\mathcal{G}}(I, \epsilon) = I$. As $I_0 \in \mathcal{V}_{\mathcal{G}}$, we have that $I \neq 0$. Finally, as $X = \mathcal{V}_{\mathcal{G}}$ before entering the loop, $\Delta_{\mathcal{H}}(\mathcal{V}_{\mathcal{H}}, w) = \Delta_{\mathcal{H}}(X, \epsilon) = X$, so the invariants hold before entering the loop.

Suppose the invariants were true before the current iteration of the loop and I, J, w, and X had the values I_n, J_n, w_n and X_n, respectively. Furthermore, suppose the loop does not exit after the current iteration. Because of this, then the algorithm must enter the if statement on line 8 (and not enter the if statement on line 12), so there is a word u such that $\delta_{\mathcal{G}}(I_n, u) \neq 0$ and $\delta_{\mathcal{H}}(J_n, u) = 0$. We can see that $w = w_n u$, $I = \delta_{\mathcal{G}}(I_n, u)$, and $X = \Delta_{\mathcal{H}}(X_n, u)$ at the end of the current iteration, so from this, we can see

$$\delta_{\mathcal{G}}(I_0, w) = \delta_{\mathcal{G}}(I_0, w_n u) = \delta_{\mathcal{G}}(\delta_{\mathcal{G}}(I_0, w_n), u) = \delta_{\mathcal{G}}(I_n, u) = I$$

and similarly,

$$\Delta_{\mathcal{H}}(\mathcal{V}_{\mathcal{H}}, w) = \Delta_{\mathcal{H}}(\mathcal{V}_{\mathcal{H}}, w_n u) = \Delta_{\mathcal{H}}(\Delta_{\mathcal{H}}(\mathcal{V}_{\mathcal{H}}, w_n), u) = \Delta_{\mathcal{H}}(X_n, u) = X.$$

Therefore, all the invariants hold before the next iteration of the loop. Finally, note that as $\delta_{\mathcal{H}}(J_n, u) = 0$ and $J_n \in X$, $|X|$ is strictly less than $|X_n|$. This guarantees the loop terminates as if the exit condition is never true, then $|X| = 0$ will be true for some iteration so $X = \varnothing$ and the algorithm exits on line 13. □

Theorem 4.5. Algorithm 1 returns true if and only if $\mathsf{X}_{\mathcal{G}} \subseteq \mathsf{X}_{\mathcal{H}}$.

Proof. Suppose the algorithm returned false. If so, then it exited at line 13, so we know $X = \varnothing$. By Lemma 4.4, $\delta_{\mathcal{G}}(I_0, w) = I \neq 0$ so $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$ and $\Delta_{\mathcal{H}}(\mathcal{V}_{\mathcal{H}}, w) = \varnothing$ so $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$. Therefore, $\mathsf{X}_{\mathcal{G}} \nsubseteq \mathsf{X}_{\mathcal{H}}$.

Conversely, suppose $\mathsf{X}_{\mathcal{G}} \nsubseteq \mathsf{X}_{\mathcal{H}}$. Then, there is a word $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$ but $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$. As $w \in \mathcal{B}(\mathsf{X}_{\mathcal{G}})$, then $\delta_{\mathcal{G}}(I^*, w) \neq 0$ for some vertex I^* in $\mathcal{G}$, and as $w \notin \mathcal{B}(\mathsf{X}_{\mathcal{H}})$, then $\delta_{\mathcal{H}}(J^*, w) = 0$ for all vertices J^* in $\mathcal{H}$. Therefore, for any value of I, as $\mathcal{G}$ is irreducible, there is a path π starting at I and ending at I^*, so

$$\delta(I, \mathcal{L}(\pi)w) = \delta_{\mathcal{G}}(\delta_{\mathcal{G}}(I, \mathcal{L}(\pi)), w) = \delta_{\mathcal{G}}(I^*, w) \neq 0$$

and for any value of J, $\delta_{\mathcal{H}}(J, \mathcal{L}(\pi)w) = \delta_{\mathcal{H}}(\delta_{\mathcal{H}}(J, \mathcal{L}(\pi)), w) = \varnothing$. Hence, the algorithm can always find a word u on line 7. If the algorithm picks u to be $\mathcal{L}(\pi)w$, then the algorithm terminates, as $\Delta_{\mathcal{H}}(\mathcal{V}_{\mathcal{H}}, \mathcal{L}(\pi)w) = \varnothing$. If not, then as discussed in Lemma 4.4, the the current value of $|X|$ is strictly less than the previous value of $|X|$. Therefore, as the algorithm can always find a word u on line 7, then $|X| = 0$ eventually and thus will return false eventually. □

Theorem 4.6. Algorithm 1 can be computed in $O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|^3 + |\mathcal{E}_\mathcal{G}| \cdot |\mathcal{E}_\mathcal{H}| \cdot |\mathcal{V}_\mathcal{H}|)$ time.

Proof. As $|X|$ goes down by at minimum once every iteration of the loop, and $|X| = O(|\mathcal{V}_\mathcal{H}|)$, the loop repeats $O(|\mathcal{V}_\mathcal{H}|)$ times. To compute line 7, one can construct the graph $\mathcal{G} * \mathcal{H}^0$ and and determine if there is a path from (I, J) to any vertex in the set $\{(K, 0) : K \in \mathcal{V}_\mathcal{G}\}$ using, say, a depth-first search. As $|\mathcal{V}_{\mathcal{G}*\mathcal{H}^0}| = O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|)$ and $|\mathcal{E}_{\mathcal{G}*\mathcal{H}^0}| = O(|\mathcal{E}_\mathcal{G}| \cdot |\mathcal{E}_\mathcal{H}|)$, then with depth-first search, this step can be computed in $O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}| + |\mathcal{E}_\mathcal{G}| \cdot |\mathcal{E}_\mathcal{H}|)$ time. As $|u| = O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|)$, computing $\delta_\mathcal{G}(I, u)$ takes $O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|)$ time. Similarly, computing $\Delta_\mathcal{H}(X, u)$ is reduces to computing $\delta_\mathcal{H}$ at most $|X|$ times, so this can be computed in $O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|) \cdot O(|\mathcal{V}_\mathcal{H}|) = O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|^2)$ time. In total, this is

$$\begin{aligned}
&O(|\mathcal{V}_\mathcal{H}|) \cdot \big(O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}| + |\mathcal{E}_\mathcal{G}| \cdot |\mathcal{E}_\mathcal{H}|) + O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|) + O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|^2)\big) \\
&= O(|\mathcal{V}_\mathcal{H}|) \cdot O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|^2 + |\mathcal{E}_\mathcal{G}| \cdot |\mathcal{E}_\mathcal{H}|) \\
&= O(|\mathcal{V}_\mathcal{G}| \cdot |\mathcal{V}_\mathcal{H}|^3 + |\mathcal{E}_\mathcal{G}| \cdot |\mathcal{E}_\mathcal{H}| \cdot |\mathcal{V}_\mathcal{H}|).
\end{aligned}$$

□

Chapter 5

Discussion and future work

Although the algorithm does not fully solve the problem of deciding the irreducibility problem, it does shed some light on a general minimization procedure. For a $\mathcal{G}^{\rightarrow}\mathcal{H}$-like graph, if Algorithm 1 returns false on inputs $\mathcal{G}$ and $\mathcal{H}$, then we actually know the presentation is *synchronizing*, meaning that every vertex is synchronizing. As such, since $\mathcal{G}^{\rightarrow}\mathcal{H}$ is follower-separated, then by [Jon96, Corollary 5.4], it is minimal.

Looking forwards, determining when $\mathsf{X}_{\mathcal{G}^{\rightarrow}\mathcal{H}}$ is irreducible when $\mathsf{X}_{\mathcal{G}} \subseteq \mathsf{X}_{\mathcal{H}}$ intuitively seems to be linked with how to check if all words coming from $\mathcal{G}$ link up "properly" with $\mathcal{H}$. For example, in Figure 3.1, this it is simple to see that $\mathcal{G}$ links up with $\mathcal{H}$ properly. On the contrary, the presentation in Figure 5.1 is similar to Figure 3.1 in that $\mathsf{X}_{\mathcal{G}} \subseteq \mathsf{X}_{\mathcal{H}}$, but connects to the wrong place in $\mathcal{H}$, and fails to present the even shift, as it introduces the word 101 into the language.

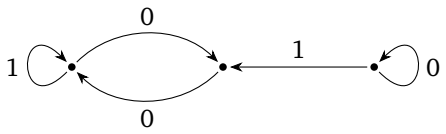

Figure 5.1

The results here are likely to be easily extended to presentations with arbitrarily many irreducible components. In fact, [Jon96] notes that for any presentation of an irreducible sofic shift, the subgraph induced by the synchronizing vertices presents the whole shift. If given an arbitrary presentation that presents an irreducible shift, then to minimize it, one would need to find a synchronizing word and then identify the irreducible component it synchronizes to. Although not explicitly discussed

here, finding a synchronizing word should be doable in polynomial-time with a modification to [Epp90]'s algorithm for finding a synchronizing word for DFAs.[1] Subshift testing can also be used for partial testing of the irreducibility of the shift a presentation presents. As each shift presented by an irreducible component is a subshift of the shift the whole presentation presents, it is easy to see that if a presentation presents an irreducible shift, then it is necessary that each irreducible component present a subshift of the synchronizing component.

Finally, the hardness of the general problem of minimization seems to have not been investigated. Those familiar with automata theory can see that presentations of sofic shifts are closely related to *nondeterministic finite automata* (NFA), as such presentations can be interpreted as NFAs by making the set of start states all the vertices of the presentation and every state a final state. If we restrict ourselves to deterministic presentations, then the only nondeterminism in the interpreted NFA is the multiple start states. In [HSY01] and [Mal04], minimization of a variant of a DFA that allows multiple start states (MDFA) is shown to be PSPACE-complete in the former for the general case and NP-complete in the latter for a restricted case. Every presentation of a sofic shift gives a MDFA, but only ones where every state is a final state. Further investigation should be done to see if this restriction affects the hardness of minimization.

[1] Synchronizing words for DFAs, also known as reset words, differ slightly from synchronizing words for presentations as the transition function for DFAs is total to begin with, so the action of taking the transition of the reset word takes you to the synchronizing state no matter where you start, while for presentations, the action of taking the transition of the synchronizing word either brings you to 0 or the synchronizing state.

Bibliography

[MH38] Marston Morse and Gustav A Hedlund. “Symbolic dynamics”. In: *American Journal of Mathematics* 60.4 (1938), pp. 815–866.

[Wei73] Benjamin Weiss. “Subshifts of finite type and sofic systems”. In: *Monatshefte für Mathematik* 77.5 (1973), pp. 462–474.

[Fis75] Roland Fischer. “Sofic systems and graphs”. In: *Monatshefte für Mathematik* 80.3 (1975), pp. 179–186.

[Epp90] David Eppstein. “Reset sequences for monotonic automata”. In: *SIAM Journal on Computing* 19.3 (1990), pp. 500–510.

[MR91] Brian H Marcus and Ron M Roth. “Bounds on the number of states in encoder graphs for input-constrained channels”. In: *IEEE transactions on information theory* 37.3 (1991), pp. 742–758.

[JM94] Natasa Jonoska and Brian Marcus. “Minimal presentations for irreducible sofic shifts”. In: *IEEE transactions on information theory* 40.6 (1994), pp. 1818–1825.

[Lin+95] Douglas Lind et al. *An introduction to symbolic dynamics and coding*. Cambridge university press, 1995.

[Jon96] Nataša Jonoska. “Sofic shifts with synchronizing presentations”. In: *Theoretical Computer Science* 158.1-2 (1996), pp. 81–115.

[HSY01] Markus Holzer, Kai Salomaa, and Sheng Yu. “On the state complexity of k-entry deterministic finite automata”. In: *Journal of Automata, Languages and Combinatorics* 6.4 (2001), pp. 453–466.

[HMU01] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. “Introduction to automata theory, languages, and computation”. In: *Acm Sigact News* 32.1 (2001), pp. 60–65.

[Mal04] Andreas Malcher. “Minimizing finite automata is computationally hard”. In: *Theoretical Computer Science* 327.3 (2004), pp. 375–390.