On the complexity of isomorphism in finite group theory and symbolic dynamics

by

Tyler Schrock

B.S., Troy University, 2013

M.A., University Colorado Boulder, 2017

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics

2019

This thesis entitled:
On the complexity of isomorphism in finite group theory and symbolic dynamics
written by Tyler Schrock
has been approved for the Department of Mathematics

_____

Prof. Joshua A. Grochow

_____

Prof. Nathaniel Thiem

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Schrock, Tyler (Ph.D., Mathematics)

On the complexity of isomorphism in finite group theory and symbolic dynamics

Thesis directed by Prof. Joshua A. Grochow

This thesis looks at the complexity of isomorphism in two fairly distinct areas of mathematics.

First, we consider several computational problems related to sub-shifts of finite type and, in particular, conjugacy restricted to k-block codes: verifying a proposed k-block conjugacy, deciding if two shifts admit a k-block conjugacy, reducing the representation size of a shift via a k-block conjugacy, and recognizing if a sofic shift is a shift of finite type. We give a polynomial-time algorithm for verification, show GI-hardness for deciding conjugacy, show NP-hardness for reducing representation size, and give a polynomial-time algorithm for recognizing shifts of finite type. Our approach focuses on 1-block conjugacies between vertex shifts, from which we generalize to k-block conjugacies and to edge shifts. We also highlight several open problems.

Second, we consider isomorphism between quotients of centrally indecomposible genus 2 p-groups. We show isomorphism between quotients of such groups by non-central subgroups can be determined in polynomial time. The centrally indecomposible genus 2 groups split into two cases: flat and sloped [20]. We give a polynomial-time algorithm which correctly decides isomorphism between quotients of the flat genus 2 groups $H_1^\flat(\mathbb{F}_q)$ by central subgroups whenever the algorithm succeeds; we believe the algorithm always succeeds and have tested it on tens of millions of random examples. We again highlight several open problems.

Acknowledgements

First, I thank my advisors[1] Josh Grochow, Raf Frongillo, and Nat Thiem. I will forever be grateful for their guidance, support, and patience.

I also thank the rest of my thesis committee, Robin Deeley and Peter Mayr, for their time and editorial support.

Thanks to Matt Miller, Lara Pudwell, Ken Roblee, and Vitaly Voloshin for going beyond the call of duty to foster my interest of mathematics. Without their challenging and encouragement, I never would have attempted grad school.

Thanks to my Great Exchange family and the beauty of the Southwest for giving me reasons to stay in Boulder when grad school sucked. Thanks to the National Park Service[2] for protecting many precious areas of our country for the enjoyment of all generations. Thanks to Brandon Sanderson[3] for renewing my love of reading while in grad school and Terry Pratchett for giving me an appreciation of footnotes.[4]

Finally, I thank the many friends and family members who have supported and encouraged me throughout grad school. I especially thank my parents for believing I could do this even when I didn't believe it myself.

[1] official or not
[2] now and forever America's best idea
[3] Life before death, strength before weakness, journey before destination.
[4] GNU Terry Pratchett

Contents

Chapter

Tables

Table

Figures

Figure

Algorithms

Algorithm

Chapter 1

Introduction

Generally speaking, this thesis looks at the complexity of determining isomorphism in two fairly distinct fields of mathematics.

1.1 A non-technical introduction to isomorphism

Before introducing our main problems of study, we first introduce isomorphism (What is it and why should we study it?) via language accessible to a non-mathematical audience.

Suppose we live in a world where two distinct temperature systems exist, say °F and °C. Even more, maybe we know a lot of information about °F for the context of what feels hot/cold, how to cook different foods, and other situations encountered in daily life by the average American. At the same time, maybe we know a lot of information about °C in the context of science. Since both °F and °C are measurements of the same thing, there should be a way to convert between them without losing any information. That is, given a recipe (or anything else) written in °F, is there a way to rewrite the recipe to be written in °C for use in a science experiment such that someone else could convert back to °F to get the exact same starting recipe? In fact, there is such a conversion for temperature (using the formula $°\mathrm{F} = \frac{9}{5}°\mathrm{C} + 32$), so °F and °C are essentially the same, even though, on the surface, they appear quite different. This extent of sameness is called isomorphism, where etymologically, "iso" means "same" and "morph[ic/ism]" means "shape."

In this situation, we say the temperature scales of °F and °C are **isomorphic** while the function doing the conversion ($°\mathrm{F} = \frac{9}{5}°\mathrm{C} + 32$) is an **isomorphism**. Once we find an isomorphism

between two things, we can transfer knowledge about one to other in a perfect (lossless) way. In math, different mathematical "objects" appear in many contexts. One such object is called a group. If it happens that a group appears while studying something in Physics and another group appears while studying something in Chemistry, it would be nice to know if the two groups are isomorphic (the "same"). If the two groups are isomorphic, there is a good chance the two topics are related even if they don't appear to be at first glance, and we might be able to use the isomorphism between the groups to transfer Physics knowledge to the Chemistry situation (and vice versa).

Unfortunately, finding an isomorphism between two things can be very difficult. And sometimes harder is the problem of showing the absence of any isomorphism between two things. For an example of non-isomorphism, consider the problem of translating between languages. Every language has the ability to communicate ideas, so it might seem like all languages are isomorphic to one another. However, there is essentially never a way to translate between languages without losing information. For a specific example, consider the problem of translating English to ancient Japanese. Separate words differentiating the colors blue and green in Japanese is a fairly recent creation; historically, both colors were described by the word ao (青). It is possible to translate from English to ancient Japanese, but the translation loses information. To see this, note that translating perfectly back to the original English message given only the translated Japanese message is impossible if the original English message included both the words green and blue. Thus there can be no isomorphism between English messages and ancient Japanese messages.

1.2 Summary of results

Stepping up a level in technicality, we now introduce the problems of study in this thesis.

1.2.1 Symbolic dynamics

In Chapter 3, we look at isomorphism in the field of symbolic dynamics. Specifically, we study isomorphism of shifts of finite type, where the specific type of isomorphism we are interested in is that of topological **conjugacy**. Leaving the technical definition of a shift of finite type for

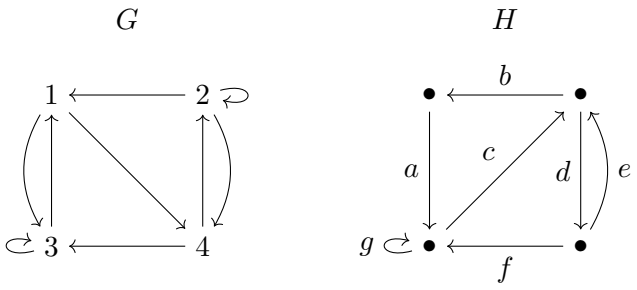

Figure 1.1: (G) An example of a directed graph representing a vertex shift. (H) An example of a directed graph representing an edge shift.

§2.3, we instead note that two ways to present shifts of finite type are via vertex shifts and edge shifts, which we now describe.

Given a directed graph G, label the vertices distinctly (or edges for the case of an edge shift). See Figure 1.1 for an example of both a vertex shift and an edge shift. Consider something (say a particle) which lives on the vertices of the graph. As time advances by one step, the particle moves forward on any edge leaving its current vertex. Assuming the particle lived infinitely in the past and will live infinitely in the future, every walk on G indexed by $\mathbb{Z}$ corresponds to a possible trajectory of the particle. The **vertex shift** associated to G, denoted X_G, is the collection of all possible walks on G indexed by $\mathbb{Z}$ and corresponds to all possible trajectories of a particle living on G. Similarly, the **edge shift** associated to H where the edges of H are labeled as in Figure 1.1, denoted X_H^e, is the collection of all possible walks on H indexed by $\mathbb{Z}$.

Given two vertex shifts X_G, X_H, we can map a **point** p in X_G (a walk on G indexed by $\mathbb{Z}$) to a point in X_H by looking at the block of vertices $p_{[i-m,i+a]}$ between times $i-m$ and $i+a$ in the walk and mapping that block to a single vertex q_i in H. Letting $k = m+a+1$, we call such a mapping a **k-block code**. In the case of 1-block codes, a function from X_G to X_H is evaluated on the point p by looking at every character (vertex) in p and mapping it to a new character (vertex) in H. If it happens that such a mapping from X_G to X_H is bijective (on walks indexed by $\mathbb{Z}$), we say the mapping is a **conjugacy**. If X_G, X_H are conjugate, we write $X_G \cong X_H$.

Motivating our study is the following question, which has been studied since at least 1973 [74]

(but almost certainly longer) and is probably the biggest open question in symbolic dynamics [18]. Given two shifts of finite type X, Y, are X, Y conjugate? Whether or not this question is decidable by an algorithm is still an open problem. As a possible step toward showing the decidability of this problem, we first show the question of verifying a proposed conjugacy is in P.

Theorem A (Corollary 3.2.2.4, Theorem 3.5.1)**.** *Given two shifts of finite type X, Y with alphabets $\mathcal{A}_X, \mathcal{A}_Y$ as either vertex shifts or edge shifts and a proposed k-block conjugacy $\Phi_\infty : X \to Y$, deciding if Φ_∞ is a conjugacy can be determined in $O(|\mathcal{A}_X|^{4k})$ time.*

As a consequence of this theorem, we have that one way to show the conjugacy problem is decidable is to show one can restrict attention to only a finite number of possible conjugacies between two shifts of finite type. Turning to the conjugacy problem itself, we do not resolve its decidability; rather we show it is unlikely to have an efficient solution without employing deep techniques, by showing the conjugacy problem is at least as hard as the (in)famous Graph Isomorphism problem.

Theorem B (Corollary 3.3.7, Theorem 3.5.3)**.** *Given two shifts of finite type X, Y as either vertex shifts or edge shifts, deciding if there is a k-block conjugacy from X to Y is GI-hard for each k.*

Then we look to applications of shifts of finite type. Vertex shifts appear while studying many things (e.g., Markov partitions), so finding smaller representations of a given vertex shift would improve practical algorithms. In the case of Markov partitions specifically, reducing the size of a given vertex shift by a 1-block conjugacy is especially important. We show that, unfortunately, finding optimal solutions to this problem is NP-complete.

Theorem C (Theorem 3.4.9)**.** *Given a vertex shift X_G and an integer ℓ, deciding if there is a vertex shift X_H where $|V_H| = |V_G| - \ell$ such that X_G, X_H are conjugate via a 1-block code is NP-complete.*

Then we look at the more general object of sofic shifts, which strictly generalize shifts of finite type. Sofic shifts arise in practice, so recognizing when a sofic shift is a shift of finite type is a natural question. Even more, the theory of sofic shifts is less developed than that of shifts of finite type; it would be useful to recognize when a sofic shift is also of finite type, so the more

developed theory can be applied. Using methods similar to those used to prove Theorem A, we find an efficient algorithm to answer the question: Given a nice enough presentation of a sofic shift X, is X a shift of finite type? For our result, we need a presentation which is irreducible and right-resolving. While a right-resolving irreducible presentation exists for every irreducible sofic shift, finding such a presentation (given a less nice presentation) may require exponential blow-up in size using the current known algorithms.

Theorem D (Theorem 3.6.4)**.** *Given an irreducible right-resolving presentation of a sofic shift X as a labeled multigraph, deciding if X is a shift of finite type can be determined in $O(n^4)$ time where n is the number of edges that are labeled.*

1.2.2 Finite group theory

The other main problem we consider is the group isomorphism problem, where we look into a subclass of p-groups; p-groups in general are widely believed to be the hardest cases of group isomorphism. Let GpI be the problem of determining isomorphism between two groups given by generating sets (e.g., as matrices over finite fields) and GpI-CayleyTable be the problem of determining isomorphism between two groups given by Cayley tables (see §2.5.3 for a more extended discussion of ways to represent groups in a computer and the known complexity bounds on the various representation choices). Also, let GI be the problem of determining isomorphism between two graphs. As GI is a common candidate for an NP-intermediate problem and graphs have many practical algorithms, deciding if GI is in P is considered to be one of the most important open question in complexity theory [42].

Unfortunately, graphs are very unstructured, which can make testing isomorphism difficult. Groups, in comparison, have lots of structure. Regarding the complexity of these problems, it is known that

$$\textsf{GpI-CayleyTable} \leq_m^P \textsf{GI} \leq_m^P \textsf{GpI},$$

where the fact that GpI-CayleyTable $\leq_m^P$ GI is known by [57] and the fact that GI $\leq_m^P$ GpI seems to

be well-known among the experts (for a written reference, see [29]). As GI is sandwiched between two variants of the group isomorphism problem, graph isomorphism and group isomorphism are intimately related. Even more, the known bounds on the complexities of the three problems are very similar. In Chapter 4, we will focus on p-groups of exponent p and nilpotence class 2; let GpI-Class2 be GpI restricted to these groups (GpI-Class2 seems to be the main bottleneck to solving GpI efficiently, but the only known formal reduction is from p-groups of exponent p and class $< p$ [29]). Provided the generating sets for GpI-Class2 are sets of matrices (see §2.5.3 for other options), we know GpI-CayleyTable, GI, GpI-Class2 $\in$ NP $\cap$ coAM [25, 27, 29] (see §2.2.2 for an introduction to the complexity classes relevant to this thesis), and a similar sandwiching of GI still holds. Thus any of GpI-CayleyTable, GI, GpI-Class2 being NP-complete implies the polynomial hierarchy collapses [4, 9, 16]. Even more, using standard derandomization assumptions, coAM = coNP [58]; that is, GpI-CayleyTable, GI, GpI-Class2 $\in$ NP $\cap$ coNP under common assumptions. Furthermore, the best known runtimes for algorithms solving any one of GpI-CayleyTable, GI, GpI are all quasi-polynomial in the order of the object (group or graph). While there is some evidence that GpI-CayleyTable is in P while GI and GpI are not in P (in fact, Babai expressed this idea in [5]) as GpI-CayleyTable for solvable groups is in NP $\cap$ coNP under weaker assumptions than mentioned above [3] and there is a known quasi-polynomial algorithm for GpI-CayleyTable [56][1] which is much simpler than any of the known quasi-polynomial algorithms for GI or GpI, the best known algorithm for any of the three problems runs in $|G|^{O((\log |G|)^c)}$-time. In particular, the best known algorithm for GpI-CayleyTable runs in $|G|^{\Theta(\log |G|)}$-time, where the coefficient of the exponent has only recently been reduced to $\frac{1}{2}$ [62]. The best known algorithm for GpI also runs in $|G|^{\Theta(\log |G|)}$-time by [78], which is based on many recent advances [45, 7, 60, 73, 49, 10, 8, 63, 62, 20, 28, 34, 32, 24, 17], so having the exponentially larger input size of a full Cayley table is not known to improve the asymptotic runtime of testing isomorphism of groups. Finally, for GI, the best known algorithm runs in $|V|^{\Theta((\log |V|)^2)}$-time. This algorithm was first announced by Babai in [6] after which Helfgott found a flaw and the claim was withdrawn. Since then, the algorithm has been corrected by Babai (with an update posted

[1] Miller credits Tarjan with this algorithm.

on Babai's personal website) and confirmed by Helfgott. Babai's arXiv posting has not yet been updated, but Helfgott, et al. show this updated result in [31].

The subclass of p-groups studied in Chapter 4 are quotients of genus 2 groups. Roughly speaking, a p-group G of exponent p and nilpotence class 2 is genus 2 if (1) G is defined over some field $\mathbb{F}$, (2) $\mathbb{F}$ is the largest field G can be defined over, and (3) G' is 2 dimensional over $\mathbb{F}$ (see §4.2 for a technical definition). For an example, the group $H_m^\flat(\mathbb{F}_q)$ below is a genus 2 group.

$$H_m^\flat(\mathbb{F}_q) = \left\{ \left[\begin{array}{c|cccc|c} I_2 & e_1 & \cdots & e_m & 0 & z_1 \\ & 0 & e_1 & \cdots & e_m & z_2 \\ \hline & & & & & f_0 \\ & & & I_{m+1} & & \vdots \\ & & & & & f_m \\ \hline & & & & & 1 \end{array}\right] : e_i, f_i, z_i \in \mathbb{F}_q \right\}$$

Note $H_m^\flat(\mathbb{F}_q)$ could be written over $\mathbb{F}_p$ instead of $\mathbb{F}_q$. If this were done, the dimension of G' over $\mathbb{F}_p$ would be larger than 2, so G would appear to have genus larger than 2. However, the technical definition of genus is nuanced enough to disallow this (see Example 2.5.2.3 for an example of a genus 2 group which at first glance appears to be genus 4).

Specifically, in Chapter 4, we ask the question: Given two groups G, H which are known to be quotients of genus 2 groups, is $G \cong H$?

Regarding the focus on quotients of genus 2 groups, we note that isomorphism of both genus 1 groups [49] and genus 2 groups [20] can be decided efficiently. Unfortunately, these results rely on classifying the centrally indecomposible groups of low genus, so extending the strategy to genus 3 groups involves solving a wild problem [13], which therefore seems like not a promising approach to higher genus. Considering the problem of quotient groups, a group which is a quotient of a genus g group often has genus larger than g. Despite this, determining isomorphism of quotients of genus 1 group is solvable in polynomial time [49]; however, the known algorithm also relies on a classification of the centrally indecomposible genus 1 groups, so, again, extending these ideas probably fails for quotients of genus 3 groups. (For each of these results, isomorphism is solvable

in polynomial time in the size of the generators of G, i.e., $O(\text{poly}(\log |G|))$.) Thus, we consider the final piece which may be approachable by current strategies: quotients of genus 2 groups.

To answer our question, we look to the strategy developed to determine isomorphism between quotients of genus 1 groups in [49]. The rough outline of this strategy is as follows. First, any non-abelian quotient of a genus 1 group is shown to be a quotient by a central subgroup. Second, given a non-abelian quotient G of a genus 1 group H, the field H is genus 1 over is shown to be recoverable via linear algebra involving only G. Then G can be written explicitly as a quotient $\hat{H}/N$ where the genus 1 group $\hat{H}$ is canonically constructed from G but the normal subgroup N is not canonical. Third, given two groups G_1, G_2 which are written as $G_1 \cong \hat{H}/N_1$ and $G_1 \cong \hat{H}/N_2$ in step two, $G_1 \cong G_2$ if and only if there exists $\varphi \in \text{Aut}(\hat{H})$ such that $\varphi(N_1) = N_2$.

Unfortunately, this strategy fails to generalize in several key ways. First, there exist non-abelian quotients of genus 2 groups by non-central subgroups. In these cases, we show a reduction to the case of quotients of genus 1 groups. Second, attempting to use the same strategy to write a given group G as a quotient H/N of a canonical genus 2 groups seems to fail at every turn. In the genus 1 case, the field over which a group is genus 1 can be recovered from each of its non-abelian quotients by looking at the center of the quotient's adjoint; this is not true in the genus 2 case. Also, in the genus 1 case, knowing the field over which the group is genus 1 and the order of the genus 1 group restricts the possible genus 1 groups to a single option; this also is not true in the genus 2 case. We develop a new method to recover the field. Then we solve the isomorphism problem for some of the cases where the genus 2 group is uniquely given by the field and the order of the group.

Our first main result shows an efficient algorithm to decide isomorphism in quotients of genus 2 groups when the normal subgroup is not central.

Theorem E (Theorem 4.3.5)**.** *Given a group G which is quotient of centrally indecomposible genus 2 group by a non-central normal subgroup, G is also a quotient of a genus 1 group. In particular, combined with [49], we conclude that isomorphism of such groups can be decided in polynomial time.*

Then we look at the general problem of isomorphism between quotients of genus 2 groups. While we don't find a full isomorphism test, we make major steps toward one of the two cases of indecomposible genus 2 groups.

Theorem F (Conjecture 4.4.4.5)**.** *If Conjectures 4.4.2.5, and 4.4.4.1 are true, then the algorithm of §4.4 decides in polynomial time the isomorphism of quotients of the centrally indecomposible genus 2 group $H_1^\flat(\mathbb{F}_{p^n})$ by central subgroups.*

Regarding the missing pieces conjectured to be true, Conjecture 4.4.2.5 seems to be true based on tens of millions of random examples, and while a few details need verified, we believe Conjecture 4.4.4.1 is true via a slight generalization of the proof in [49] for the more general case of any centrally indecomposible genus 2 group.

1.3 Organization

Chapter 2 gives technical background information for the later chapters. New results begin in Chapter 3, where we investigate various problems related to the conjugacy of shifts of finite type. Then we conclude in Chapter 4 by looking at isomorphism among groups with are quotients of genus 2 groups.

Chapter 2

Preliminaries

In this chapter, we introduce necessary definitions, notation, and background for Chapter 3 and Chapter 4.

2.1 Graph theory

We begin with basic graph-theoretic definitions and conventions. An **undirected graph** $G^u = (V, E)$ is a set of vertices V along with a set of edges E where each $e \in E$ is an unordered pair of vertices (possibly indistinct) from V. When multiple graphs are in play, we will write $G^u = (V_G, E_G)$ to clarify which graphs the vertices or edges correspond to. Pictorially, a graph is represented by drawing each vertex as a node and drawing each edges as a line starting at one edge and ending at the other. See Figure 2.1 for example graphs.

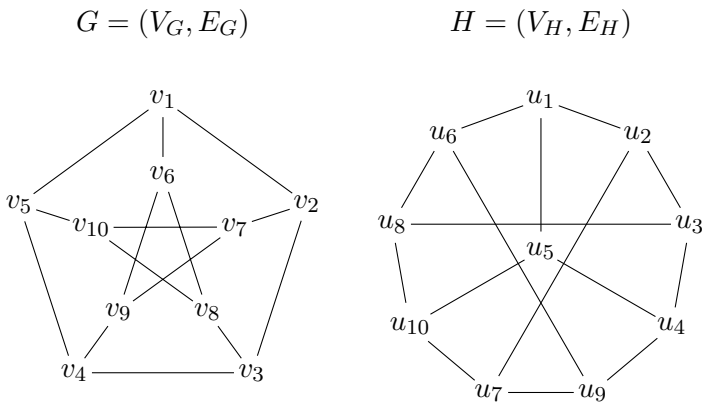

Figure 2.1: Two undirected graphs which are isomorphic via the map $v_i \mapsto u_i$.

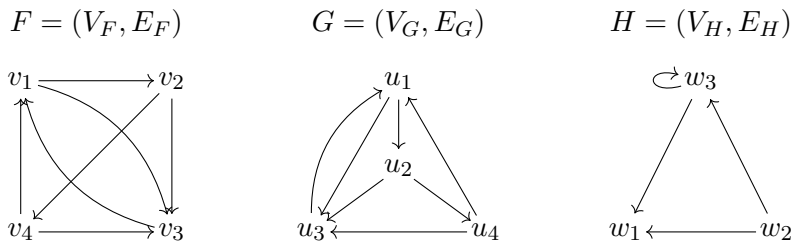

Figure 2.2: Three directed graphs, where graphs F and G are isomorphic via the map $v_i \mapsto u_i$.

For a graph $G = (V, E)$, a **walk** on G is a list of vertices $v_1 v_2 \cdots v_n$ such that $\{v_i, v_{i+1}\} \in E$ for $1 \leq i < n$. For example, in G in Figure 2.1, the path $v_1 v_2 v_7 v_9$ is a walk on G while $v_4 v_5 v_3$ is not a walk on G, since $\{v_5, v_3\} \notin V_G$. Given a graph $G = (V, E)$, a **cycle** of length n will mean a sequence $v_1 v_2 \cdots v_n \in V$ such that $v_1 v_2 \cdots v_n v_1$ is a walk on G. We define $C_n(G)$ to be the set of cycles of length n in $G = (V, E)$. Note that cycles in our definition have a designated starting point, e.g., if $v_1 v_2 v_3$ is a cycle in G, then $v_1 v_2 v_3, v_2 v_3 v_1, v_3 v_1 v_2$ are three distinct cycles in $C_3(G)$.

Two graph G, H are **isomorphic** if there is a bijective mapping $\varphi : V_G \to V_H$ such that φ preserves the edge relation; that is, $\varphi : V_G \to V_H$ is an **isomorphism** from G to H if φ is a bijection and $\{u, v\} \in E_G$ if and only if $\{\varphi(u), \varphi(v)\} \in E_H$. If G, H are isomorphic, we write $G \cong H$. See Figure 2.1 for an example of two graphs which are isomorphic via the isomorphism $\varphi : V_G \to V_H$ defined by $\varphi(v_i) = u_i$. Given two graphs G, H, the complexity of deciding if $G \cong H$ is a major open problem in mathematics [42, 6].

Related, a **directed graph** $G = (V, E)$ is a set of vertices V along with a set of edges $E \subseteq V \times V$. For an edge $e = (v_1, v_2) \in E$, the vertex v_1 is called the **initial vertex** of e and v_2 is called the **terminal vertex** of e. Pictorially, a directed graph is represented by drawing each vertex as a node and each edge (v_1, v_2) as an arrow starting at v_1 and ending at v_2. See Figure 2.2 for examples of directed graphs.

For a directed graph $G = (V, E)$ and a vertex $v \in V$, we define $N^+(v) = \{u \in V : (v, u) \in E\}$ and $N^-(v) = \{u \in V : (u, v) \in E\}$ to be the set of **out-neighbors** and **in-neighbors** of v, respectively. For an example, consider H in Figure 2.2: $N^+(w_3) = \{w_1, w_3\}$ and $N^-(w_3) =$

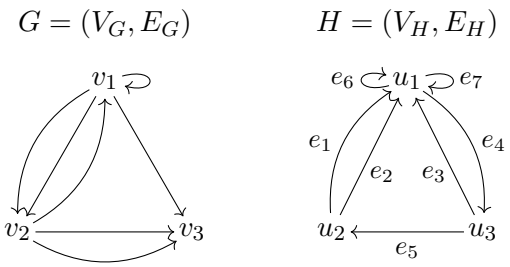

Figure 2.3: Two examples of multigraphs. The edges of G are not labeled, so it is impossible to specify walks on G. The edges of H are labeled, so it is possible to specify walks on H. In particular, the walks $e_5e_1e_6$ and $e_5e_2e_7$ both start at u_3 and end at u_1.

$\{w_2, w_3\}$. Similar to undirected graphs a **walk** on the directed graph G is a listing of vertices $v_1v_2\cdots v_n$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < n$. As an example again in graph H, the path $w_3w_3w_1w_2$ is a walk on H but $w_3w_2w_1$ is not a walk (as the edges have a direction). Then **cycles** and the set $C_n(G)$ are defined identically to the undirected case.

Again, as in the undirected case, two graph G, H are **isomorphic** if there is a bijective mapping $\varphi : V_G \to V_H$ such that φ preserves the edge relation; that is, $\varphi : V_G \to V_H$ is an **isomorphism** from G to H is φ is a bijection and $(u, v) \in E_G$ if and only if $(\varphi(u), \varphi(v)) \in E_H$. See Figure 2.2 for an example where $\varphi : V_F \to V_G$ is an isomorphism defined by $\varphi(v_i) = u_i$ while H is isomorphic to neither F nor G.

Generalizing directed graphs to allow E to be a multiset, we get **directed multigraphs**. As a subtle difference from directed graphs, note that it is necessary to name all the edges in a multigraph in order to refer to a particular walk or cycle. See Figure 2.3 for examples of multigraphs and walks on multigraphs.

Consider any directed graph (or multigraph) $G = (V, E)$. G is **strongly connected** if for every pair of vertices (u, v), there is a walk in G starting at u and ending at v. For example, in Figure 2.2, the graphs F, G are strongly connected while H is not (as there is no walk starting at w_1 and ending at w_3). And in Figure 2.3, the multigraph H is strongly connected while G is not.

We note that while undirected graphs are used in §2.2 for examples and to define the complexity class GI, the main usage of graphs in this thesis are in Chapter 3 where all graphs are

directed. Furthermore, there also are undirected multigraphs; however, all multigraphs in this thesis are directed.

2.2 Computational complexity

This section assumes a basic knowledge of Turing machines and other computational models. For an introduction, see any introductory textbook in the field such as the one by Sipser [69].

2.2.1 Asymptotic complexity

Consider the following algorithm A, which sorts a list of numbers.[1] A says to

(1) Step through the portion of the list which is not yet sorted.

(2) Find the lowest number in the unsorted portion of the list.

(3) Switch the number found in step (2) with the first element of the list which hasn't been marked as sorted.

(4) Mark the first unsorted position as sorted.

(5) Repeat until all list positions have been marked as sorted.

We would like to give some measurement of how efficient A is in terms of time and storage spaced used. Before we can, we note that a few things are clear. The number of steps in the algorithm (and hence the time used) depends on both the size of the list given to the algorithm as well as the particular details of the computational model implementing A. Also, the space used depends on how the data is encoded and whether/how the input's space is counted.

Looking ahead, we note that choice of computational model will not affect the definitions of complexity classes such as P, NP, or PSPACE, so we establish the following conventions. When working with Turing machines, we use a two tape Turing machine with a read-only input tape where the input size is not counted in the space used. However, for general algorithm analysis, we adopt

[1] A happens to be selection sort.

the RAM model. In the RAM model, basic arithmetic and logical operators (adding, multiplying, conditional branching, checking equality, etc.) take exactly one time step. Additionally, memory can be accessed randomly in an efficient manner. That is, reading the data stored at index 18 followed by reading the data stored at index 3 takes exactly 2 time steps.

Let $T(n)$ be the maximum number of steps needed to complete this algorithm on any list of length n. Doing the consecutive iterations unintelligently causes us to look at each of the n numbers of each of the n passes through the list. At each index, we perform at most three operations by first checking if the index has been marked as sorted. If not, we compare the number with the lowest number seen so far and potentially update the index of minimum value for the pass. This takes at most $3n^2$ steps. And at the end of each pass through the list we perform one swap of values with takes 3 steps followed by marking one index as sorted. Overall, we have a runtime no worse than $T_1(n) = 3n^2 + 4n$.

Noting that we don't have to check if an index is marked as sorted if we instead kept track of which pass we were on (and hence where to start in the list), we could reduce the runtime to $T_2(n) = 2 \cdot \frac{(n+1)n}{2} + 4n = n^2 + 5n$. However, no matter what choices we make to improve the implementation of our algorithm, the runtime will always be quadratic. Because of this, we drop constants and lower order terms by writing that for the runtime $T(n)$ of algorithm A, we have $T(n) = O(n^2)$ (said "$T(n)$ is big oh of n^2"). More generally, for any two functions $f, g : \mathbb{N} \to \mathbb{R}^+$, if there exist positive constants M, n_0 such that $n > n_0$ implies $f(n) \leq Mg(n)$, we write $f(n) = O(g(n))$ and say $g(n)$ is an **asymptotic upper bound** for $f(n)$. In practice and for most nice functions, there is a limit test which typically suffices to show if $f(n) = O(g(n))$. Namely, $f(n) = O(g(n))$ whenever $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

Relatedly, we have $f(n) = \Omega(g(n))$ (said "$f(n)$ is big omega of $g(n)$") if $g(n) = O(f(n))$. Also, $f(n) = \Theta(g(n))$ (said "$f(n)$ is theta of $g(n)$") if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. Big Ω notation and Θ notation also have limit tests. Provided the appropriate limit exists, we have $f(n) = \Omega(g(n))$ if $\lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ and $f(n) = \Theta(g(n))$ if $0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

While big O notation may seem lazy and imprecise, it also often too fine of a measurement

of time complexity. We would like to introduce a fairly robust notion of what it means for an algorithm to be efficient. As hinted at earlier, the big O time complexity of an algorithm depends on the model of computation (RAM model, multi-tape Turning machine, one-tape Turing machine, etc.), so defining an algorithm to be efficient if it runs in $O(f(n))$ for some function f will not be a robust definition unless we establish a particular model of computation. Even if we were willing to restrict to a certain model of computation, we have the issue that an algorithm with an efficient number of steps which calls another efficient algorithm as a subroutine could end up being inefficient. Even more, it is difficult to impossible to justify why an algorithm running in $O(f(n))$ time is efficient while an algorithm running in $O(f(n)^{1+\epsilon})$ is inefficient.

While these challenges to defining efficiency based on a particular big O run time are not insurmountable, they do provide some motivation for considering more robust classes of complexity.

2.2.2 Complexity classes

Rather than continuing to look at all algorithms, we, as is typical, choose to consider only decision problems. A **decision problem** is any yes/no question to be determined by a computer such as

- TRAVELING-SALESMAN: Given a graph G and an integer k, is there a walk on G of length less than or equal to k which passes through every vertex?
- EUL-PATH: Given a graph, is there a walk which passes over every edge exactly once?
- HAM-PATH: Given a graph, is there a walk which passes through every vertex exactly once?
- PRIMES: Given a number n, is n prime?
- GI: Given two graphs G, H, is G isomorphic to H?

To some extent decision problems are simply a convenient formality; however, given the ability to solve a decision problem in $O(f(n))$ time, one can typically solve the more general search problem

(if it exists) with only a small penalty in runtime. For example, consider TRAVELING-SALESMAN, and suppose we can solve TRAVELING-SALESMAN in $O(f(|V|))$ time. Then we can solve the variant of TRAVELING-SALESMAN asking for the smallest walk reaching every vertex in $O(f(|V|)\log(m))$, where m is the length of the shortest such walk by using doubling binary search [14]. Furthermore, in our more robust complexity classes below, this small added penalty in runtime will not affect a problem's complexity classification.

Given a decision problem D, the time complexity of D is defined to be the running time of the fastest worst-case of **any** algorithm which answers D. The complexity class P (called **polynomial time**) is defined to consist of the decision problems which are solvable by deterministic algorithms whose runtimes are polynomial functions. That is, $\mathsf{P} = \bigcup_{c\in\mathbb{N}} \mathsf{DTIME}(O(n^c))$. We claim the class P is fairly good definition of problems which can be solved efficiently. Conveniently, a problem being in P does not depend on the model of computation. Also, P is closed under calls to subroutines in P. But still we should ask: why P when there are problems in P which have cannot be solved faster than $O(n^{100})$ time [69, Theorem 9.10] and a $O(n^{100})$ algorithm is definitely not efficient enough for real life use? Thankfully naturally occurring problems in P requiring $O(n^k)$ time for large k are rare. Also, for a problem to be in P, the algorithm must do something significantly more intelligent than brute force, so being in P is a yardstick for our understanding of the structure of a problem.

The complexity class NP (called **nondeterministic polynomial time**) is the collection of decision problems which are solvable in polynomial time by nondeterministic algorithms. Rather than introduce nondeterminism, we instead note that an equivalent definition of NP is a decision problem D is in NP if there exists a deterministic verifier V running in polynomial time such that for every yes instance of D, there exists a polynomial sized witness w such that when w is given to V as input, V verifies that the answer to D is, in fact, yes. The complexity class coNP is the collection of decision problems whose complements are in NP.

For a few easy examples, we consider the decision problems above. A graph is known to have an Eulerian path if and only if the number of vertices with odd degree is 2 or fewer. Given a graph $G = (V, E)$ as an adjacency matrix (there are similar algorithms for other representations), we can

sum a row of the adjacency matrix to count how many neighbors any one vertex has. Summing each row and counting the number of vertices with an odd number of neighbors gives an algorithm deciding EUL-PATH which runs in $O(|V|^2)$ time. Since the graph was described in $O(|V|^2)$ space, EUL-PATH is in P.

Next, we note that there is unfortunately no known similar characterization for Hamiltonian paths. However, showing HAM-PATH is in NP is still easy. Suppose $G = (V, E)$ is a graph which does have a Hamiltonian path. Let the witness be a list of vertices which form a Hamiltonian path. Since every vertex is visited exactly once, this list has length $|V|$ which is polynomial in size. For the verifier program, we first check that each edge implied by the witness is, in fact, an edge of the graph. Naively, this takes $O(|V| \cdot |E|)$ time. Then we check that every vertex in the graph is listed in the given witness, which can easily be done in $O(|V|^2)$ time. Thus our verifier runs in $O(|V| \cdot |E| + |V|^2) = O(|V|^3)$ time, which is polynomial in the size of G. Thus HAM-PATH is in NP. Noting that our verifier for HAM-PATH can say nothing about graphs which do **not** have a Hamiltonian path, we turn to the next example.

Similar to the requirement for a problem to be in NP, for a problem to be in coNP, we need a polynomial-time verifier and polynomial-sized witnesses. However, we only need to be able to answer "no" rather than "yes" as for NP problems. Considering the PRIMES example, we let our witness be any two factors of n. Then our verifier will multiply the two factors and ensure that the product is the given number n. If the product is not n, we can say nothing about the primality of n. Since the decision problem NOT-PRIME is in NP, we have PRIMES is in coNP. (It turns out PRIMES is actually in P [2], but the proof is much deeper.)

Considering the size of our witness and the runtime of our verifier, we pause to question what the size of the input is. When the input to a decision problem is a single number, we usually describe the input size as the number of bits required to represent it in a computer. That is, the input size is $O(\log(n))$. Since the two factors of n are both smaller numbers, our witness has size $O(\log(n) + \log(n)) = O(\log(n))$. Finally, while performing multiplication as repeated addition is not a polynomial algorithm, the multiplication algorithm typically learned in elementary school is

polynomial in time (even without relying on the assumptions of the RAM model). And faster still are the algorithms encoded in computer processors.

Recalling that we did not account for the memory to store the numbers themselves in the sorting example, we note that the input size would have been more precisely stated as $O(n \log(m))$ where n is the length of the list and m is the largest number in the list. Observing that there is some conflict between the RAM model's ability to perform arithmetic operations on any number in a single step and at the same time accounting for the storage size of numbers, we prevent our algorithm analysis from becoming needlessly complicated by establishing the convention that all numbers in a list (or similar) are stored in constant space. If storage space or encoding style affects results in a non-polynomial way, we will make an explicit note.

Regarding the final example, GI is known to be in NP but not P or coNP. However, GI is known to be another complexity class called coAM. The classes AM and coAM were introduced by Babai and Moran [4, 9] and at essentially the same time by Goldwasser, Micali, and Rackoff [26] with GI as one of the main motivations.[2] The two definitions differed on whether the coin flips were public or private, but Goldwasser and Sipser then showed the two definitions define the same class of problems [27]. The class AM consists of the problems solvable by interactive proofs known as Arthur-Merlin protocols. King Arthur is a mortal who has at his power random coin flips and deterministic algorithms. Merlin, the all-powerful wizard, can answer however he wants. A problem is in AM if both of the following are true.

- If the answer is "yes," then Merlin **can** act in such a way that Arthur says "yes" with probability at least 0.6.[3]
- If the answer is "no," then no matter what Merlin does, Arthur **will** say "no" with probability at least 0.6.

[2] These definitions were the first introductions of interactive proof systems and resulted in the five authors jointly winning the 1993 Gödel Prize.

[3] The number 0.6 can be replaced by any fixed probability strictly larger than 0.5. Then the procedure can be repeated to increase the probability of correctness arbitrarily close to 1. For an example, given a procedure that answers correctly with probability 0.55, we can run it four times in a row to get a procedure that answers correctly with probability $1 - (1 - 0.55)^4 = 0.96$.

Similar to coNP, the complexity class coAM is the class of problems where "no" instances are in AM. As mentioned above, GI is in coAM. But to give a better idea of what AM is, we give a non-rigorous example of a problem in AM.

Consider the problem of a color blind person, Arthur, trying to determine if a red marker and a green marker are actually different colors. His friend Merlin claims to be able to tell the difference. Arthur devises the following decision procedure. He will hold one marker in each hand with only the colored caps visible. He asks Merlin which marker is green and puts both hands behind his back. After either switching them to opposite hands or not (depending on a mental random coin flip), he will show the markers to Merlin and ask which marker is green. Then Arthur will repeat the same test. Arthur will say "yes" the markers are different if Merlin picks the pre-established green marker both times. Arthur will answer "no" the markers are not different otherwise. If the markers really are different, Merlin **can** answer correctly both times, so Merlin **can** act in such a way that Arthur says "yes" with probability 1. If the markers are the same, Merlin will be randomly guessing whether or not Arthur switched hands. Merlin will guess correctly with probability 0.25, so Arthur will say "no" with probability 0.75. Since both 1 and 0.75 are greater than 0.6, this problem is in AM.

Looking back to our idea that P is roughly the class of problems which can be solved efficiently, we get

- NP is the class of problems where we can efficiently verify someone else's claim that an answer is yes.
- coNP is the class of problems where we can efficiently verify someone else's claim that an anwser is no.
- AM is the class of problems where we can efficiently verify someone else's claim that an answer is yes, but only with arbitrarily high probability.
- coAM is the class of problems where we can efficiently verify someone else's claim that an answer is no, but only with arbitrarily high probability.

Considering the relations between these complexity classes, a few things are are clear: $\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathrm{AM}$ and $\mathsf{P} \subseteq \mathsf{coNP} \subseteq \mathsf{coAM}$. None of the preceding containments are known to be strict, and determining if $\mathsf{P} \subsetneq \mathsf{NP}$ is a Clay Millennium Prize Problem.[4]

2.2.3 Reductions and $\mathcal{C}$-hard problems

Earlier, we saw $\mathsf{EUL\text{-}PATH} \in \mathsf{P}$ and $\mathsf{HAM\text{-}PATH} \in \mathsf{NP}$. However, $\mathsf{HAM\text{-}PATH}$ is not known to be in P. In fact, $\mathsf{HAM\text{-}PATH} \in \mathsf{P}$ if and only if $\mathsf{P} = \mathsf{NP}$. We would like to say $\mathsf{HAM\text{-}PATH}$ is at least as hard as $\mathsf{EUL\text{-}PATH}$, which leads us develop a way to compare difficulties of decision problems.

Suppose A, B are two decision problems. Consider a computational model M^A which has an extra instruction allowing it solve any instance of A in a single step. Since polynomial time algorithms are our idea of efficient, we will say B is **polynomial-time Turing reducible** (or **Cook reducible**) to A, written $B \leq_T^P A$, if M^A can solve B in polynomial time. Making the restriction that we can solve an instance of A only once during our algorithm and only as the final step, we get that B is **polynomial-time many-one reducible** (or **Karp reducible**) to A, written $B \leq_m^P A$. If $B \leq_m^P A$, we say A is B**-hard** or A is **at least as hard** as B. The idea behind many-one reducibility is that B is many-one reducible to A if we can, in polynomial time, change any instance of B into an instance of A where the instances of A and B have the same answer. Extending hardness to complexity classes, we say a problem A is $\mathcal{C}$**-hard** for any complexity class $\mathcal{C}$ if $B \leq_m^P A$ for every $B \in \mathcal{C}$. That is, A is $\mathcal{C}$-hard if A is at least as hard as every problem in $\mathcal{C}$. Additionally, if we have A is both $\mathcal{C}$-hard and $A \in \mathcal{C}$, then A is $\mathcal{C}$**-complete**. The most important class of complete problems are NP-complete problems. Considering our examples again, both $\mathsf{HAM\text{-}PATH}$ and $\mathsf{TRAVELING\text{-}SALESMAN}$ are NP-complete. At the end of the previous section, we mentioned one of the most important open problem in mathematics of whether or not $\mathsf{P} = \mathsf{NP}$. With our new vocabulary, we note that $\mathsf{P} = \mathsf{NP}$ if a single NP-complete problem is shown to be in P, and $\mathsf{P} \neq \mathsf{NP}$ if it can be shown that a single NP-complete problem is not

[4] with a $1 million prize attached

contained in P. If there exists problems in NP \ P which are not NP-complete, they are called **NP-intermediate** problems. If fact, Ladner's theorem says NP-intermediate problems do exist if P ≠ NP [43]. Many attempts to resolve the P vs NP question have centered around the hope that GI is NP-intermediate as GI not known to be in P nor is it known to be NP-complete. As such, a complexity class of problems, which will be important later, is defined around GI. As is standard, we abuse notation to define the complexity class GI by $\mathsf{GI} = \{A : A \leq_T^P \mathsf{GI}\}$. Many problems known to be GI-complete are restrictions or generalizations of GI. Of particular concern in Chapter 3 are the GI-complete problems of determining isomorphism between directed graphs and determining isomorphism between directed multigraphs.

2.2.4 Decidability

So far we have only introduced decision problems which have nice property that there is an obvious brute force option to solve it. While brute force algorithms typically run in $O(c^n)$ time, and hence are impractical for use, decision problems need not be solvable by any algorithm at all. A decision problem D is **decidable** if there is an algorithm A such that when A is given any instance of D, A is guaranteed to determine the correct answer in a finite amount of time. A decision problem which is not decidable is called **undecidable**. The following problems are all undecidable.

- HALT: Given the description of the Turing machine M and some input w, will M halt or run forever when given w as input?

- A group G is a **finitely presentable group** if there exist a finite set of generators S and finite set of relations R such that $G = \langle S \mid R \rangle$.

  GpI-FinitelyPresented: Given two finitely presentable groups G, H, is $G \cong H$? [55, Corollary 3.5].

- A **Markov property** P of a finitely presentable group is any property of a group such that:

  (1) P is preserved under group isomorphism.

(2) There exists a finitely presentable group with property P.

(3) There exists a finitely presentable group which cannot be embedded in any finitely presentable group with property P.

HasMarkovProperty: Given a group G and a Markov property P, does G have P? [1, 61].

2.3 Symbolic dynamics

In Chapter 3, the main focus will be discrete dynamical systems (discrete in both space/state and time), known as shift spaces. Of particular concern are shifts of finite type (SFTs). For a more in-depth introduction to symbolic dynamics, see any introductory textbook to the field such as [51, 40]. Fix an **alphabet** $\mathcal{A} = \{1, 2, \ldots, n\}$. A **bi-infinite sequence** in $\mathcal{A}$ is any sequence with entries in $\mathcal{A}$ indexed by $\mathbb{Z}$ (i.e., a sequence which is infinite in two directions). Given a bi-infinite sequence $\cdots a_{-2}a_{-1}.a_0a_1a_2\cdots$, where the period is placed between index -1 and index 0, the **shift operator** σ shifts the sequence to the left by 1. That is, $\sigma(\cdots a_{-2}a_{-1}.a_0a_1a_2\cdots) = \cdots a_{-1}a_0.a_1a_2a_3\cdots$. Then for any fixed alphabet $\mathcal{A}$, a **shift space** (or **shift**) is any collection of bi-infinite sequences from $\mathcal{A}$, which is closed under σ (and σ^{-1}).

For an example, consider the set of all bi-infinite sequences with entries in alphabet $\mathcal{A} = \{0, 1\}$. This is a shift space called the **full shift on two letters** (or the **full 2-shift**) and is denoted X_{full} (where $\mathcal{A}$ is implied) or $\{0, 1\}^{\mathbb{Z}}$. For better terminology, any bi-infinite sequence in $\mathcal{A}^{\mathbb{Z}}$ (for any alphabet $\mathcal{A}$) is called a **point**. Any finite sequence of letters from $\mathcal{A}$ is a **block** (or **word**). Extending this terminology, a sequence of characters which is infinite in exactly one directions will be called an **infinite word**. For any point $p \in \mathcal{A}^{\mathbb{Z}}$, the subword of p from index i to j is the word $p_{[i,j]} = p_i \cdots p_j$. In a similar fashion, we define the sub-infinite words $p_{[i,\infty)}$ and $p_{(\infty,i]}$ in the obvious way. Collecting all blocks which are subwords of some p in a fixed shift space X, we have $\mathcal{B}_n(X) = \{w : \text{there exists } p \in X, i \in \mathbb{Z} \text{ such that } p_{[i,i+(n-1)]} = w\}$.

If a shift X is contained in the shift Y, then X is called a **subshift** of Y. In particular, consider the shift $X_{(01)^*}$ consisting of points in $\{0, 1\}^{\mathbb{Z}}$ where no two 1s nor no two 0s are adjacent.

Then $X_{(01)^*} = \{\cdots 010.101 \cdots, \cdots 101.010 \cdots\}$ and $X_{(01)^*}$ is a subshift of $\{0,1\}^{\mathbb{Z}}$.

Furthermore, after fixing an alphabet $\mathcal{A}$, consider a list of words $\mathcal{F}$ in $\mathcal{A}$. Then the collection of points from $\mathcal{A}^{\mathbb{Z}}$ which contain no word in $\mathcal{F}$ as a subword form a subshift of $\mathcal{A}^{\mathbb{Z}}$ denoted $X_{\mathcal{F}}$. The words in this list $\mathcal{F}$ are called the **forbidden blocks** of $X_{\mathcal{F}}$. In fact, every shift space can be written as $X_{\mathcal{F}}$ for some set of forbidden blocks. Note that for any shift other than a full shift, there are many possible collections of forbidden blocks. In particular, consider $\mathcal{F}_1 = \{00, 11\}$ and $\mathcal{F}_2 = \{000, 001, 011, 110, 111\}$; both $X_{\mathcal{F}_1}$ and $X_{\mathcal{F}_2}$ are descriptions of $X_{(01)^*}$. While these lists of forbidden blocks are finite, we could have written a description of $X_{(01)^*}$ by using the infinite set of forbidden blocks $\mathcal{F}_3 = \{00w, 11w : w \text{ is any block in the alphabet } \{0,1\}\}$.

So far every example shift has had a description whose list of forbidden blocks is finite; this need not be the case. For example, consider the shift $X_{\mathcal{F}_4}$ where $\mathcal{F}_4 = \{10^k1 : k \text{ is odd}\}$. There is no way to describe $X_{\mathcal{F}_4}$ with a set of forbidden blocks where the set of forbidden blocks is finite in size (see [51, Example 2.1.5] for a proof). In the special case where the shift X can be represented by a finite list of forbidden blocks, X is called a **shift of finite type** (SFT). SFTs will be the primary focus of Chapter 3.

Even though most shift spaces are infinite in size, there are frequently concise, even finite, representations. Given a directed graph $G = (V, E)$ with labeled vertices (each distinct), we associate to it the shift space $X_G = \{(v_i)_{i\in\mathbb{Z}} : v_i \in V, (v_i, v_{i+1}) \in E \text{ for all } i \in \mathbb{Z}\}$, which is the collection of all bi-infinite walks on G. Note that X_G is a shift of finite type with $\mathcal{F} = \{v_iv_j : (v_i, v_j) \notin E\}$. Any shift of this form is called a **vertex shift**. Not only is every vertex shift a SFT, but also, every SFT is conjugate to a vertex shift (but possibly over a different alphabet). See Figure 2.4 for examples of vertex shifts.

Now consider a directed multigraph $G = (V, E)$ with labeled edges (each distinct). Associated to G is the shift space X_G^e called the **edge shift** of G where the points are the bi-infinite walks on G. Again, note that X_G^e is a SFT where $\mathcal{F} = \{e_1e_2 : e_1 \text{ does not terminate at the initial vertex of } e_2\}$. As with vertex shifts, we also have that every SFT is conjugate to an edge shift. See Figure 2.5 for examples of edge shifts.

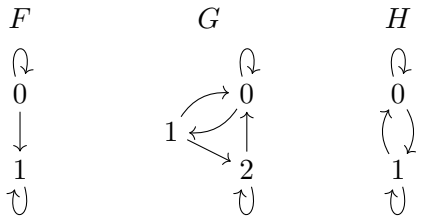

Figure 2.4: Three examples of vertex shifts. X_G and X_H are conjugate via the 1-block code mapping $1, 2 \mapsto 1, 0 \mapsto 0$ as 1 and 2 can be amalgamated in G to form H.

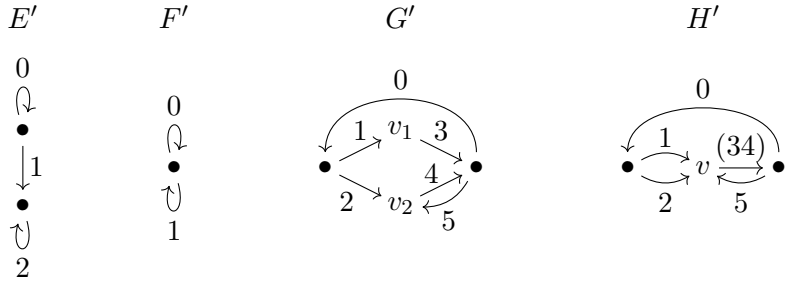

Figure 2.5: Four examples of edge shifts. $X_{G'}$ and $X_{H'}$ are conjugate as G' can be transformed into H' by amalgamating vertices v_1 and v_2.

Consider a vertex in a directed graph (or multigraph) with either no in-neighbor or no out-neighbors. Such a vertex is called **stranded**. A directed graph (or multigraph) with no stranded vertices is called **essential**. Note that no point in X_G (or X_G^e) can pass through a stranded vertex and that stranded vertices can be efficiently trimmed from G (in an iterative fashion) to create an essential graph. Thus we can assume without loss of generality that any directed graph (or multigraph) representing a vertex (or edge) shift is essential. Additionally, we restrict our attention to directed graphs (or multigraphs) which are connected. For the problems of finding or verifying a conjugacy between vertex (or edge) shifts, the function $\Phi_\infty : X_G \to X_H$ is a conjugacy if and only if for each strongly connected component G_i of G, Φ_∞ restricted to a G_i as $\Phi_\infty|_{G_i} : X_{G_i} \to X_{H_i}$ is a conjugacy. Furthermore, as all graphs (and similarly for multigraphs) in this section (and again in Chapter 3) are directed, we will often write "graph" instead of "directed graph."

As is discussed in §3.7, there are cases where representing a shift space by a vertex shift may take exponentially more space that representing the same shift as an edge shift. In particular, consider the full d-shift. As an edge shift, the adjacency matrix of the smallest graph representing this shift is a 1×1 matrix with d as the single entry; however, as a vertex shift, the adjacency matrix of the smallest graph representing this shift is a $d \times d$ matrix where every entry is a 1. Note also that shift X_H in Figure 2.4 and shift $X_{F'}$ in Figure 2.5 are the minimal representations of the full 2-shift. In §3.4, we look at the problem of finding a "smaller" representation of SFT (see Theorem C) which is conjugate to a given SFT. Noting that vertex shifts and edge shifts need not specify a labeling as the labels are distinct, it is tempting to focus on finding small edge shift representations. Complicating this matter further, however, is that there are cases where every edge shift representation of a shift space requires a larger alphabet than the smallest vertex shift representation.[5] For an example, consider the shifts X_F in Figure 2.4 and $X_{E'}$ in Figure 2.5. The shift spaces X_F and $X_{E'}$ are conjugate, but there is no edge shift on two letters[6] which is conjugate to this shift. In general, "smaller" representations is ill-defined unless restricting to vertex shifts,

[5] For every edge shift there is a conjugate vertex shift with the same alphabet.
[6] To verify this fact, it is easy to check each essential edge shift on two letters.

and talking about the complexity of algorithms for edge shifts (at all) is quite nuanced. As such, we mainly focus on vertex shifts. (Note that §3.5 and §3.6 are exceptions.)

Definition 2.3.1. A shift X is **irreducible** if for every pair of words w_1, w_2 in X, there is a word w_3 such that $w_1w_3w_2$ is a word in X, and X is **reducible** if it is not irreducible. In graph-theoretic terms, a vertex shift X_G (or edge shift X_G^e) is irreducible if and only if G is strongly connected.

For examples, note that shifts X_G, X_H from Figure 2.4 and $X_{F'}, X_{G'}, X_{H'}$ from Figure 2.5 are irreducible while shifts X_F and $X_{E'}$ are reducible.

Given a shift X with alphabet $\mathcal{A}_1$, we can transform X into a shift space over another alphabet $\mathcal{A}_2$ in the following way. Fix integers m, a with $-m \leq a$. Then letting $\mathcal{B}_n(X)$ denote the set of blocks of size n from the shift X and given a function $\Phi : \mathcal{B}_{m+a+1}(X) \to \mathcal{A}_2$, the corresponding **sliding block code** with memory m and anticipation a is the function Φ_∞ defined by $\Phi_\infty((x_i)_{i\in\mathbb{Z}}) = (\Phi(x_{[i-m,i+a]}))_{i\in\mathbb{Z}}$. That is, Φ_∞ looks at a block of size $m + a + 1$ through a window to determine a character from $\mathcal{A}_2$. Then the window is slid infinitely in both directions. Letting $k = m + a + 1$, we will call any sliding block code with window size k a **k-block code**. Given a sliding block code as $\Phi : \mathcal{A}_1^k \to \mathcal{A}_2$, we extend Φ to all finite and infinite words w of length at least k by $\Phi((w_i)_{i\in I}) = (\Phi(w_{[i-m,i+a]}))_{i-m,i+a\in I}$, where $I \subsetneq \mathbb{Z}$. That is, we extend Φ to words by sliding Φ over the entire word. Up to this point, we've been using the word conjugacy without formally defining it. A sliding block code $\Phi_\infty : X \to Y$ is a **conjugacy** if it is both injective and surjective. If there exists a conjugacy $\Phi_\infty : X \to Y$, X and Y are called **conjugate** shift spaces and we write $X \cong Y$.

Pausing to motivate sliding block codes, we note that any sliding block code commutes with the shift map; that is, for any 1-block code $\Phi_\infty : X \to Y$, we have $\Phi_\infty \circ \sigma_X = \sigma_Y \circ \Phi_\infty$ [51, Proposition 1.5.7]. Even more, any shift space is a metric space with metric

$$\rho(x, y) = \begin{cases} 2^{-k} & \text{if } x \neq y \text{ and } k \text{ is maximal so that } x_{[-k,k]} = y_{[-k,k]}, \\ 0 & \text{if } x = y. \end{cases}$$

While the details of this metric will not be important for our purposes, we will use that the induced topological space is compact [51, Example 6.1.17(5)]. Furthermore, we have the following theorem.

Theorem 2.3.2 (Curtis–Hedlund–Lyndon Theorem [30])**.** *A function $\varphi : X \to Y$ is continuous with respect to the topology induced by ρ if and only if φ is a sliding block code.*

Let X be any shift space with alphabet $\mathcal{A}_1$. We define the kth higher block shift $X^{[k]}$ (which is conjugate to X) with alphabet $\mathcal{A}_2 = \mathcal{B}_k(X)$ by the image of X under $\beta_k : X \to (\mathcal{B}_k(X))^{\mathbb{Z}}$ where for any point $p \in X$, $\beta_k(p)_i = p_{[i,i+k-1]}$. If $X = X_G$ happens to be a vertex shift, we can construct the kth higher block shift in terms of the graph. For any directed graph G, construct the graph $G^{[k]}$ by $V_{G^{[k]}} = \{v_1 \cdots v_k : v_1 \cdots v_k \text{ is a path in } G\}$ and $E_{G^{[k]}} = \{(v_1 v_2 \cdots v_k, v_2 \cdots v_k v_{k+1}) : v_1 \cdots v_{k+1} \text{ is a path in } G\}$. Then $X_{G^{[k]}} = X_G^{[k]}$. When dealing with k-block codes, it is often useful to **pass to a higher block shift** by noting that there is a k-block conjugacy $\Phi_\infty : X \to Y$ if and only if there is a 1-block conjugacy $\Phi_\infty^{[k]} : X^{[k]} \to Y$ [51, Proposition 1.5.12].

Furthermore, any sliding block code $\Phi_\infty : X_G \to X_H$ between vertex shifts induces the function $\Phi_{\mathsf{c}} : \bigcup_{n=1}^{\infty} C_n(G) \to \bigcup_{n=1}^{\infty} C_n(H)$ defined as follows. Given a cycle c in G, there is a unique cycle d in H with $|c| = |d|$ such that $\Phi_\infty(c^\infty) = d^\infty$; we set $\Phi_{\mathsf{c}}(c) = d$. In the special case of a 1-block code, the block map $\Phi : \mathcal{A}_1 \to \mathcal{A}_2$ is simply a map between the alphabets. In this case, when X_G is a vertex shift, we have $\Phi_{\mathsf{c}}(v_1 \cdots v_n) = \Phi(v_1) \cdots \Phi(v_n)$.

Definition 2.3.3. Let X_G be a vertex shift. We say states $u, v \in V_G$ can be **amalgamated** if one the following conditions is met.[7]

(1) $N^+(u) = N^+(v)$ and $N^-(u) \cap N^-(v) = \emptyset$

(2) $N^-(u) = N^-(v)$ and $N^+(u) \cap N^+(v) = \emptyset$

If u and v are amalgamated, they are replaced by the vertex uv which has $N^+(uv) = N^+(u) \cup N^+(v)$ and $N^-(uv) = N^-(u) \cup N^-(v)$. Dual to the notion of amalgamation is the notion of splitting:

[7] We remind the reader that $N^-(u)$ is the set of in-neighbors of u and $N^+(v)$ is the set of out-neighbors of v as defined in §2.1.

Definition 2.3.4. Let X_G be a vertex shift. A vertex $v \in V_G$ can be split into two vertices v_1 and v_2 provided the edges of v_1, v_2 satisfy one of the following conditions.

(1) $\{N^+(v_1), N^+(v_2)\}$ is a partition of $N^+(v)$ and $N^-(v_1) = N^-(v_2) = N^-(v)$.

(2) $\{N^-(v_1), N^-(v_2)\}$ is a partition of $N^-(v)$ and $N^+(v_1) = N^+(v_2) = N^+(v)$.

The corresponding new graph is called a **state splitting** of v. Note that state splittings and amalgamations are inverse operations. For an example, consider shifts X_G and X_H in Figure 2.4. As G can be transformed into H by amalgamating vertices 1 and 2, we also have H can be transformed into G by splitting vertex 1.

The definitions for edge shifts are similar. Since edges shifts are based on multigraphs, $N^-(v)$ and $N^+(v)$ are multisets. The definition of a state splitting is identical noting that the partition is a multiset partition. For amalgamations, two vertices u, v can be amalgamated if $N^-(u) = N^-(v)$ or $N^+(u) = N^+(v)$. In the case where $N^-(u) = N^-(v)$, u, v are replaced by a single vertex uv with $N^-(uv) = N^-(u) = N^-(v)$ and $N^+(uv) = N^+(u) \uplus N^+(v)$, where $N^+(u) \uplus N^+(v)$ is the multiset disjoint union. For an example, consider shifts $X_{G'}$ and $X_{H'}$ in Figure 2.5. As G' can be transformed into H' by amalgamating vertices v_1 and v_2, we also have H' can be transformed into G' by splitting vertex v.

Theorem 2.3.5 ([74, 51])**.** *Let X_G, X_H be vertex shifts (or edge shifts). Then X_G and X_H are conjugate if and only if there is a sequence of state splittings followed by a sequence of amalgamations which transform G into H.*

In the case of a 1-block code $\Phi : V_G \to V_H$, we may view the block map as a partition of the vertices of G, where each element of the partition is converted to a vertex of H. In light of Theorem 2.3.5, it may be tempting to think that every 1-block conjugacy can be written as a sequence of amalgamations only, as intuitively splitting a vertex while requiring the vertices be re-amalgamated has no benefit. Yet this statement is not true; there are simple examples of two

graphs admitting a 1-block conjugacy, where no pair of vertices can be amalgamated in either graph (Figure 2.6).

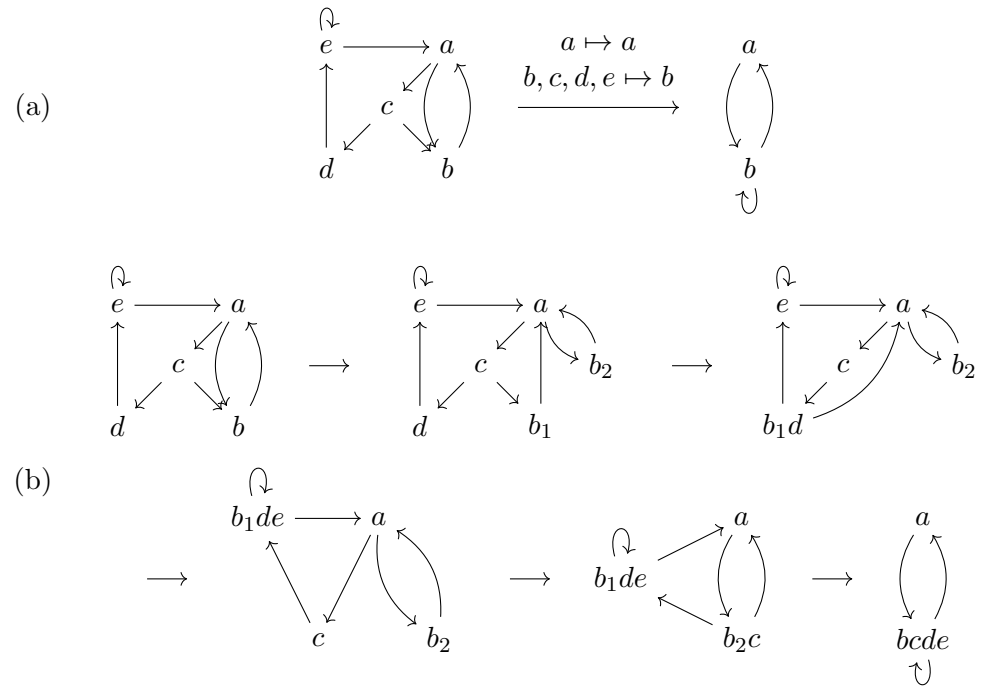

Figure 2.6: (a) A minimal example of two vertex shifts which are conjugate by a 1-block code but not by a sequence of amalgamations. (b) The conjugacy, demonstrated via a splitting followed by four amalgamations.

2.4 Tensors

2.4.1 Introduction

In chapter 4, we look at the problem of determining isomorphism between groups. To do this, we translate the problem of isomorphism of p-groups of exponent p and nilpotence class 2 to the problem of isomorphism of a subclass of tensors. We now introduce tensors and the tensor isomorphism problem.

A d-**tensor** is a multilinear map $T : V_1 \times V_2 \times \cdots \times V_{d-1} \rightarrowtail V_d$ where the V_i are vector spaces over the field $\mathbb{F}$. Note that we are using the symbol $\rightarrowtail$ to mean a map is multilinear. If T is a d-tensor, we say T has **valence** d. For examples, 1-tensors are vectors and 2-tensors are

linear maps. Noting that tensors are defined as multilinear maps independent of basis, we say two tensors T, T' are **isomorphic** (this is also called **tensor equivalence**) if, after picking bases for T, T', there is a change of basis for T which transforms it to be T'. Going back to our examples, two 1-tensors v, v' are isomorphic if and only if they are both non-zero or both zero. Similarly, recall that linear maps are simply matrices. Then two 2-tensors M, M' are isomorphic if and only if they have the same rank.

Valence 1 and 2 tensors are well-understood as the content of a standard undergraduate course in linear algebra. Increasing in valence, we note that tensors of valence higher than 3 exist and are highly important; however, we only need valence 3 tensors for our context, so our remaining examples and definitions will be tailored to the valence 3 case. Furthermore, concerning ourselves with the computational problem of determining tensor equivalence, isomorphism of valence d tensors reduces to the case of isomorphism of valence 3 tensors for all $d \geq 1$ [29].

2.4.2 Valence 3 tensors

Restricting to the case of valence 3, a valence 3 tensor is a bilinear map $T : U \times V \rightarrowtail W$, where we write either $(u, v) \mapsto T(u, v)$ or $(u, v) \mapsto uTv^T$. Motivating the second choice of notation, pick a basis for T and consider the special case where W is a field and U, V are vectors spaces over W. Then $T : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}$ is given by a matrix, and for any vectors $u \in U, v \in V$, we have $T(u, v) = uTv^T$ (as multiplication of matrices, where we note our convention that vectors are row vectors). Relating back to other commonly studied objects, the valence 3 tensors with $U = V = \mathbb{F}^n$ and $W = \mathbb{F}$ are the bilinear forms over $\mathbb{F}$.

Pick a basis for the 3-tensor T where $T : \mathbb{F}^n \times \mathbb{F}^m \rightarrowtail \mathbb{F}^\ell$. We will also call T a $(n \times m \times \ell)$-tensor as $T = (T_{ijk})_{(i,j,k)\in[n]\times[m]\times[\ell]}$ where $[n] = \{1, 2, \ldots, n\}$, i.e., T can be written as a $n \times m \times \ell$ cube of field elements. Slicing this cube in any direction gives an ordered list of matrices. Fixing an orientation to 3-tensors, we think of this vector having height n, width m, and depth ℓ. Of particular concern will be the slices of T such that there are ℓ slices and each slice is a $n \times m$ matrix. These slices are notated $\{T_{**k}\}_{k=1\ldots,\ell}$. Then to evaluate $T(u, v)$ for any $u \in \mathbb{F}^n, v \in \mathbb{F}^m$, we

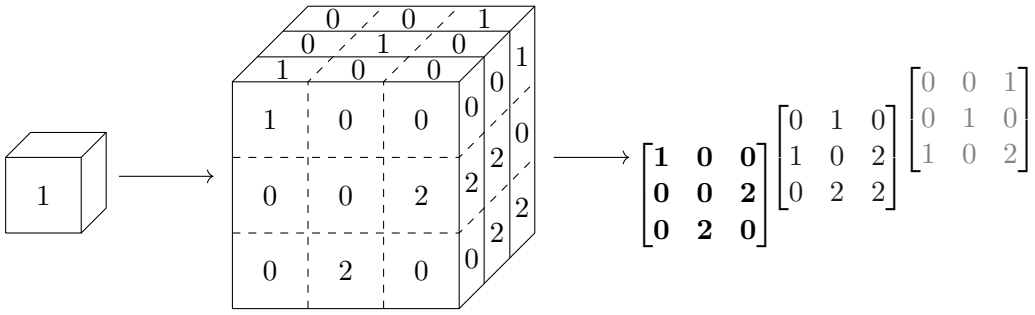

Figure 2.7: An example of the multiplication tensor over $\mathbb{F}_{5^3} = \mathbb{F}_5[x]/(x^3 - 2x - 2)$ being embedding into $\mathbb{F}_5$ and then sliced front to back.

have $T(u, v) = \langle uT_{**1}v^T, \ldots, uT_{**\ell}v^T \rangle \in \mathbb{F}^\ell$. For example, consider the tensor in Figure 2.7. Then letting $u = \langle 1, -1, 2 \rangle, v = \langle 1, 0, -1 \rangle$,

$$T(u,v) = uTv^T \tag{2.1}$$

$$= \left\langle uT_{**1}v^T, uT_{**2}v^T, uT_{**3}v^T \right\rangle \tag{2.2}$$

$$= \left\langle u \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 2 & 0 \end{bmatrix} v^T, u \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 2 & 2 \end{bmatrix} v^T, u \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix} v^T \right\rangle \tag{2.3}$$

$$= \langle -2, 2, 2 \rangle. \tag{2.4}$$

Recalling that we can change the basis of the vector space $\mathbb{F}^n$ by multiplying every vector by an element $A \in \mathrm{GL}_n(\mathbb{F})$, we change the basis of a tensor T in the input coordinates by

$$uT_{**k}v^T \mapsto (uA)T_{**k}(vB)^T = u(AT_{**k}B^T)v^T \text{for all } k.$$

Also, from studying matrices, we know the transformation $M \mapsto AMB^T$ for arbitrary $(A, B) \in \mathrm{GL}_n(\mathbb{F}) \times \mathrm{GL}_m(\mathbb{F})$ corresponds to doing arbitrary row and column operations to M. Thus we have that changing the basis in the input coordinates corresponds to doing arbitrary row and column operations with the added restriction that the same operations must be done to every slice. That is, we are allowed to add row 3 to row 1 under the condition that we add row 3 to row 1 in every

single slice at the same time. We will denote a change of basis of this form by $T \mapsto ATB^T$ (or $T_{**k} \mapsto AT_{**k}B^T$). Changing basis in the output coordinates, however, is more complicated as our tensor seems to be sliced in the wrong direction to say anything. To correct this, we can "glue" the slices of T back together, slice it in either of the other directions, apply a change of basis in the third direction, "glue" it back together, and slice it front to back again. While this certainly works, we instead note that a change of basis in the third direction is equivalent to performing **slice operations**, which, similar to row operations, are the operations:

- Multiply any one slice T_{**k} by a scalar $\alpha \in \mathbb{F} \setminus \{0\}$.
- Swap any two slices T_{**k} and $T_{**k'}$.
- For any scalar $\alpha \in \mathbb{F} \setminus \{0\}$, add αT_{**k} to the slice $T_{**k'}$.

In the case where only the slice operation of swapping slices is used, we call the basis change a **shuffle** of the slices. We will denote any change of basis in the third direction by $T_{**k} \mapsto \sum_{k'} \alpha_{k'} T_{**k'}$.

2.4.3 Symmetric and alternating tensors

Sometimes, similar to bilinear forms, tensors have nice properties which we would like to keep invariant under change of basis. For example, in bilinear forms, b is symmetric if $b(u, v) = b(v, u)$ for all $u \in U, v \in V$ and b is alternating if $b(v, v) = 0$ for all $v \in V$. To generalize either of these to tensors, we need that both input vector spaces are the same; that is, we must restrict to tensors $T : V \times V \rightarrowtail W$. Furthermore, if we chose a basis for V in one coordinate, we must choose the same basis for V in the other coordinate. Then we define symmetric 3-tensors and alternating 3-tensors in the same fashion as bilinear forms. Namely, a 3-tensor is **symmetric** if $T(u, v) = T(v, u)$ for all $u, v \in V$ and a 3-tensor is **alternating** if $T(v, v) = 0$ for all $v \in V$. Picking a basis, we have in our evaluation notation that T is symmetric if and only if for each k, $uT_{**k}v^T = vT_{**k}u^T$ for all $u, v \in V$; that is, T is symmetric if and only if each slice T_{**k} is a symmetric matrix. Note that the tensor in Figure 2.7 is symmetric. Similarly, T is alternating if and only if each slice T_{**k} is skew-symmetric ($T_{**k} = -T_{**k}^T$) with 0s on the diagonal.

To ensure that a tensor being symmetric or alternating is a property independent of any basis, we must restrict the legal basis changes. Suppose $T : V \times V \rightarrowtail W$ is either symmetric or alternating. Then changing the basis of V by multiplying every vector by $A \in \mathrm{GL}_n(\mathbb{F})$ gives

$$uTv^T \mapsto (uA)T(vA)^T = u(ATA^T)v^T.$$

In terms of row and column operations, these basis changes correspond to performing any row operation and then following it up with the same column operation. For example, first add row 3 to row 1 in every slice. Then we must immediately add column 3 to column 1 in every slice. Thankfully, W is allowed to be an arbitrary vector space over $\mathbb{F}$, so the legal slice operations for symmetric or alternating tensors are unaffected. When making the transformation $T \mapsto ATA^T$ via a row operation followed by the mandatory column operation, we will call this simultaneous operation a **row/column operation**. In Chapter 4, we will primarily (but not universally) be concerned with alternating tensors and these restricted basis changes. (The problem of determining if two alternating tensors are equivalent under these restricted basis changes is known as **pseudo-isometry** (of bilinear maps) or **alternating matrix space isometry**.)

2.4.4 Writing a tensor over $\mathbb{F}_q$ as a tensor over $\mathbb{F}_p$

So far, we have made no restrictions on $\mathbb{F}$. In Chapter 4, all of our tensors will be defined over finite fields. Suppose $T : \mathbb{F}_{p^n}^c \times \mathbb{F}_{p^n}^c \rightarrowtail \mathbb{F}_{p^n}^d$ is a symmetric or alternating 3-tensor. As vector spaces, $\mathbb{F}_{p^n}$ is also a n dimensional vector space over $\mathbb{F}_p$. We will introduce a canonical way to transform T from a chosen $\mathbb{F}_{p^n}$-basis to be a $(cn \times cn \times dn)$-tensor over $\mathbb{F}_p$. Thus given a second tensor $T' : \mathbb{F}_{p^m}^{c'} \times \mathbb{F}_{p^m}^{c'} \rightarrowtail \mathbb{F}_{p^m}^{d'}$, we can write both tensors over $\mathbb{F}_p$ and ask if T, T' are isomorphic over $\mathbb{F}_p$.

To do this, we "blow up" each entry from $\mathbb{F}_{p^n}$ to be a $n \times n \times n$ block of numbers from $\mathbb{F}_p$. Once we can transform any $(1 \times 1 \times 1)$-tensor over $\mathbb{F}_{p^n}$ to be a $(n \times n \times n)$-tensor over $\mathbb{F}_p$, we will "glue" the "blown up" entries of a larger tensor together in the obvious way. Unfortunately, this process depends on the representation of $\mathbb{F}_{p^n}$. Fix any representation of $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[x]/(a(x))$, and

pick the basis of $\mathbb{F}_{p^n}$ as a n-dimensional vector space over $\mathbb{F}_p$ to be $(1, x, x^2, \ldots, x^{n-1})$. Consider any $\alpha = \alpha_0 + \alpha_1 x + \cdots \alpha_{n-1} x^{n-1} \in \mathbb{F}_{p^n}$; we define the projection functions $\pi_i : \mathbb{F}_{p^n} \to \mathbb{F}_p$ by $\pi_i(\alpha) = \alpha_i$. Letting T be the $(1 \times 1 \times 1)$-tensor with α as its single entry, we construct the corresponding $(n \times n \times n)$-tensor T' by setting $T'_{ijk} = \pi_k(x^{i-1} \cdot \alpha \cdot x^{j-1})$ for $i, j, k \in \{1, \ldots, n\}$. Then for $\beta, \gamma \in \mathbb{F}_{p^n}$,

$$\pi_i(\beta T \gamma^T) = \beta T'_{**i} \gamma^T.$$

For an example, see Figure 2.7, where the $(1 \times 1 \times 1)$-tensor with entry $1 \in \mathbb{F}_5[x]/(x^3 - 2x - 2)$ is embedded as $(3 \times 3 \times 3)$-tensor over $\mathbb{F}_5$; continuing our example with those tensors, we denote them by S and S' respectively. Note that S encodes multiplication, i.e., $S(\beta, \gamma) = \beta \cdot \gamma$, so we have that S' also encodes the multiplication structure of $\mathbb{F}_{p^n}$. For an example, consider that $(1 - x + 2x^2) \cdot (1 - x^2) = (-2 + 2x + 2x^2)$ in $\mathbb{F}_p[x]/(x^3 - 2x - 2)$. By the calculation in (2.1)–(2.4), we see $S'(\langle 1, -1, 2\rangle, \langle 1, 0, -1\rangle) = \langle -2, 2, 2\rangle$, so this particular multiplication (and, in fact, every $\mathbb{F}_{p^n}$ multiplication) is encoded in S'.

2.5 Finite group theory

In Chapter 4, the main focus will be the group isomorphism problem, which given two groups G, H asks: Is $G \cong H$? We now introduce enough definitions to give an intuitive explanation of our problem, and then discuss some technicalities regarding representing groups in computers.

2.5.1 Background

Of particular concern will be p-groups of exponent p which are nilpotent of class 2. A group G is called a **p-group** if $|G| = p^\alpha$ for some prime p. The **exponent** of G is the smallest e such that $G^e = \{g^e : g \in G\} = \{1\}$. A **central series** for a group G is a series

$$\{1\} = A_0 \trianglelefteq A_1 \trianglelefteq \cdots \trianglelefteq A_n = G$$

of subgroups A_i of G such that $[G, A_i] \leq A_{i-1}$. A group G is **nilpotent** if G has a central series and is **nilpotent of class** c (or simply **class** c) if its shortest central series has length c. As an

alternate definition specific to groups of nilpotence class 2, G is nilpotent of class 2 if G is nonabelian and $G' \leq Z(G)$. Pausing to motivate the study of isomorphism restricted to class 2 p-groups, we note there is theoretical evidence, beyond working with practical algorithms, which suggests this case of isomorphism is hard. First, classifying p-groups is a wild problem (in the technical sense of "wild") [68]. Even more, there is a polynomial time reduction from the case of p-groups of exponent p and nilpotence class $c < p$ to the case of p-groups of exponent p and nilpotence class 2 [29]. Using three known results regarding the structure of nilpotent groups, we reduce isomorphism in class 2 nilpotent groups to an even nicer class of groups.

Proposition 2.5.1.1 ([36, Theorem 1.26])**.** *A finite group G is nilpotent if and only if G splits as the direct product of its Sylow p-subgroups.*

Proposition 2.5.1.2 ([77])**.** *Determining whether a group G is a nontrivial direct product, and, if it is, finding G_1, G_2 such that $G \cong G_1 \times G_2$ can be done in polynomial time.*

Proposition 2.5.1.3 ([47, Exercise 1.2(1)])**.** *Suppose G is a p-group of exponent p such that $G' \lneq Z(G)$. Then $G \cong \tilde{G} \times A$ where A is abelian and $Z(\tilde{G}) = \tilde{G}' \cong G'$.*

That is, Propositions 2.5.1.1 and 2.5.1.2 say that in order to determine isomorphism of nilpotent groups in polynomial time, it is sufficient to determine isomorphism of p-groups in polynomial time. As discussed above, it is commonly believed that determining isomorphism among p-groups of class 2 and exponent p is the most difficult case of group isomorphism. Chapter 4 is dedicated to such groups, and Propositions 2.5.1.3 and 2.5.1.2 say it is sufficient to add the assumption that $G' = Z(G)$ (these groups are also called the nonabelian special p-groups). Finally, considering that all 2-groups of exponent 2 are abelian, we assume p is an odd prime for the remainder of this thesis unless explicitly stated otherwise.

2.5.2 A brief introduction to genus 2 groups

Under the restriction to p-groups of class 2 and exponent p, we now give an intuitive definition of the genus of a group. (For a rigorous definition, see §4.2.) For a directly indecomposible p-group

G of exponent p with $G' = Z(G)$, G' is elementary abelian; that is, $G' = Z(G) \cong \mathbb{Z}/p\mathbb{Z} \times \cdots \mathbb{Z}/p\mathbb{Z}$. Since $\mathbb{F}_p \cong \mathbb{Z}/p\mathbb{Z}$ as groups, G' is a d-dimensional vector space over $\mathbb{F}_p$; however, it might be the case that G can be viewed as being defined over a field extension $\mathbb{F}_q$ of $\mathbb{F}_p$. Viewing G over the largest possible $\mathbb{F}_q$, G' is a g-dimensional vector space over $\mathbb{F}_q$. Then g is the **genus** of G. For a few examples, consider the following groups which (respectively) are genus 1 and genus 2.

Example 2.5.2.1.

$$H_m(\mathbb{F}_q) = \left\{ \begin{bmatrix} 1 & e & z \\ 0 & I_m & f^T \\ 0 & 0 & 1 \end{bmatrix} : e, f \in (\mathbb{F}_q)^m, z \in \mathbb{F}_q \right\}$$

Example 2.5.2.2.

$$H_m^\flat(\mathbb{F}_q) = \left\{ \left[\begin{array}{c|ccc|c} & e_1 & \cdots & e_m & z_1 \\ I_2 & 0 & e_1 & \cdots & e_m \\ \hline & & & & f_0 \\ & & I_{m+1} & & \vdots \\ & & & & f_m \\ \hline & & & & 1 \end{array}\right] : e_i, f_i, z_i \in \mathbb{F}_q \right\}$$

It is not difficult to verify

$$Z(H_m(\mathbb{F}_q)) = \left\{ \begin{bmatrix} 1 & 0 & z \\ 0 & I_m & 0 \\ 0 & 0 & 1 \end{bmatrix} : z \in \mathbb{F}_q \right\} \text{ and}$$

$$Z(H_m^\flat(\mathbb{F}_q)) = \left\{ \left[\begin{array}{cc|c} I_2 & 0 & \begin{matrix} z_1 \\ z_2 \end{matrix} \\ \hline 0 & I_{m+1} & 0 \\ \hline 0 & 0 & 1 \end{array}\right] : z_i \in \mathbb{F}_q \right\},$$

so $H_m(\mathbb{F}_q)$ has genus 1 and $H_m^\flat(\mathbb{F}_q)$ has genus at most 2. To show $H_m^\flat(\mathbb{F}_q)$ has genus 2, which it does, we need to show $H_m^\flat(\mathbb{F}_q)$ cannot be defined over a larger field. Exploring this idea more, consider the following group.

Example 2.5.2.3.

$$H = \left\{ \left[\begin{array}{c|cccc|c} & a_1 & a_2 & 0 & 0 & z_1 \\ & a_2 & 2a_2 & 0 & 0 & z_2 \\ I_4 & 0 & 0 & a_1 & a_2 & z_3 \\ & 0 & 0 & a_2 & 2a_2 & z_4 \\ \hline & & & & & b_1 \\ & & & I_4 & & b_2 \\ & & & & & b_3 \\ & & & & & b_4 \\ \hline & & & & & 1 \end{array}\right] : a_i, b_i, z_i \in \mathbb{F}_3 \right\}.$$

At first glance, H appears to be genus 4 as $Z(H) = H'$ is 4-dimensional over $\mathbb{F}_3$. However, the map $\varphi : H \to H_1^\flat(\mathbb{F}_3[x]/(x^2+x+2))$ defined by

$$\left[\begin{array}{c|cccc|c} & a_1 & a_2 & 0 & 0 & z_1 \\ & a_2 & 2a_2 & 0 & 0 & z_2 \\ I_4 & 0 & 0 & a_1 & a_2 & z_3 \\ & 0 & 0 & a_2 & 2a_2 & z_4 \\ \hline & & & & & b_1 \\ & & & I_4 & & b_2 \\ & & & & & b_3 \\ & & & & & b_4 \\ \hline & & & & & 1 \end{array}\right] \mapsto \left[\begin{array}{c|cc|c} & a_1x+a_2 & 0 & z_1x+z_2 \\ I_2 & 0 & a_1x+a_2 & z_3x+z_4 \\ \hline & & & b_1x+b_2 \\ & I_2 & & b_3x+b_4 \\ \hline & & & 1 \end{array}\right]$$

is an isomorphism, so $H \cong H_1^\flat(\mathbb{F}_9)$ and H is actually a genus 2 group.

Generalizing our notion of genus slightly, consider any p-group of exponent p with $G' \leq Z(G)$. Then G splits as a direct product $A \times G_1 \times \cdots \times G_m$ where $m \geq 1$, A is abelian, and $G_i' = Z(G_i)$. Let g_i be the genus of G_i, the genus of G is defined to be $\max\{g_1, \ldots, g_m\}$. Looking ahead, Chapter 4 focuses on groups which are quotients of genus 2 groups.

2.5.3 Group representation styles

As runtime analysis depends on the input size, we next discuss possible options for representing groups in computers. In mathematics, it is very common to represent a group as a list of generators and relations. However, these descriptions are ill-suited to computers as the following problems (and many more) are all undecidable (see [55, Corollary 3.4]). Given a group $G\langle g_1, \ldots, g_n \mid R_1 = 1, \ldots, R_m = 1\rangle$, determine:

- Is G a finite group?
- Is G the trivial group?
- Does a word w written in terms of $g_1, \ldots, g_n$ represent the identity element of G?

On the opposite extreme, another common way to represent a finite group is as its Cayley table. While this does not have the problem of simple problems being undecidable, the group G's Cayley table takes $O(|G|^2)$ space to store, which is cumbersome when $|G|$ is large. In particular, attempting to store the Cayley tables of the groups of study in Chapter 4 in RAM will fail, in all but the smallest examples, due to a lack of memory addresses. For example, in theory, a 64-bit processor can address 2^{64} bytes of memory (18.4 exabytes); there are only 7 flat genus 2 groups of size $\sqrt{2^{64}}$ or smaller having quotients where the quotient group has genus greater than 2. That said, representing groups as Cayley tables is a useful theoretical problem as the size of the input is polynomial in $|G|$, so we define the decision problem GpI-CayleyTable as: Given groups G, H as their Cayley tables, is $G \cong H$? In practice, isomorphism algorithms which are given a full Cayley table perform limited Cayley table lookup operations, and furthermore, having a full Cayley table seems to provide limited runtime benefit compared to having only a nice set of generating elements, as we now explain.

To balance these two extremes, we will insist on representing groups by a small set of generators such that common procedures can be completed efficiently. For the size of the generating set, let $|G| = n = p_1^{\alpha_1} \cdots p_m^{\alpha_m}$ and define $\mu(n) = \max\{\alpha_1, \ldots, \alpha_m\}$. Then G can generated by $\mu(n) + 1$ elements [15, Corollary 16.7]. In the worst case, $n = p^\alpha$, so $\mu(n) = \alpha$. Thus G always has

a generating set of size $O(\log(|G|))$. Regarding the required efficient algorithms, we will work with any presentation of finite groups such that the group can be stored in $O(\text{poly}(\log(|G|)))$ space and there are $O(\text{poly}(\log(|G|)))$ algorithms to

- Find $|G|$.
- Given $g \in G$ and $\{g_1, \ldots, g_m\} \subseteq G$, either write g as a word over $\{g_1, \ldots, g_m\}$ or prove $g \notin \langle g_1, \ldots, g_m \rangle$.
- Find generators for $Z(G)$ and G'.
- Decide if G is nilpotent of class 2.
- If G is nilpotent of class 2, find a system of bilinear forms for $\text{Bi}(G)$ (see Definition 4.2.1.1).

In particular, all these requirements are satisfied by permutation groups [67], permutation group quotients [37, 67, 20], and matrices over finite fields [52]. Furthermore, there are also presentations satisfying these requirements specific to certain classes of groups such as black box polycyclic representations of solvable groups [32].[8] Chapter 4 is focused on the group isomorphism problem where the groups are given as generating sets. We define GpI to be the decision problem: Given finite groups G, H as generating sets, is $G \cong H$? For any given result, it is important to specify how the generating set is given. If the details of the generating set are either not important or clear from context, we will just write GpI.

As an addendum to our discussion of the complexity of GI and GpI in §1.2.2, consider Figure 2.8 where an arrow $A \to B$ means A reduces to B, i.e., $A \leq_m^P B$. In Chapter 4, we consider nilpotent groups of class 2. For class 2 groups, if the generating set is given by a set of matrices, GpI is known to be in $\mathsf{NP} \cap \mathsf{coAM}$. Removing the restriction of class 2 groups, the general case of generators given by permutations is known to be in NP [53]. When the generators are given by matrices instead, GpI is only known to be in NP if you assume an order oracle. Even harder are black box representations; deciding isomorphism is only known to be in Σ_2^{P} [11], and black

[8] Note that nilpotent groups are solvable.

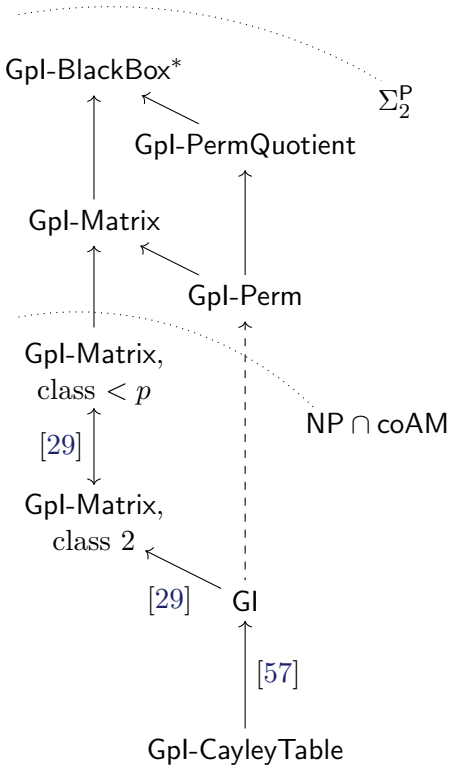

Figure 2.8: The current complexity landscape for GpI. The reductions represented by unlabeled solid arrows are obvious special cases. The reduction represented by the dashed arrow seems to be well-known by the experts but might not be present in the literature. (*) Black box representations are officially only in the promise hierarchy; to get around this fact, one can consider groups of black box type instead [21]. Isomorphism of groups of black box type is in NP rather than just Σ_2^{P}.

box representations are in the promise hierarchy, so reducing to black box representations from problems not in the promise hierarchy is unusual. Instead, Dietrich and Wilson propose the use of groups of "black box type" [21]. Thankfully, the models discussed above all have representations as groups of black box type, and GpI for groups of black box type is in NP.

Chapter 3

Conjugacy and recognition of shifts of finite type

This chapter is joint work with Rafael Frongillo which also resulted in [66].

3.1 Overview

One-dimensional subshifts of finite type (SFTs) are of fundamental importance in the study of symbolic dynamical systems. Despite their central role in symbolic dynamics, however, several basic questions about SFTs remain open, particularly with regard to computation. Most prominent is the conjugacy problem: whether there is an algorithm to decide if two given SFTs are conjugate. In this work, we study restricted versions of the conjugacy problem, with an eye toward applications (algorithms to simplify representations of SFTs) as well as developing insights toward the full conjugacy problem. In particular, we address the computational complexity of deciding or verifying conjugacy when given a bound on the block size of the corresponding sliding block code. We focus on the case of vertex shifts; see §3.5 for an extension to edge shifts.

First consider the question of verification: given two vertex shifts and a proposed sliding block code, what is the computational complexity of verifying that the code induces a conjugacy? We give a polynomial-time algorithm, for both the irreducible and reducible cases (§3.2, Theorem A). An algorithm deciding the irreducible case was apparently known to the experts [19], but to our knowledge had not appeared in the literature previously. Second, the question of deciding k-block conjugacy: given two vertex shifts, what is the complexity of deciding if there exists a sliding block code, with block length at most k, that induces a conjugacy? By the first result on efficient

verification, this problem is in NP; we show it to be GI-hard for all k (§3.3, Theorem B). Third, the question of reduction: given a vertex shift and integer ℓ, what is the complexity of deciding whether there exists a k-block conjugacy which reduces the number of vertices by ℓ? Extending a construction from previous work [22], we show that this problem, for $k = 1$, is NP-complete (§3.4, Theorem C). Fourth, the question of recognition: given a sofic shift, what is the complexity of deciding if the sofic shift is a SFT? The same key idea for the verification algorithm can also be used to give a polynomial-time algorithm for this problem (§3.6, Theorem D).

It is interesting to contrast our results with those of previous work [22], on the special case of $k = 1$ with the restriction that the block code be a sequence of amalgamations. (Recall that any conjugacy can be expressed as a sequences of splittings followed by amalgamations; see Theorem 2.3.5.) This previous work shows that the analogous version of our third problem, of reducing the number of vertices using only amalgamations, is NP-complete, but it does not address the verification problem; intuitively it seems plausible that verification would also be NP-hard. Returning to our setting, note that general 1-block codes need not be sequences of amalgamations (Figure 2.6). Thus, while it is unsurprising that the reduction problem remains NP-hard in our setting, it is perhaps surprising given that verification can be done in polynomial time, as a priori the number of splittings required could be super-polynomial.

We conclude this section with a common way a sliding block code can fail to be injective. Given a k-block code $\Phi_\infty : X_G \to X_H$, if there exist distinct words w_2, w_2' such that $\Phi(w_1w_2w_3) = \Phi(w_1w_2'w_3)$ with $|w_1| = |w_3| = k$, we say Φ_∞ **collapses a diamond**. As we now state, if a sliding block code is injective, it cannot collapse a diamond. (As we discuss in §3.2.2, if Φ_c is injective, collapsing a diamond is actually the only way Φ_∞ can fail to be injective.) We prove the result for completeness; see, e.g., [51, Theorem 8.1.16] for a similar result in the irreducible case.

Lemma 3.1.1. *Let $\Phi_\infty : X_G \to X_H$ be a k-block code. If Φ collapses a diamond, then Φ_∞ is not injective.*

Proof. Suppose Φ collapses a diamond. That is, $\Phi(w_1w_2w_3) = \Phi(w_1w_2'w_3)$ for some words w_1, w_3

of length k and distinct words w_2, w_2' in G. Consider any infinite word w_0 which can precede w_1 and any infinite word w_4 which can follow w_3. Then $\Phi_\infty(w_0 w_1 w_2 w_3 w_4) = \Phi_\infty(w_0 w_1 w_2' w_3 w_4)$, so Φ_∞ is not injective. □

3.2 Verification: testing a k-block map for conjugacy

Given a pair of directed graphs G, H, and a proposed k-block map Φ, we wish to verify whether or not Φ induces a conjugacy between the vertex shifts X_G, X_H. We will focus in this section on the case $k = 1$, as the case $k > 1$ follows immediately by recoding to the kth higher block shift. When G and H are irreducible (strongly connected), this problem boils down to checking that the two graphs have the same number of cycles of each length up to some bound depending on G, and furthermore that Φ induces an injection on these cycles. While cycle counting can be done efficiently using powers of the adjacency matrices, the challenge remains of checking injectivity efficiently.

The reducible case, when G and H are not strongly connected, is much more complex. We give counterexamples to several statements which would have led to a straightforward algorithm wherein one subdivides the graphs into their irreducible (strongly connected) components and uses the algorithm for the irreducible case on each, together with some other global checks. Instead, we give a more direct reduction to the irreducible case: we efficiently augment the graphs and block map with new vertices and edges, until the resulting graphs are irreducible, in such a way as to preserve conjugacy (or lack thereof).

3.2.1 Irreducible case

As described above, we will focus first on 1-block codes. When G, H are irreducible, the following straightforward topological result allows us to restrict attention to the map induced on cycles between the graphs.

Remark 3.2.1.1. The results in this section actually only need the periodic points of the shift spaces to be dense; that is, every result in this section applies to all nonwandering shifts of finite type.

However, recalling our assumption in §2.3 that every graph is connected, the nonwandering shifts of finite type which can be represented by a connected graph are exactly the irreducible shifts of finite type.

Proposition 3.2.1.2. *Suppose X, Y are compact metric spaces, $\psi : X \to Y$ is continuous, and $D \subseteq Y$ is a dense subset of Y. If ψ surjects onto D, then ψ surjects onto all of Y.*

Proof. Suppose X, Y are compact metric spaces, $\psi : X \to Y$ is continuous, $D \subseteq Y$ is a dense subset of Y, and $D \subseteq \psi(X) \subseteq Y$. Let $p \in Y$. Since D is dense, there is a sequence $\{p_n\}$ in D which converges to p. Since ψ surjects onto D, every p_n has a preimage in X. Pick $\gamma : D \to X$ such that $\psi \circ \gamma = \mathrm{id}_D$. Then $\{\gamma(p_n)\}$ is a sequence in X. Since X is compact, there is a subsequence $\{\gamma(p_{n_k})\}$ which converges to some $q \in X$. As limits commute with continuous functions, we have

$$\psi(q) = \psi(\lim_{n_k \to \infty} \gamma(p_{n_k})) = \lim_{n_k \to \infty} \psi(\gamma(p_{n_k})) = \lim_{n_k \to \infty} p_{n_k} = p.$$

Thus $\psi(q) = p$, so ψ is surjective on all of Y. □

We will apply Proposition 3.2.1.2 with D being the set of periodic points of X_H, which are in bijection with cycles of H. The following result, that Φ induces a 1-block conjugacy if and only if it induces a bijection on cycles, appears to be known (see [51, Exercise 9.1.8] for a related result); we give the proof for completeness.

Theorem 3.2.1.3. *Irreducible vertex shifts X_G, X_H are conjugate via a 1-block code if and only if there is a vertex map $\Phi : V_G \to V_H$ such that the induced map Φ_{c} is a bijection.*

Proof. If Φ_∞ is a conjugacy, then it is a bijection on periodic points; we conclude Φ_{c} is a bijection. For the converse, suppose Φ_∞ is not a conjugacy. We proceed in cases.

(Case 1) If Φ_∞ is not injective, there exist distinct points $p, q \in X_G$ such that $\Phi_\infty(p) = \Phi_\infty(q)$. (Case 1a) Suppose first that p, q disagree at $|V_G|^2 + 1$ consecutive indices, meaning the words $p_{[a,b]}, q_{[a,b]}$ disagree at every index for some $a, b \in \mathbb{Z}$ with $b - a = |V_G|^2 + 1$. Consider all possible pairs of states in G; there are $|V_G|^2$ such pairs. Thus there exist distinct indices $c, d \in \{a, a+1 \ldots, b\}$ such that $(p_c, q_c) = (p_d, q_d)$. But then $\Phi_{\mathsf{c}}(p_{[c,d-1]}) = \Phi_{\mathsf{c}}(q_{[c,d-1]})$.

(Case 1b) Suppose instead that p, q do not disagree at $|V_G|^2 + 1$ consecutive indices: there exist indices a, b with $a < b-1$ such that p, q agree at indices a and b, but p, q disagree at every index between a and b. Let w be any word connecting $p_b = q_b$ to $p_a = q_a$. Then $\Phi_{\mathsf{c}}(p_{[a,b]}w) = \Phi_{\mathsf{c}}(q_{[a,b]}w)$.

(Case 2) If Φ_∞ is not surjective, suppose for a contradiction that Φ_{c} is bijective. Then every periodic point in X_H is mapped to by Φ_∞. Since the periodic points are a dense subset of the compact metric space X_H, Proposition 3.2.1.2 contradicts the fact that Φ_∞ is not surjective. □

To verify that the cycle map Φ_{c} is bijective, we will test for injectivity explicitly, and rely on counting arguments to check surjectivity. For injectivity, it turns out that checking cycles up to length $|V_G|^2$ suffices.

Proposition 3.2.1.4. *Suppose $\Phi_\infty : X_G \to X_H$ is a 1-block code between irreducible vertex shifts. If Φ_{c} is injective on $\bigcup_{n=1}^{|V_G|^2} C_n(G)$, then Φ_{c} is injective.*

Proof. Let c, d be distinct cycles of size $|c| = |d| = k > |V_G|^2$, and suppose Φ_{c} is injective on all cycles of size less than k. There are $|V_G|^2$ possible pairs of states in G. Thus there exist distinct indices a, b such that $(c_a, d_a) = (c_b, d_b)$. That is, $c_{[a,b-1]}, d_{[a,b-1]}$ are cycles of the same length and $c_{[b,a-1]}, d_{[b,a-1]}$ are cycles of the same length. Since c, d were distinct, we can assume without loss of generality that $c_{[a,b-1]}, d_{[a,b-1]}$ are distinct. By our assumption that k was minimal, $\Phi_{\mathsf{c}}(c_{[a,b-1]}) \neq \Phi_{\mathsf{c}}(d_{[a,b-1]})$. Thus $\Phi_{\mathsf{c}}(c) \neq \Phi_{\mathsf{c}}(d)$. □

Proposition 3.2.1.4 suggests the naïve algorithm of checking all cycles up to length $|V_G|^2$ to verify injectivity of Φ_{c}. This algorithm is remarkably inefficient, however; letting $n = |V_G|$, there can be $\Omega(n^{n^2})$ cycles of length up to n^2, as is the case for the complete graph. Fortunately, these checks can be performed much more efficiently, by rephrasing them as a search problem in a graph built from pairs of vertices in G. This procedure is outlined in Algorithm 1.

Theorem 3.2.1.5. *Let X_G be a vertex shift and $A = \{1, 2, \ldots, m\}$. Then any given map $\Phi : V_G \to A$ induces a map $\Phi_{\mathsf{c}} : \bigcup_n C_n(G) \to \bigcup_n A^n$. Given Φ, deciding if Φ_{c} is injective can be determined in $O(|V_G|^4)$ time.*

Proof. First we build the directed meta-graph $M = (V_M, E_M)$ where $V_M = \{(v_1, v_2) : v_1, v_2 \in V_G\}$ and $E_M = \{((v_1, v_2), (u_1, u_2)) : \Phi(v_1) = \Phi(v_2), \Phi(u_2) = \Phi(v_2), (v_1, u_1) \in E_G, \text{ and } (v_2, u_2) \in E_G\}$. That is, M is a graph on pairs of vertices from G, with an edge connecting pairs P_1, P_2 if and only if (i) there is a pair of (possibly non-distinct) edges in G connecting the two vertices in P_1 to the vertices in P_2, and (ii) the induced map on words of length two (i.e., edges) maps the two edges together. M can be constructed in $O(|V_G|^4)$ time.

Given M, the map Φ_{c} is injective if and only if there is no cycle in M which passes through a vertex $(v_1, v_2) \in V_M$ with $v_1 \neq v_2$. Furthermore, such a cycle in M exists if and only if M has a strongly connected component containing an edge and a vertex (v_1, v_2) with $v_1 \neq v_2$. Tarjan's strongly connected components algorithm [71] now applies, in $O(|V_M| + |E_M|) = O(|V_G|^4)$ time. □

Putting the above results together with the higher-block codes gives the desired algorithm to verify k-block conjugacies; the full conjugacy algorithm for $k = 1$ is outlined in Algorithm 2.

Corollary 3.2.1.6. *Given a k-block code $\Phi_\infty : X_G^k \to X_H$ between irreducible vertex shifts, deciding if Φ_∞ is a conjugacy is in* P. *In particular, it can be determined in $O(|V_G|^{4k})$ time.*

Proof. Given G, H, we first pass to the kth higher block shift $X_{G^{[k]}}$ of X_G, recalling that $\Phi_\infty^{[k]}$ is a 1-block code and Φ_∞ is a conjugacy if and only if $\Phi_\infty^{[k]}$ is a conjugacy [51, Proposition 1.5.12]. We can construct $\Phi_\infty^{[k]} : X_{G^{[k]}} \to X_H$ in time $O(|V_{G^{[k]}}| + |E_{G^{[k]}}|) = O(|V_{G^{[k]}}|^2)$. Noting that $|V_{G^{[k]}}| \leq |V_G|^k$, it thus suffices to show the case $k = 1$.

By Theorem 3.2.1.3, Φ_∞ is a conjugacy if and only if Φ_{c} is a bijection. Since $k = 1$, Theorem 3.2.1.5 shows the injectivity of Φ_{c} can be determined in $O(|V_G|^4)$ time. To show Φ_{c} is surjective, it suffices to check that $|C_i(G)| = |C_i(H)|$ for all $i \in \mathbb{N}$. Letting $A(G), A(H)$ be the adjacency matrices of G, H, we note $|C_i(G)| = \operatorname{tr}(A(G)^i)$, so our desired check is equivalent to checking $\operatorname{tr}(A(G)^i) = \operatorname{tr}(A(H)^i)$ for all $i \in \mathbb{N}$ [51, Proposition 2.2.12]. In fact, it suffices to check up to $i = |V_G|$ [33, 46]. Calculating $\operatorname{tr}(A(G)^i), \operatorname{tr}(A(H)^i)$ for all $i \in \{1, \ldots, |V_G|\}$ can be done by repeated multiplication in $O(|V_G|^{1+\omega}) = O(|V_G|^4)$ time, where ω is the exponent of matrix

multiplication.[1] □

3.2.2 Reducible case

Several useful statements about conjugacy between irreducible vertex shifts fail to hold in the reducible case. First, given a sliding block code $\Phi_\infty : X_G \to X_H$ between irreducible vertex shifts, it is known that if Φ_∞ is injective and G, H have the same topological entropy, then Φ_∞ is a conjugacy [51, Corollary 8.1.20]. (The topological entropy of a shift X is defined as $h(X) = \lim_{n\to\infty} \frac{1}{n} \log_2 |\mathcal{B}_n(X)|$.) If the shifts are reducible, however, Φ_∞ can satisfy these conditions but fail to be surjective (Figure 3.1a). Second, we have from Theorem 3.2.1.3 that if Φ_∞ is a 1-block code between irreducible vertex shifts, then the bijectivity of Φ_c implies the bijectivity of Φ_∞. In the reducible case, Φ_c can be bijective while Φ_∞ fails to be injective (Figure 3.1b) or surjective (Figure 3.1a).

As an even stronger test, one might guess for reducible vertex shifts that if $\Phi_\infty : X_G \to X_H$ is surjective and the induced maps between irreducible subgraphs are all conjugacies, then Φ_∞ is a conjugacy. If true, this statement would suggest applying the algorithm in Corollary 3.2.1.6 to each irreducible subgraph, at which point one would only need to test surjectivity. Yet this statement is also false; bijectivity of Φ_c implies neither the injectivity nor the surjectivity of Φ_∞ (Figure 3.1). By extending the argument of Theorem 3.2.1.3, one can correct the statement by adding a check for diamonds: if Φ_∞ is surjective, the induced maps between irreducible subgraphs are all conjugacies, and Φ does not collapse a diamond, then Φ_∞ is a conjugacy. Unfortunately, while this revised statement does break the problem of verifying a proposed 1-block conjugacy into more manageable pieces, how to turn it into a decision procedure, let alone an efficient algorithm, is far from clear.

To verify a potential conjugacy between vertex shifts efficiently, we will instead apply a more direct reduction to the irreducible case. Given a 1-block code $\Phi_\infty : X_G \to X_H$ between reducible vertex shifts, we will extend G and H to irreducible graphs while preserving the conjugacy or

[1] Multiplication of $n \times n$ matrices can be performed in $O(n^\omega)$ time for some exponent ω. The smallest possible number ω is known as the exponent of matrix multiplication, and its exact value is unknown. While the algorithm typically performed by hand runs in $O(n^3)$ time, there are more efficient algorithms. The current known bounds are $2 \leq \omega < 2.3728639$ [44].

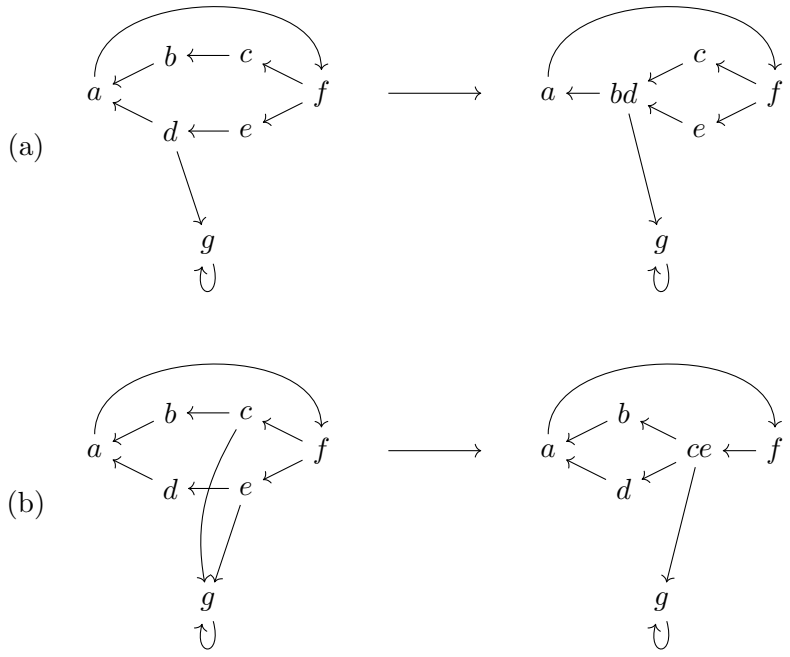

Figure 3.1: Counterexamples showing various statements which hold in the irreducible case fail in the reducible case. Note that all four shifts have the same topological entropy, $h(X) = \frac{1}{4}$. (a) A 1-block code between two reducible shifts which restricts to conjugacies between the irreducible components (and hence Φ_{c} is a bijection) but is not surjective. (b) A 1-block code between two reducible shifts which restricts to conjugacies between the irreducible components but is not injective.

non-conjugacy of Φ_∞. The key operation for this extension is the following procedure, which adds a new sink vertex to a sink component in such a way as to preserve conjugacy/non-conjugacy. We will then apply this procedure to every sink component, and in reverse to every source component, until we have enough structure to connect the new sink vertices back to the new source vertices through a new vertex $*$, rendering both graphs irreducible.

Let T be a sink component of H and $T' = \Phi^{-1}(T)$ be the subgraph of G which maps to T under Φ_∞. The procedure is as follows:

(1) Pick an arbitrary vertex v in T.

(2) Pick an arbitrary cycle c in T ending at v of length $|c| \leq |T|$.

(3) Add the vertex t along with the edges $(t,t),(v,t)$ to H. Call this new graph $\hat{H}$.

(4) Select the vertices $v' \in \Phi^{-1}(v)$ which are followed by an infinite word w' such that $\Phi(v'w') = vc^\infty$. Call this set of vertices V'.

(5) Add the vertex t' and the edges $\{(v', t') : v' \in V' \cup \{t'\}\}$ to G. Call this new graph $\hat{G}$.

(6) Define $\hat{\Phi}_\infty : X_{\hat{G}} \to X_{\hat{H}}$ by $\hat{\Phi}(u) = \begin{cases} \Phi(u), & \text{if } u \neq t' \\ t, & \text{if } u = t' \end{cases}$.

Proposition 3.2.2.1. *Let $\Phi_\infty : X_G \to X_H$ be a 1-block code between reducible vertex shifts. Then $\hat{\Phi}_\infty : X_{\hat{G}} \to X_{\hat{H}}$ as described above is a conjugacy if and only if Φ_∞ is a conjugacy.*

Proof. Since X_G is a subshift of $X_{\hat{G}}$ (and similarly for H) and $\hat{\Phi}_\infty$ extends Φ_∞, it immediately follows that Φ_∞ is a conjugacy whenever $\hat{\Phi}_\infty$ is. For the converse, suppose $\hat{\Phi}_\infty$ is not a conjugacy.

If $\hat{\Phi}_\infty$ is not injective, there are distinct points $p_1, p_2 \in X_{\hat{G}}$ such that $\hat{\Phi}_\infty(p_1) = \hat{\Phi}_\infty(p_2) = q$. If $q \in X_H$, then $p_1, p_2 \in X_G$ by the definition of $\hat{\Phi}_\infty$, so Φ_∞ is not injective. Note that by the construction of $X_{\hat{H}}$, all points in $X_{\hat{H}} \setminus X_H$ have form wvt^∞, where v is the vertex chosen in (1), t is the vertex added in (3), and w is some infinite word; so if $q \notin X_H$, then $q = wvt^\infty$. By the definition of $\hat{\Phi}_\infty$, we have $p_1 = w_1 v_1 t'^\infty, p_2 = w_2 v_2 t'^\infty$. By the construction of $N^-(t')$, there exist infinite words w_1', w_2' such that $v_1 w_1', v_2 w_2'$ are words in G and $\Phi(w_1') = c^\infty = \Phi(w_2')$. Thus $\Phi_\infty(w_1 v_1 w_1') = \Phi_\infty(w_2 v_2 w_2')$, and Φ_∞ is not injective.

If $\hat{\Phi}_\infty$ is not surjective, then there exists $p \in X_{\hat{H}}$ which is not mapped to. If $p \in X_H$, then Φ is not surjective. Otherwise, $p \notin X_H$, so $p = wvt^\infty$. But again noting the construction of $N^-(t)$, the point wvc^∞ is not mapped to. □

We now construct the final graphs G^*, H^* using the above procedure as well as one additional step below. Let $T_1, \ldots, T_m$ be the sink components of H, and $S_1, \ldots, S_\ell$ the source components, with $T_i' = \Phi^{-1}(T_i)$ and $S_i' = \Phi^{-1}(S_i)$ the corresponding inverse image subgraphs of G. We apply the above procedure iteratively to every sink component, and every source component by reversing the edge direction in G, H, applying the procedure, and reversing edges back. Let $\hat{G}, \hat{H}$ denote the graphs after applying the procedure to the m sinks and ℓ sources. Note that, by construction,

each sink or source component in $\hat{H}$ consists only of a single state. Furthermore, the preimage of each of these states under the induced map $\hat{\Phi}$ in $\hat{G}$ also consists of a single state. Denote the source states in $\hat{H}$ as $\{s_1, \ldots, s_\ell\}$ and the sink states as $\{t_1, \ldots, t_m\}$. In $\hat{G}$, denote the preimages as $s'_i = \Phi^{-1}(s_i), t'_j = \Phi^{-1}(t_j)$. We extend $\hat{H}, \hat{G}$ to the irreducible graphs H^*, G^* as follows:

(1) Add a new vertex $*$ to both $\hat{H}$ and $\hat{G}$.

(2) In H^*, set $N^-(*) = \{t_1, \ldots, t_m\}$, $N^+(*) = \{s_1, \ldots, s_\ell\}$. In G^*, set $N^-(*) = \{t'_1, \ldots, t'_m\}$, $N^+(*) = \{s'_1, \ldots, s'_\ell\}$.

(3) Define $\Phi^*_\infty : X_{G^*} \to X_{H^*}$ by $\Phi^*(u) = \begin{cases} \hat{\Phi}(u), & \text{if } u \neq * \\ *, & \text{if } u = * \end{cases}$.

Proposition 3.2.2.2. *Let $\Phi_\infty : X_G \to X_H$ be a 1-block code between reducible vertex shifts. Then $\Phi^*_\infty : X_{G^*} \to X_{H^*}$ as described in the construction above is a conjugacy if and only if Φ_∞ is a conjugacy.*

Proof. By Proposition 3.2.2.1, $\hat{\Phi}_\infty : X_{\hat{G}} \to X_{\hat{H}}$ is a conjugacy if and only if Φ_∞ is a conjugacy, where, as above, the graphs $\hat{G}, \hat{H}$ immediately precede the addition of the vertex $*$. As in the proof of Proposition 3.2.2.1, $X_{\hat{G}}$ is a subshift of X_{G^*} (and similarly for $\hat{H}$) and Φ^*_∞ extends $\hat{\Phi}_\infty$, so $\hat{\Phi}_\infty$ is a conjugacy if Φ^*_∞ is. For the other direction, suppose Φ^*_∞ is not a conjugacy.

First suppose Φ^*_∞ is not injective. Since G^*, H^* are irreducible, we have distinct cycles c, d in G^* from Theorem 3.2.1.3 such that $\Phi^*_{\mathsf{c}}(c) = \Phi^*_{\mathsf{c}}(d)$. Without loss of generality, c, d pass through $*$. Since Φ^* is bijective on $\{s_1, \ldots, s_\ell, t_1, \ldots, t_m\}$, we have $c = *s_i w_1 t_j, d = *s_i w_2 t_j$. But then $\hat{\Phi}$ collapses the diamond $(s_i w_1 t_j, s_i w_2 t_j)$, so by Lemma 3.1.1, $\hat{\Phi}_\infty$ is not injective.

Now suppose Φ^*_∞ is not surjective. Again by Theorem 3.2.1.3, we know there is a cycle c which is not in the image of Φ^*_{c}. Without loss of generality, we can assume $c = *s_i w t_j$. By the construction of $N^-(*), N^+(*)$, we conclude that $s_i w t_j$ is not in the image of $\hat{\Phi}$. Thus $s_i^\infty w t_j^\infty$ is a point in $X_{\hat{H}}$ which is not in the image of $\hat{\Phi}_\infty$. □

We now have that given reducible vertex shifts X_G, X_H and a proposed 1-block conjugacy

between them, the shifts can be embedded into irreducible shifts such that the conjugacy or non-conjugacy is preserved. Next we show this embedding can be performed efficiently; the procedure described in the proof is outlined in Algorithm 5.

Theorem 3.2.2.3. *Given reducible vertex shifts X_G, X_H and a 1-block code as $\Phi : V_G \to V_H$, the graphs G^* and H^* can be constructed in $O(|V_G|^3)$ time.*

Proof. Let T be an arbitrary sink component in H and T' be the subgraph $\Phi^{-1}(T)$ of G. We will show the corresponding sink vertices t, t' can be added in $O(|V_{T'}|^3)$ time. Iterating over all sink components $T \in \mathcal{T}$ and source components $S \in \mathcal{S}$ will give an overall complexity of $O(\sum_{T' \in \mathcal{T}'} |V_{T'}|^3 + \sum_{S' \in \mathcal{S}'} |V_{S'}|^3) = O(|V_G|^3)$ time. (Adding the vertex $*$ takes linear time.)

Let v be an arbitrary vertex of T, and let c be the shortest cycle in T through v, which can be computed using breadth-first search in $O(|V_T| + |E_T|) = O(|V_H|^2) = O(|V_G|^2)$ time. Note that $|c| \leq |T|$, so we have completed steps 1 and 2. Step 3 is constant time. The only nontrivial step that remains is step 4, the computation of the set $V' \subseteq V_{T'}$, from which steps 5 and 6 follow trivially in linear time.

Let $C = (V_C, E_C)$ be the subgraph of T corresponding to c, and let $C' = (V_{C'}, E_{C'})$ be the subgraph of T' which maps onto C as follows: $V_{C'} = \Phi^{-1}(V_C)$, and $E_{C'} = \{(u', v') \in E_{T'} : (\Phi(u'), \Phi(v')) \in E_C\}$. The subgraph C' can be constructed in $O(|V_{T'}|^2)$ time. Note that infinite walks in C' starting from any $v' \in \Phi^{-1}(v)$ are precisely the walks in T' that map onto c^∞, and moreover, there is an infinite walk in C' starting from v' if and only if there is a path in C' from v' to a cycle in C'. We therefore define $V' \subseteq \Phi^{-1}(v) \subseteq V_{C'}$ to be the set of nodes v' such that there is a path in C' from v' to a cycle in C'. To compute V', we can simply run breadth-first search from each vertex in $\Phi^{-1}(v)$, in $O(|\Phi^{-1}(v)| \cdot (|V_{C'}| + |E_{C'}|)) = O(|V_{T'}|^3)$ time. □

We have now seen an efficient procedure to embed a pair of reducible graphs into a pair of irreducible graphs, such that the original pair admits a 1-block conjugacy if and only if the embedded pair does. Moreover, the embedded irreducible graphs have at most twice the number of vertices as the original graphs. With this procedure in hand, we can extend our verification

algorithm to the reducible case.

Corollary 3.2.2.4 (Theorem A)**.** *Given vertex shifts X_G, X_H and a k-block code Φ_∞ as $\Phi : V_G^k \to V_H$, deciding if Φ_∞ is a conjugacy can be determined in $O(|V_G|^{4k})$ time.*

Proof. If G, H are irreducible, Corollary 3.2.1.6 applies immediately. For the reducible case, as in Corollary 3.2.1.6, by passing to the kth higher block shift it suffices to show the case $k = 1$. By Theorem 3.2.2.3, we can embed G, H into the irreducible shifts G^*, H^* in $O(|V_G|^3)$ time. By Corollary 3.2.1.6, we can verify if Φ_∞^* (and hence Φ_∞) is a conjugacy in $O(|V_{G^*}|^4)$ time. Furthermore, $|V_{G^*}| < 2|V_G|$, so this verification runs in $O(|V_G|^4)$ time. □

3.3 Deciding k-block conjugacy

We now turn to the question of deciding k-block conjugacy. Specifically, we wish to understand the complexity of the problem k-BC, which is to decide given directed graphs G, H whether the vertex shifts X_G, X_H are conjugate via a k-block code $\Phi_\infty : X_G \to X_H$. Note that the description size of Φ is polynomial in $|V_G|$ and $|V_H|$, and thus from Corollary 3.2.2.4 we know that a potential k-block conjugacy can be verified in polynomial time; hence, k-BC is in NP. We will show that k-BC is GI-hard for all k, where GI is the class of problems with a polynomial-time Turing reduction to the Graph Isomorphism problem [42]. (A graph isomorphism is bijection between the vertices of two graphs which preserves the edges/non-edge relation; the Graph Isomorphism problem is to decide if two given undirected graphs are isomorphic. See §2.1.)

Definition 3.3.1. Given directed graphs G, H, the **k-Block Conjugacy Problem**, denoted k-BC, is to decide if there is a k-block conjugacy $\Phi_\infty : X_G \to X_H$ between the vertex shifts X_G and X_H.

To begin, we give the straightforward result that the case $k = 1$ is GI-hard, essentially because 1-block conjugacies between vertex shifts for equal sized graphs must be isomorphisms.

Theorem 3.3.2. *The 1-Block Conjugacy Problem,* 1-BC*, is* GI*-hard.*

Proof. Given strongly connected graphs directed G, H with $|V_G| = |V_H|$, we show that the shifts X_G, X_H are conjugate via 1-block code if and only if the graphs are isomorphic (cf. [51, Exercise 2.2.14]). The result then follows as graph isomorphism between strongly connected directed graphs is GI-hard, by the usual reduction from the undirected case (replace each edge with two directed edges).

First suppose $\Psi : V_G \to V_H$ is a graph isomorphism. As $\Psi(v_1v_2)$ is a legal word in X_H for all words of length 2, by definition of a graph isomorphism, we have that $\Psi_\infty : X_G \to X_H$ is a valid 1-block code. Letting $\Phi = \Psi^{-1} : V_H \to V_G$, we have $\Psi_\infty(\Phi_\infty((x_i)_{i\in\mathbb{Z}})) = (\Psi(\Phi(x_i)))_{i\in\mathbb{Z}} = (x_i)_{i\in\mathbb{Z}}$ for all $x \in X_H$, and $\Phi_\infty(\Psi_\infty((x_i)_{i\in\mathbb{Z}})) = (\Phi(\Psi(x_i)))_{i\in\mathbb{Z}} = (x_i)_{i\in\mathbb{Z}}$ for all $x \in X_G$. Thus, Φ_∞ is the 2-sided inverse of Ψ_∞, and Ψ_∞ is a 1-block conjugacy.

For the other direction, suppose $\Phi_\infty : X_G \to X_H$ is a 1-block conjugacy. Then $\{\Phi(v) : v \in V_G\}$ must be exactly the set of words of length 1 in X_H, i.e., the vertices of H. Since $|V_G| = |V_H|$, $\Phi : V_G \to V_H$ is a bijection. Also, for any edge $(v_1, v_2) \in E_G$, we have $\Phi(v_1v_2) = \Phi(v_1)\Phi(v_2)$, so $(\Phi(v_1), \Phi(v_2)) \in E_H$ as Φ_∞ is a well-defined sliding block code. Even more, consider any pair v_3, v_4 of vertices in V_G such that $(v_3, v_4) \notin E_G$. Noting that $\Phi(v_3)\Phi(v_4)$ has a unique preimage as Φ is a bijection and Φ_∞ is surjective, we have $(\Phi(v_3), \Phi(v_4)) \notin E_H$. Thus $\Phi : V_G \to V_H$ is a bijection on vertices which preserves the edge relationship; that is, Φ is a graph isomorphism. □

Next, we will show k-BC is GI-hard for all k, by reduction from the 1-block case. Specifically, given directed graphs G, H, we will construct graphs G', H' such that there exists a 1-block conjugacy $\Phi_\infty : X_G \to X_H$ if and only if there exists a k-block conjugacy $\Phi'_\infty : X_{G'} \to X_{H'}$. To form G', we replace every vertex $v \in V_G$ with a path $v_{\text{in}}v_1v_2\cdots v_{k-1}$ followed by the diamond with sides $v_{k-1}v_k^tv_{\text{out}}$ and $v_{k-1}v_k^bv_{\text{out}}$ (Figure 3.2a). To form H', we replace every vertex $u \in V_H$ with two parallel paths $u_{\text{in}}u_1^tu_2^t\cdots u_k^tu_{\text{out}}$ and $u_{\text{in}}u_1^bu_2^b\cdots u_k^bu_{\text{out}}$ (Figure 3.2b).

Lemma 3.3.3. *Given directed graphs G, H, let G', H' be constructed as above. If there exists a k-block conjugacy $\Phi'_\infty : X_{G'} \to X_{H'}$, then for all $v \in V_G$ there exists $u \in V_H$ such that $\Phi'(v_{in}v_1\cdots v_{k-1}) = u_{in}$.*

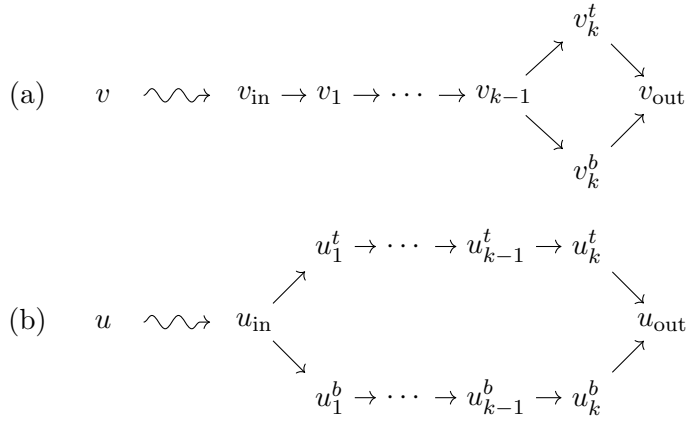

Figure 3.2: The vertex gadgets for (a) each vertex v in G, and (b) each vertex u in H.

Proof. Suppose for a contradiction that Φ'_∞ is a k-block code such that for some $v \in V_G$ we have $\Phi'(v_{\text{in}} v_1 \cdots v_{k-1}) \neq u_{\text{in}}$ for all $u \in V_H$. We break the argument into two cases.

First, suppose $\Phi'(v_{\text{in}} v_1 \cdots v_{k-1}) = u_i^t$. (The case u_i^b is identical.) Since the shift map commutes with sliding block codes, we must have $\Phi'(v_1 \cdots v_{k-1} v_k^t) = \Phi'(v_1 \cdots v_{k-1} v_k^b) = z$ where $z \in \{u_{i+1}^t, u_{\text{out}}\}$. Picking any edge $(v, \hat{v}) \in E_G$ and continuing to slide the block window, we must have $\Phi'(\hat{v}_{\text{in}} \hat{v}_1 \cdots \hat{v}_{k-1}) \in \{\hat{u}_i^t, \hat{u}_i^b\}$ for some $\hat{u} \in V_H$. Without loss of generality, assume $\Phi'(\hat{v}_{\text{in}} \hat{v}_1 \cdots \hat{v}_{k-1}) = \hat{u}_i^t$. Furthermore, since there is only one word in H' between u_i^t and $\hat{u}_i^t$ of proper length but two words in G' between $\hat{v}_{\text{in}}$ and v_{in}, we have

$$\Phi'(v_{\text{in}} \cdots v_k^t v_{\text{out}} \hat{v}_{\text{in}} \cdots \hat{v}_{k-1}) = u_i^t \cdots u_k^t u_{\text{out}} \hat{u}_{\text{in}} \hat{u}_1^t \cdots \hat{u}_i^t = \Phi'(v_{\text{in}} \cdots v_k^b v_{\text{out}} \hat{v}_{\text{in}} \cdots \hat{v}_{k-1}).$$

That is, Φ' collapses a diamond, so by Lemma 3.1.1, Φ' is not a conjugacy.

Second, suppose $\Phi'(v_{\text{in}} v_1 \cdots v_{k-1}) = u_{\text{out}}$. Pick any edge $(\hat{v}, v) \in E_G$. Then without loss of generality, $\Phi'(\hat{v}_{\text{out}} v_{\text{in}} \cdots v_{k-2}) = u_k^t$. Continuing to slide the block window, we have $\Phi'(\hat{v}_{\text{in}} \cdots \hat{v}_{k-1}) = \hat{u}_{\text{out}}$ for some $\hat{u} \in V_H$. Again, there are two words in G' between $\hat{v}_{\text{in}}$ and v_{in} but only one word in H' between $\hat{u}_{\text{out}}$ and u_{out} which passes through u_k^t. Thus, we have

$$\Phi'(\hat{v}_{\text{in}} \cdots \hat{v}_k^t \hat{v}_{\text{out}} v_{\text{in}} \cdots v_{k-1}) = \hat{u}_{\text{out}} u_{\text{in}} \cdots u_k^t u_{\text{out}} = \Phi'(\hat{v}_{\text{in}} \cdots \hat{v}_k^b \hat{v}_{\text{out}} v_{\text{in}} \cdots v_{k-1}),$$

so Φ' again collapses a diamond, and by Lemma 3.1.1, Φ' is not a conjugacy. □

We now show that graphs G, H admit a 1-block conjugacy if and only if the graphs G', H' constructed as above admit a k-block conjugacy. To do this, we first introduce a natural operation on shift spaces, which "stretches" each point by a factor N. Given alphabet $\mathcal{A}$, and any point $p = \cdots v.v' \cdots \in \mathcal{A}^{\mathbb{Z}}$, we write $p^{\langle N \rangle} = \cdots v \cdots v.v' \cdots v' \cdots$ to be the point p with each symbol repeated N times. Given a shift X over alphabet $\mathcal{A}$, we define the shift space $X^{\langle N \rangle} = \{\sigma^i(p^{\langle N \rangle}) : p \in X, i \in \mathbb{Z}\}$ where σ is the shift map. In particular, $X^{\langle N \rangle}$ contains all shifts of the Nth expansion of points in X. While in general $X^{\langle N \rangle}$ is not a vertex shift when $N > 1$, it is still structured enough that the following lemma is immediate.

Lemma 3.3.4. *Given shifts X, Y, there exists a 1-block conjugacy $\Phi_\infty : X \to Y$ if and only if there exists a 1-block conjugacy $\Phi_\infty^{\langle N \rangle} : X^{\langle N \rangle} \to Y^{\langle N \rangle}$, where $\Phi = \Phi^{\langle N \rangle}$ as block maps.*

To make use of this definition and lemma, we will project points in $X_{G'}, X_{H'}$ to $X_G^{\langle k+2 \rangle}, X_H^{\langle k+2 \rangle}$ by simply erasing the subscript and superscript information. Formally, we define the 1-block map $\Psi^G : V_{G'} \to V_G$ by $\Psi^G(u) = v$ for $u \in \{v_{\text{in}}, v_1, \ldots, v_{k-1}, v_k^t, v_k^b, v_{\text{out}}\}$, and let $\pi^G = \Psi_\infty^G : X_{G'} \to X_G^{\langle k+2 \rangle}$. We define Ψ^H, π^H similarly. Letting $S_p^G := \left(\pi^G\right)^{-1}(p) \subseteq X_{G'}$, we have that $\{S_p^G : p \in X_G^{\langle k+2 \rangle}\}$ is a partition of the points in $X_{G'}$. (Similarly for S_q^H and $X_{H'}$.)

Theorem 3.3.5. *Given graphs G, H, construct G', H' as above. Then there exists a 1-block conjugacy $\Phi_\infty : X_G \to X_H$ if and only if there exists a k-block conjugacy $\Phi' : X_{G'} \to X_{H'}$.*

Furthermore, the construction of G', H' can be done in polynomial time, so 1-BC $\leq_m$ k-BC.

Proof. The fact that G', H' can be constructed in polynomial time is obvious. Assuming the main theorem statement, the fact that 1-BC $\leq_m$ k-BC follows immediately. We now prove the main theorem statement.

($\Rightarrow$) Suppose there exists a 1-block conjugacy $\Phi_\infty : X_G \to X_H$. By Lemma 3.3.4, there is a 1-block conjugacy $\Phi_\infty^{\langle k+2 \rangle} : X_G^{\langle k+2 \rangle} \to X_H^{\langle k+2 \rangle}$. Define the k-block code $\Phi'_\infty : X_{G'} \to X_{H'}$ with no memory by

- $\Phi'(v_{\text{in}} \cdots) = \Phi(v)_{\text{in}}$

- $\Phi'(v_i \cdots v_k^t \cdots) = \Phi(v)_i^t$, $i \in \{1, \ldots, k\}$
- $\Phi'(v_i \cdots v_k^b \cdots) = \Phi(v)_i^b$, $i \in \{1, \ldots, k\}$
- $\Phi'(v_{\text{out}} \cdots) = \Phi(v)_{\text{out}}$

To show that Φ'_∞ is a bijection, we will show that for any $p \in X_G^{\langle k+2 \rangle}$ and $q \in X_H^{\langle k+2 \rangle}$ with $\Phi_\infty^{\langle k+2 \rangle}(p) = q$, the map $\Phi'_\infty : S_p^G \to S_q^H$ is a bijection. The result then follows because $\Phi_\infty^{\langle k+2 \rangle}$ is a bijection between $X_G^{\langle k+2 \rangle}$ and $X_H^{\langle k+2 \rangle}$, and the sets $\{S_p^G : p \in X_G^{\langle k+2 \rangle}\}, \{S_q^H : q \in X_H^{\langle k+2 \rangle}\}$ partition $X_{G'}, X_{H'}$.

We first claim that $\Phi'_\infty(S_p^G) \subseteq S_q^H$, which is to say, for every $p' \in X_{G'}$ such that $\pi^G(p') = p$, we have $\pi^H(\Phi'_\infty(p')) = q$. Diagramatically, we are claiming that the diagram in Figure 3.3 commutes. To see this, note that by construction of Φ', for all $p' \in X_{G'}$ and all $i \in \mathbb{Z}$, we have $\Psi^H(\Phi'_\infty(p')_i) = \Phi(\Psi^G(p'_i)) = \Phi^{\langle k+2 \rangle}(\Psi^G(p'_i))$. The condition $\pi^G(p') = \Psi_\infty^G(p') = p$ implies $\Psi^G(p'_i) = p_i$ for all $i \in \mathbb{Z}$. Combining the above with the observation that $\Phi^{\langle k+2 \rangle}(p_i) = q_i$ gives $\Psi^H(\Phi'_\infty(p')_i) = q_i$, which implies the claim.

To see that Φ'_∞ is injective on S_p^G, consider distinct points $p', p'' \in S_p^G$ which differ at index i. Since $\pi^G(p') = \pi^G(p'')$, we can assume without loss of generality that $p'_i = v_k^t$ and $p''_i = v_k^b$. Then

$$\Phi'(p'_{[i-k,i]}) = \Phi'(v_1 \cdots v_{k-1} v_k^t) = \Phi(v)_1^t \neq \Phi(v)_1^b = \Phi'(v_1 \cdots v_{k-1} v_k^b) = \Phi'(p''_{[i-k,i]}),$$

so $\Phi'_\infty(p') \neq \Phi'_\infty(p'')$.

($\Leftarrow$) Suppose there exists a k-block conjugacy $\Phi'_\infty : X_{G'} \to X_{H'}$. Without loss of generality, assume Φ'_∞ has no memory. By Lemma 3.3.3, for every $v \in V_G$, there exists $u \in V_H$ such that $\Phi'(v_{\text{in}} \cdots) = u_{\text{in}}$. Define the 1-block code $\Phi_\infty : X_G \to X_H$ by $\Phi(v) = \Psi^H(\Phi'(v_{\text{in}} v_1 \cdots v_{k-1}))$. We claim Φ_∞ is a conjugacy. To show this, we instead will show $\Phi_\infty^{\langle k+2 \rangle}$ defined by the same block map is a conjugacy. To see $\Phi_\infty^{\langle k+2 \rangle}$ is surjective, consider any $q \in X_H^{\langle k+2 \rangle}$. Picking any $q' \in S_q^H$, set $p' = \Phi'^{-1}_\infty(q')$ and $p = \pi^G(p')$. We will now show $\Phi_\infty^{\langle k+2 \rangle}(p) = q$, so the diagram in Figure 3.3 is commutative and Φ_∞ is surjective. For all $i \in \mathbb{Z}$,

$$\Phi(p_i) = \Psi^H(\Phi'((p_i)_{\text{in}} \cdots (p_i)_{k-1})) = \Psi^H((q_i)_{\text{in}}) = q_i. \tag{3.1}$$

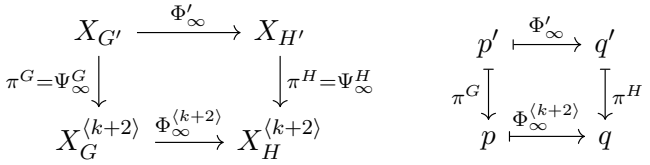

Figure 3.3: Given Φ' or Φ, one can construct the other such that this diagram commutes.

To see $\Phi_\infty^{\langle k+2\rangle}$ is injective, consider distinct $p_1, p_2 \in X_G^{\langle k+2\rangle}$. Then $S^G_{p_1} \neq S^G_{p_2}$. Since Φ'_∞ is a conjugacy, $\Phi'_\infty(S^G_{p_1}) \cap \Phi'_\infty(S^G_{p_2}) = \emptyset$. By Lemma 3.3.3, there exist $q_1, q_2 \in X_H^{\langle k+2\rangle}$ such that $S^H_{q_1} = \Phi'_\infty(S^G_{p_1})$ and $S^H_{q_2} = \Phi'_\infty(S^G_{p_2})$. By the construction of Φ (and shown in (3.1)), $\Phi_\infty^{\langle k+2\rangle}(p_1) = q_1 \neq q_2 = \Phi_\infty^{\langle k+2\rangle}(p_2)$, so $\Phi_\infty^{\langle k+2\rangle}$ is injective. □

Remark 3.3.6. The same construction could plausibly give a reduction from m-BC to ℓ-BC where $\ell = (m-1)(k+2)+k$, though if true, the proof would be much more involved.

Combining the reduction in Theorem 3.3.5 with Theorem 3.3.2 therefore gives GI-hardness for all k.

Corollary 3.3.7 (Theorem B)**.** *k-BC is GI-hard for all k.*

3.4 Reducing representation size

Thus far we have addressed two problems. We first gave an efficient algorithm, given directed graphs G, H and k-block map Φ, to verify whether $\Phi_\infty : X_G \to X_H$ is a conjugacy. We then showed that the problem of deciding whether X_G and X_H are conjugate, given only G and H, is GI-hard. We now address a problem given only G and an integer ℓ: whether we can find a k-block code which reduces the size of G by ℓ vertices while preserving conjugacy.

Definition 3.4.1. Given a directed graph G and integer ℓ, the **k-Block Reduction Problem**, denoted k-BR, is to decide if there exists a directed graph H with $|V_H| = |V_G| - \ell$ such that the vertex shifts X_G and X_H are conjugate via a k-block code.

We will show this problem is NP-complete for the case $k = 1$, by modifying the hardness proof of the State Amalgamation Problem (SAP), which asks if ℓ consecutive amalgamations can

be performed on a graph G [22]. The proof that SAP is NP-hard shows that the set of graphs satisfying a certain structure property is closed under the amalgamation operation. This structure is then leveraged to encode an NP-complete problem (Hitting Set). While 1-block codes are more general than sequences of amalgamations (Figure 2.6), we find that, surprisingly, the same set of graphs is also closed under 1-block conjugacy. In fact, the rest of the construction of [22] suffices as well, though much of the argument needs to be strengthened to the general 1-block case.

We begin by recalling the structure property.

Definition 3.4.2 ([22])**.** A directed graph G satisfies the **structure property** if it is essential and there exists a partition $\{\{\alpha\}, A, B, C\}$ of V_G such that the following four conditions hold.

(1) $N^+(\alpha) = N^-(\alpha) = \{\alpha\} \cup A \cup C$.

(2) For each $a \in A$, $N^-(a) = \{a, \alpha\}$ and $\{a, \alpha\} \subseteq N^+(a) \subseteq \{a, \alpha\} \cup B$.

(3) For each $c \in C$, $N^+(c) = \{c, \alpha\}$ and $\{c, \alpha\} \subseteq N^-(c) \subseteq \{c, \alpha\} \cup B$.

(4) For each $b \in B$, $N^-(b) \subseteq A$ and $N^+(b) \subseteq C$.

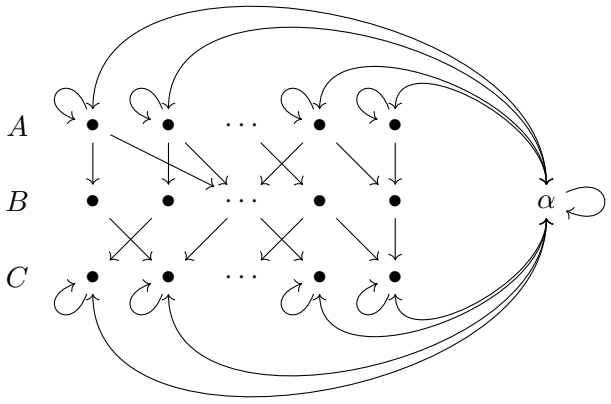

Figure 3.4: A graph which satisfies the structure property.

See Figure 3.4 for an example. We now show that the structure property is preserved under 1-block conjugacy.

Lemma 3.4.3. *Let G be a graph with the structure property having $\{\{\alpha\}, A, B, C\}$ as the partition of V_G, and let $\Phi_\infty : X_G \to X_H$ be a 1-block conjugacy. Then $\Phi(v_1) = \Phi(v_2)$ implies $v_1 = v_2$ or $v_1, v_2 \in B$. Thus H also satisfies the structure property where the vertex partition is $\{\{\Phi(\alpha)\}, \Phi(A), \Phi(B), \Phi(C)\}$.*

Proof. First note that if $\Phi : X_G \to X_H$ is a 1-block conjugacy from a graph G with vertex partition $\{\{\alpha\}, A, B, C\}$ such that $\Phi(v_1) = \Phi(v_2)$ implies $v_1 = v_2$ or $v_1, v_2 \in B$, then the fact that H satisfies the structure property with vertex partition $\{\{\Phi(\alpha)\}, \Phi(A), \Phi(B), \Phi(C)\}$ follows immediately. Now suppose for a contradiction that $v_1 \neq v_2 \in V_G$ and $\Phi(v_1) = \Phi(v_2)$; we proceed in cases.

Case 1: $v_1, v_2 \in \{\alpha\} \cup A \cup C$. Then $\Phi_\infty(v_1^\infty) = \Phi_\infty(v_2^\infty)$ and Φ_∞ is not a conjugacy.

Case 2: $v_1 = \alpha, v_2 \in B$. Let $a \in A, c \in C$ be such that av_2c is a word in X_G. (Such a, c exist as G is essential.) Then $\Phi_\infty((av_2c\alpha)^\infty) = \Phi_\infty((a\alpha c\alpha)^\infty)$ and Φ_∞ is not a conjugacy.

Case 3a: $v_1 \in A, v_2 \in B, (v_1, v_2) \notin E_G$. Let $a \in A$ be such that $(a, v_2) \in E_G$. Note that $a \neq v_1$. Consider the point

$$p = (\Phi(a)\Phi(v_2)\Phi(\alpha))^\infty = (\Phi(a)\Phi(v_1)\Phi(\alpha))^\infty$$

in X_H of period 3. Due to G having the structure property and our assumption that Φ_∞ is a 1-block conjugacy, the preimage of p must be defined by a 3-cycle whose vertices are contained in $\{\alpha\} \cup A \cup C$. In particular, the preimage must trace a self-loop, so we know $\Phi(a) = \Phi(\alpha)$ or $\Phi(a) = \Phi(v_1)$ or $\Phi(v_1) = \Phi(\alpha)$. Since we know Φ is injective on $\{\alpha\} \cup A \cup C$ by Case 1, none of these are possible.

Case 3b: $v_1 \in A, v_2 \in B, (v_1, v_2) \in E_G$. Let $c \in C$ be such that $(v_2, c) \in E_G$. Consider the point

$$p = (\Phi(v_2)\Phi(c)\Phi(\alpha))^\infty = (\Phi(v_1)\Phi(c)\Phi(\alpha))^\infty$$

in X_H of period 3. Again by the requirement that the preimage of p traces a self-loop, we know $\Phi(v_1) = \Phi(c)$ or $\Phi(v_1) = \Phi(\alpha)$ or $\Phi(\alpha) = \Phi(c)$. However, all of these situations violate the injectivity of Φ on $\{\alpha\} \cup A \cup C$.

Case 4: $v_1 \in C, v_2 \in B$. This is identical to Case 3 where the edges in the graph have been reversed. □

Pausing to note that 1-block conjugacies between graphs with the structure property are quite restricted, it tempting to believe that the added restriction of the structure property will prevent "tangling" of cycles as in Figure 2.6 and cause every 1-block conjugacy to be a sequence of only amalgamations; however, as shown in Figure 3.5, this also is not true.

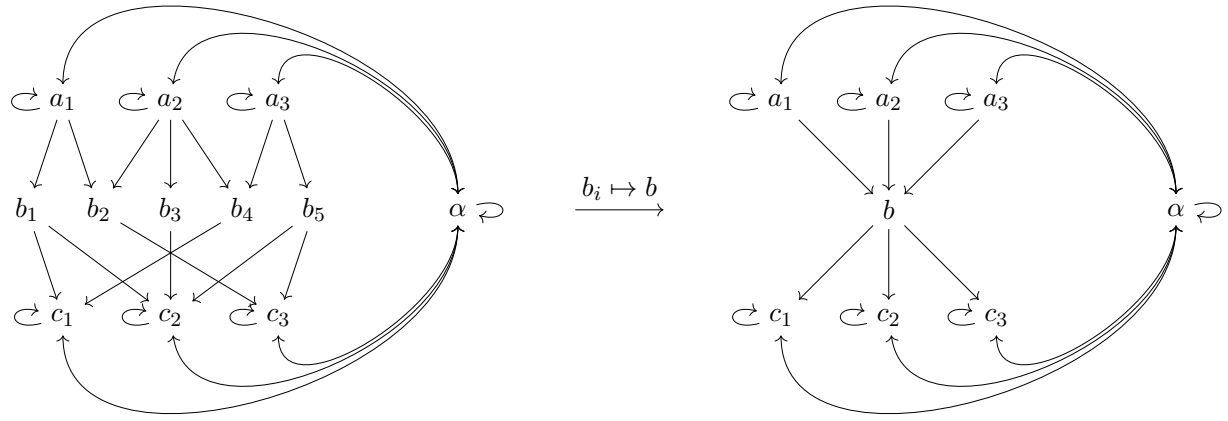

Figure 3.5: A minimal counterexample to the conjecture that any 1-block conjugacy $\Phi_\infty : X_G \to X_H$ between graphs with the structure property can be realized as a sequence of only amalgamations.

As in [22], we will need a "weight widget" which acts as a weighted switch, using the following notation. Let v be a vertex with $N^-(v) = D$ and $N^+(v) = E$. We will write $v : [D, E]$ in this situation, and as a slight abuse of notation, we will drop the curly brackets if E or D is a singleton and write $v : [u, E]$. Additionally, we extend this notation to sets S of vertices and write $S : [D, E]$ to mean $\bigcup_{s\in S} N^-(s) = D$ and $\bigcup_{s\in S} N^+(s) = E$.

Definition 3.4.4 ([22])**.** Let G satisfy the structure property with $V_G = \{\alpha\} \cup A \cup B \cup C$, and let $K > 0$ be a fixed even integer. Then for nonempty subsets $A_* \subseteq A, C_* \subseteq C$, the **weight widget** $w = \mathsf{weight}[A_*, C_*]$ is the following collection of vertices.

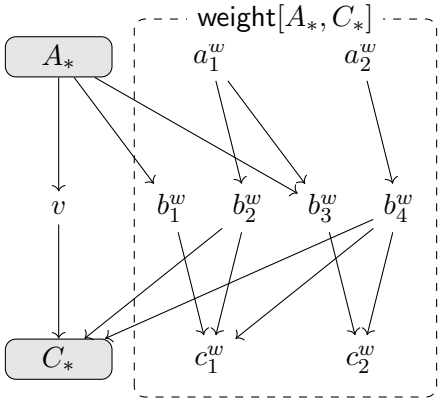

Figure 3.6: The weight widget weight$[A_*, B_*]$ with $K = 4$.

- $A_w = \{a_1^w, \ldots, a_{K/2}^w\}$
- $B_w = \{b_1^w \ldots, b_K^w\}$
- $C_w = \{c_1^w, \ldots, c_{K/2}^w\}$

where $A_w \cap A_* = \emptyset = C_w \cap C_*$, and for each $i \in \{1, \ldots, K/2\}$ we have

- $b_{2i-1} : [A_* \cup \{a_1^w, \ldots, a_{i-1}^w\}, c_i^w]$
- $b_{2i} : [a_i^w, C_* \cup \{c_1^w, \ldots, c_i^w\}]$.

Moreover, we require these to be the only images of A_w in B, i.e., $B \cap N^+(a_i^w) \subseteq B_w$ for all $a_i^w \in A_w$, and similarly for the preimage of C_w. For a given 1-block conjugacy Φ_∞, letting $S = \Phi^{-1}(\Phi(b_1^w)) \setminus B_w = \{b \in B : \Phi(b) = \Phi(b_1^w)\} \setminus B_w$, we say w is **activated** if $S : [A_*, C_*]$.

See Figure 3.6 for an example. The term "activate" comes from the following fact, which we show below in Lemma 3.4.6(1): if S is a singleton, then the construction of the weight widget allows the states $S \cup B_w$ to be amalgamated sequentially into a single state. For example, the vertex v in Figure 3.6 can activate the weight widget shown. The next two lemmas show that these amalgamations cannot be performed if the widget is not activated.

Lemma 3.4.5. *Let $w =$ weight$[A_*, C_*]$ be a weight widget in G. If $\Phi_\infty : X_G \to X_H$ is a 1-block conjugacy between graphs with the structure property, then for any $v \in V_H$, the statement $b_\ell^w \in \Phi^{-1}(v)$ for $\ell > 1$ implies $b_{\ell-1}^w \in \Phi^{-1}(v)$ or $|\Phi^{-1}(v)| = 1$.*

Proof. By contrapositive, suppose $|\Phi^{-1}(v)| > 1$ and there exists $b_\ell^w \in \Phi^{-1}(v)$ such that $b_{\ell-1}^w \notin \Phi^{-1}(v)$. Without loss of generality, let ℓ be the largest such subscript. We have two cases.

Case 1: ℓ is even. We claim there must exist $a \in N^-(v) \setminus \{\Phi(a_{\ell/2}^w), \ldots, \Phi(a_{K/2}^w)\}$. To see this claim, first note the weight widget construction guarantees that for every $b \in B \setminus B_w$, we have $N^-(b) \cap A_w = \emptyset$. Also by the weight widget construction, for any $b_{2i+1}^w \in B_w$, we have $A_* \subseteq N^-(b_{2i+1}^w) \setminus A_w$. Noting that $\mathsf{weight}[A_*, B_*]$ is not defined when $A_* = \emptyset$ and G is not essential when $N^-(b) = \emptyset$, if we have $\Phi^{-1}(v) \setminus \{b_{2i}^w : b_{2i}^w \in B_w\} \neq \emptyset$, then we must have $N^-(v) \setminus \Phi(A_w) \neq \emptyset$. That is, the claim is satisfied in the case when $\Phi^{-1}(v) \not\subseteq \{b_{2i}^w : b_{2i}^w \in B_w\}$. Assuming now that $\Phi^{-1}(v) \subseteq \{b_{2i}^w : b_{2i}^w \in B_w\}$, we note our earlier assumption that $|\Phi^{-1}(v)| > 1$ guarantees there exists $b_{2j}^w \in \Phi^{-1}(v)$ for some index $2j \neq \ell$. By our other assumption that ℓ is the largest such subscript, we have $2j < \ell$. Then $\Phi(a_j^w) \in N^-(v)$, and the claim follows. Proceeding, we then have $\Phi(a)v\Phi(c_{\ell/2}^w)$ is a word in X_H; however, $N^-(c_{\ell/2}^w) = \{b_{\ell-1}^w\} \cup \{b_\ell^w, b_{\ell+2}^w, \ldots, b_K^w\}$ and for all $b_{2i}^w \in N^-(c_{\ell/2}^w) \setminus \{b_{\ell-1}^w\}, (a, b_{2i}^w) \notin E_G$. Since $b_{\ell-1}^w \notin \Phi^{-1}(v)$ by assumption, this word has no preimage in X_G, so Φ_∞ is not a conjugacy.

Case 2: ℓ is odd. Using an argument symmetric to the one in case 1, we get that there must exist $c \in N^+(v) \setminus \{\Phi(c_{(\ell+1)/2}^w), \ldots, \Phi(c_{K/2}^w)\}$. Then $\Phi(a_{(\ell-1)/2}^w)v\Phi(c)$ is a word in X_H with no preimage, so Φ_∞ is not a conjugacy. □

Lemma 3.4.6. *Suppose $w = \mathsf{weight}[A_*, C_*]$ is a weight widget in G. Then*

(1) Suppose $\Phi_\infty : X_G \to X_{G'}$ is a 1-block conjugacy such that $\Phi^{-1}(\Phi(b_i^w)) = \{b_i^w\}$ for all $b_i^w \in B_w$ and $V_{G'}$ contains $v : [A_*, B_*]$. Defining $\Phi'_\infty : X_G \to X_H$ by

$$\Phi'(u) = \begin{cases} \Phi(u), & \text{if } u \notin \Phi^{-1}(v) \cup B_w \\ v, & \text{if } u \in \Phi^{-1}(v) \cup B_w \end{cases}$$

where H is the minimal graph induced by G and Φ', then Φ' is a 1-block conjugacy with $|V_H| = |V_{G'}| - K$.

(2) If $w = \mathsf{weight}[A_*, C_*]$ is not activated and $\Phi_\infty : X_G \to X_H$ is a 1-block conjugacy, then $\Phi^{-1}(\Phi(b_i^w))$ is a singleton for every b_i^w with $i > 1$.

Proof. (1) Consider the sequence of splittings followed by the sequence of amalgamations which transforms G into G'. Then note that by the construction of the weight widget, $\{v : [A_*, C_*]\} \cup B_w$ can be amalgamated sequentially for an additional K amalgamations.

(2) Suppose $b_1^w \in \Phi^{-1}(v)$ for some $v \in V_H$ and consider $V = \Phi^{-1}(v) \setminus B_w$. By Lemma 3.4.5 it suffices to show $b_2^w \notin \Phi^{-1}(v)$. By definition of w not being activated, we have two cases.

Case 1: $N^-(V) \neq A_*$. If there is some $a \in N^-(V) \setminus A_*$, then $\Phi(a)v\Phi(c_1^w)$ is a word in X_H. Since there is no state in G connecting a with c_1^w, the word has no preimage in X_G and Φ_∞ is not a conjugacy. Otherwise, there is some $a \in A_* \setminus N^-(V)$. By contrapositive, suppose $b_2^w \in \Phi^{-1}(v)$. Picking any $c \in C_*$, we have $\Phi(a)v\Phi(c)$ is a word in X_H. Since there is no state in $\Phi^{-1}(v)$ connecting a with c, the word has no preimage and Φ_∞ is not a conjugacy.

Case 2: $N^+(V) \neq C_*$. By contrapositive, suppose $b_2^w \in \Phi^{-1}(v)$. If there is some $c \in N^+(V) \setminus C_*$, then $\Phi(a_1^w)v\Phi(c)$ is a word in X_H. Since there is no state in G connecting a_1^w with c, the word has no preimage and Φ_∞ is not a conjugacy. Otherwise, there is some $c \in C_* \setminus N^+(V)$. Considering any $a \in N^-(V)$, we have $\Phi(a)v\Phi(c)$ is a word in X_H. Since there is no state in $\Phi^{-1}(v)$ connecting a with c, the word has no preimage and Φ_∞ is not a conjugacy. □

We now define the Hitting Set problem, which is NP-complete [38], and state a lemma which we will need in the proof.

Definition 3.4.7. Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a collection of sets with $\bigcup_i S_i = U$. Given a subset $S \subseteq U$, we define its **hit set** as $\text{hit}(S) = \{S_i : S \cap S_i \neq \emptyset\}$. Given $\mathcal{S}, U$, and an integer t, the **Hitting Set problem**, denoted HittingSet, is to decide whether there is a set H of cardinality t such that $\text{hit}(H) = \mathcal{S}$. We will also overload this notation, and write $\text{hit}(s)$ to mean $\text{hit}(\{s\})$ for $s \in U$.

Lemma 3.4.8 ([22])**.** *Let $(\mathcal{S}, U, t)$ be an instance of* HittingSet*. Suppose for some $t \leq |\mathcal{S}|$ there is no H with $|H| \leq t$ and hit$(H) = \mathcal{S}$. Then for all $H \subseteq U$, $|\text{hit}(H)| - |H| < |\mathcal{S}| - t$.*

We now show that 1-BR is NP-complete, by reduction from HittingSet. Given the lemmas developed above, the result essentially follows from the argument in [22], with minor modifications

for the 1-block case; for completeness, we give the full proof.

Theorem 3.4.9 (Theorem C)**.** 1-BR *is* NP*-complete.*

Proof. First we show 1-BR is in NP. Given a vertex shift X_G and $\Phi : V_G \to \{1, 2, \ldots, |V_G| - n\}$ from a proposed 1-block conjugacy Φ_∞, we construct the minimal image graph G' such that $\Phi_\infty : X_G \to X_{G'}$ is well-defined. In particular, $V_{G'} = \{\Phi^{-1}(u) : u \in V_G\}$ and $E_{G'} = \{(\Phi^{-1}(v_1), \Phi^{-1}(v_2)) : (v_1, v_2) \in E_G\}$. By Corollary 3.2.2.4, we can determine if Φ_∞ is a conjugacy in $O(|V_G|^4)$ time.

To show hardness, we reduce from HittingSet; let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be the collection of sets and t the given integer. Defining $n = |U|$ for $U = \bigcup_i S_i$, we set the parameter $K = 5mn$ for the weight widgets. Then, as in [22], we build the following graph $G = (V_G, E_G)$ with the structure property $V_G = A \cup B \cup C \cup \{\alpha\}$.

(1) Start with $A = \mathcal{S}, B = \emptyset, C = U \cup \{\beta\}$, where β is a new vertex.

(2) For each $u \in A, v \in C$, add $b_{uv} : [u, v]$. That is, add the vertex $b_{S_i s}$ and path $S_i \to b_{S_i s} \to s$ for every (s, S_i) with $s \in S_i$ as well as the vertex $b_{S_i \beta}$ and path $S_i \to b_{S_i \beta} \to \beta$ for every $S_i \in \mathcal{S}$.

(3) For each (s, S_i) with $s \in S_i$, add the weight widget $w = \mathsf{weight}[S_i, \{s, \beta\}] = (A_w, B_w, C_w)$. Note that A_w will added to A, B_w to B, and C_w to C.

(4) For each $s \in U$, add the weight widget $\mathsf{weight}[\mathrm{hit}(s), \{s\}]$.

(5) Finally, add the vertex α and the necessary edges for G to have the structure property, i.e., add the edges $\{(a, \alpha), (\alpha, a) : a \in A\} \cup \{(b, \alpha), (\alpha, b) : b \in B\} \cup \{(v, v) : v \in A \cup B \cup \{\alpha\}\}$.

Summarizing, if W is the collection of weight widgets added, $A = \mathcal{S} \cup \bigcup_{w \in W} A_w$, $B = \{b_{S_i s} : S_i \in \mathcal{S}, s \in S_i\} \cup \{b_{S_i \beta} : S_i \in \mathcal{S}\} \cup \bigcup_{w \in W} B_w$, and $C = U \cup \{\beta\} \cup \bigcup_{w \in W} C_w$. (See Figure 3.7 for an example with $\mathcal{S} = \{\{u_1, u_2\}, \{u_2, u_3\}\}$ where only steps (1), (2), and (5) have been performed.)

We will show there is a hitting set of size t if and only if there is a 1-block conjugacy $\Phi_\infty : X_G \to X_{G'}$ such that $|V_{G'}| \leq |V_G| - (m + n - t)K$. The idea behind the reduction is that

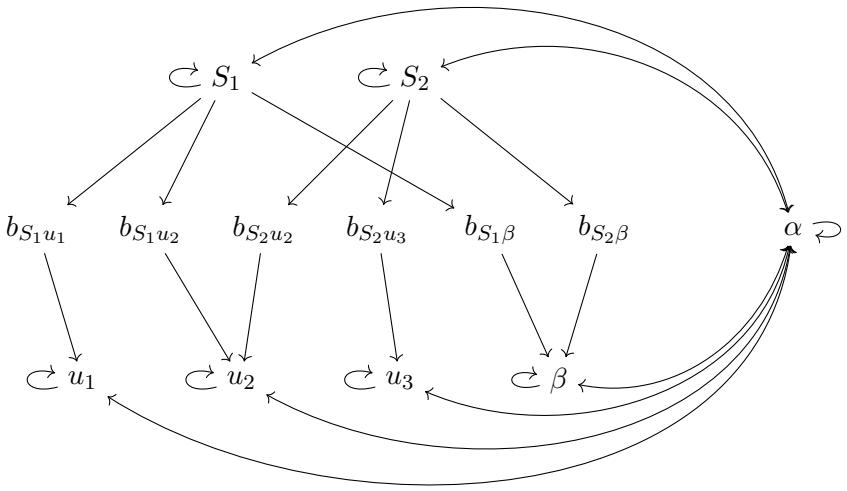

Figure 3.7: The graph constructed in Theorem 3.4.9 (without any weight widgets attached) for the HittingSet instance with $\mathcal{S} = \{\{u_1, u_2\}, \{u_2, u_3\}\}$.

s can either choose to be in the hitting set by combining some b_{S_is} with the appropriate $b_{S_i\beta}$ to activate some of the $\mathsf{weight}[S_i, \{s, \beta\}]$, or choose not to be in the hitting set by combining all b_{S_is} for $S_i \in \text{hit}(s)$ to activate $\mathsf{weight}[\text{hit}(s), \{s\}]$. We will be able to activate $|\text{hit}(H)|+|U \setminus H| = m+(n-t)$ weight widgets if there is a hitting set of size t and strictly fewer if no such set exists. By our choice of K, any reduction in the number of vertices not caused by activating weight widgets will be insignificant.

First, suppose there is a hitting set H for $\mathcal{S}$ of size t. We will give a sequence of $(m+n-t)K$ consecutive amalgamations, which together constitute a 1-block reducing the number of vertices by $(m+n-t)K$. For each $S_i \in \mathcal{S}$, pick some $s \in H$ such that $S_i \in \text{hit}(s)$. After amalgamating b_{S_is} with $b_{S_i\beta}$, the weight widget $w = \mathsf{weight}[S_i, \{s, \beta\}]$ can be activated and B_w amalgamated sequentially. Doing this for each S_i gives a total of $m(K+1) \geq mK$ consecutive amalgamations. As the above amalgamations only affected the vertices in B associated with H, next consider any $s \in U \setminus H$. We can amalgamate the vertices $\{b_{S_is} : S_i \in \text{hit}(s)\}$ in any order to form $b_{\text{hit}(s)s} : [\text{hit}(s), \{s\}]$ which can then activate $\mathsf{weight}[\text{hit}(s), \{s\}]$. Amalgamating all the vertices in this weight widgets give a total of at least K amalgamations for each vertex $s \in U \setminus H$. Thus we

can perform $mK + (n-t)K = (m+n-t)K$ consecutive amalgamations, so there is a 1-block conjugacy $\Phi_\infty : X_G \to X_{G'}$ such that $|V_G| \geq |V_{G'}| - (m+n-t)K$.

Next suppose there is no hitting set H of size t. Let $\Phi : X_G \to X_{G'}$ be a 1-block conjugacy such that $N = |V_G| - |V_{G'}|$ is as large as possible. Define

$$\begin{aligned}
\overline{H} &= \{s \in U : \mathsf{weight}[\mathrm{hit}(s), \{s\}] \text{ is activated}\},\\
F &= \{S_i : \mathsf{weight}[S_i, \{s, \beta\}] \text{ is activated for some } s \in S_i\},\\
H &= U \setminus \overline{H}.
\end{aligned}$$

Note that there is a single path in G from S_i to s, through the vertex $b_{S_i s}$, which is required to activate both $\mathsf{weight}[\mathrm{hit}(s), \{s\}]$ and $\mathsf{weight}[S_i, \{s, \beta\}]$. Thus for every $b_{S_i s}$ we have that if $S_i \in F$, then $s \in H$. That is,

$$F \subseteq \{S_i : s \in H \text{ for some } b_{S_i s}\}. \tag{3.2}$$

We now count how much smaller $|V_{G'}|$ could be than $|V_G|$. By construction, each activated widget can lead to reducing the number of vertices by at most K. Let $B_{\text{non-weight}} = \{b_{S_i s} : S_i \in \mathcal{S}, s \in S_i\} \cup \{b_{S_i \beta} : S_i \in \mathcal{S}\}$ be the vertices in B not in weight widgets. By Lemma 3.4.6, if $u \in \Phi^{-1}(v)$ with $|\Phi^{-1}(v)| > 1$ for some u not in an activated widget, then $u \in B_{\text{non-weight}} \cup \bigcup_{w \in W} w_1$. Thus V_G can be reduced by at most $(|F| + |\overline{H}|)K + |B_{\text{non-weight}}| + |W|$. Since

$$|B_{\text{non-weight}}| + |W| = (mn + m) + (mn + n) < K, \tag{3.3}$$

we have

$$\begin{aligned}
|V_G| - |V_{G'}| &\leq (|F| + |\overline{H}|)K + |B_{\text{non-weight}}| + |W| \\
&< (|F| + |\overline{H}|)K + K && \text{(by (3.3))} \\
&\leq (|\mathrm{hit}(H)| + (n - |H|) + 1)K && \text{(by (3.2) and } H = U \setminus \overline{H}) \\
&\leq (m + n - t)K && \text{(by Lemma 3.4.8).}
\end{aligned}$$

□

3.5 Edge shifts

Thus far we have restricted our attention to vertex shifts, rather than edge shifts, though the latter are perhaps more commonly used in the literature. For various reasons, the problems we consider are in general more appropriate for vertex shifts, as we discuss in §3.7. (Vertex shifts are also motivated by applications (§3.7).) Nonetheless, we now give some results for edge shifts, for the first two problems: verifying k-block conjugacies, and testing pairs of shifts for conjugacy. The third problem remains open for edge shifts.

In the following, we will leverage our results for vertex shifts, using the standard conversion from edge shifts to vertex shifts: edges become vertices, and pairs of adjacent edges become edges [51, Proposition 2.3.9]. More formally, we recall that given edge shift X_G^e, its vertex shift representation is the shift $X_{G'}$ where $V_{G'} = E_G$ and $E_{G'} = \{(e_i, e_j) : e_i e_j \text{ is a word in } X_G^e\}$. Thus, for any edge shifts X_G^e, X_H^e, there exists a k-block conjugacy $\Phi_\infty : X_G^e \to X_H^e$ if and only if there exists a k-block conjugacy $\Phi'_\infty : X_{G'} \to X_{H'}$ between the vertex shift representations of X_G and X_H.

First, we observe that our verification algorithm for vertex shifts immediately applies to edge shifts.

Theorem 3.5.1 (Theorem A)**.** *Given directed multigraphs G, H and a proposed k-block code $\Phi_\infty : X_G^e \to X_H^e$, deciding if Φ_∞ is a conjugacy can be determined in $O(|E_G|^{4k})$.*

Proof. Given edge shifts X_G^e, X_H^e, we first construct their vertex shift representations $X_{G'}, X_{H'}$ as above. Letting Φ'_∞ be the corresponding block code between the vertex shifts, by Corollary 3.2.2.4, we can determine if Φ'_∞ is a conjugacy in $O(|V_{G'}|^{4k}) = O(|E_G|^{4k})$ time. □

We now turn to the k-block conjugacy problem for edge shifts, where we again show GI-hardness.

Definition 3.5.2. Given directed mult-graphs G, H, the **k-Block Conjugacy Problem**, denoted k-BC^e, is to decide is there is a k-block conjugacy $\Phi_\infty : X_G^e \to X_H^e$ between the edge shifts X_G^e, X_H^e.

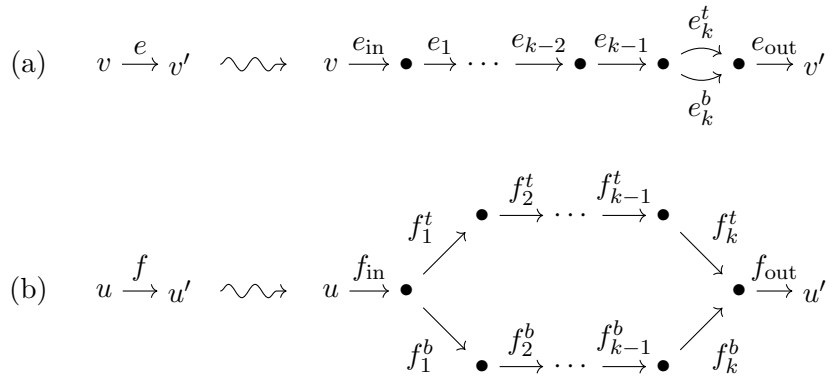

Figure 3.8: (a) The edge gadget for each pre-image graph. (b) The edge gadget for each image graph.

Theorem 3.5.3 (Theorem B)**.** *k-BC^e is GI-hard.*

Proof. We first show that 1-BC^e is GI-hard. Given directed graphs G, H with $|E_G| = |E_H|$, as in the vertex shift case, we will argue that there exists a 1-block conjugacy between the edge shifts if and only if the graphs are isomorphic. Suppose first that G, H are isomorphic. Let G', H' be the directed graphs for vertex shifts, as described above, so that $X_{G'} = X_G^e$ and $X_{H'} = X_H^e$. Since G, H are isomorphic and G', H' are created by an isomorphism-invariant deterministic procedure, G', H' are isomorphic. By Theorem 3.3.2, there exists a 1-block conjugacy $\Phi_\infty : X_{G'} \to X_{H'}$, so $X_{G'} = X_G^e$ is conjugate to $X_{H'} = X_H^e$ via a 1-block code.

Now suppose $\Phi_\infty : X_G^e \to X_H^e$ is a 1-block conjugacy. Noting that $\Phi : E_G \to E_H$ is a map on edges, we show that Φ can be realized as a map on V_G. To do this, it suffices to show (i) for any two edges $(v_1, v_2), (v_1, v_3)$ starting at the same vertex, $\Phi((v_1, v_2)), \Phi((v_1, v_3))$ also start at the same vertex and (ii) for any two edges $(u_1, u_2), (u_3, u_2)$ ending at the same vertex, $\Phi((u_1, u_2)), \Phi((u_3, u_2))$ also end at the same vertex. To see condition (i), consider any $(v_4, v_1) \in E_G$. As $\Phi((v_4, v_1)(v_1, v_2)), \Phi((v_4, v_1)(v_1, v_3))$ must both be words in X_H^e, we must have that $\Phi((v_1, v_2)), \Phi((v_1, v_3))$ both start at the same vertex. Similarly, for condition (ii), consider any $(u_2, u_4) \in E_G$, and note that $\Phi((u_1, u_2)(u_2, u_4)), \Phi((u_3, u_2)(u_2, u_4))$ are both words in X_H^e, so $\Phi(u_1, u_2), \Phi(u_3, u_2)$ must end at the same vertex. Thus Φ can be realized as a map $\Psi : V_G \to V_H$ on vertices which is surjective and preserves the edge/non-edge relation. To show Ψ is actually a

graph isomorphism, consider the inverse Φ_∞^{-1}. Since Φ_∞ is 1-block conjugacy and $|E_G| = |E_H|$, Φ_∞^{-1} is a 1-block code. Again, Φ_∞^{-1} can be realized as a surjective vertex map Ψ' which preserves the edge/non-edge relation. Since both $\Psi : V_G \to V_H$ and $\Psi' : V_H \to V_G$ are surjective maps between finite sets, we actually have Ψ, Ψ' are bijections. Thus Ψ is a graph isomorphism from G to H.

We now reduce 1-BC^e to k-BC^e, as we did with vertex shifts. Given edge shifts X_G^e, X_H^e, construct $\hat{G}, \hat{H}$ as follows. To form $\hat{G}$, substitute each edge in G with a path of length k followed by two parallel edges and a final edge (Figure 3.8a). Construct $\hat{H}$ by substituting each edge in H with a single edge followed by two parallel paths of length k followed by a single edge (Figure 3.8b). Then construct the vertex shift representations $G', H', \hat{G}', \hat{H}'$ of $G, H, \hat{G}, \hat{H}$. By construction of the edge gadget, $\hat{G}', \hat{H}'$ can be formed from G', H' by using the vertex gadget in Figure 3.2. Thus by Theorem 3.3.5, there exists a 1-block conjugacy $\Phi'_\infty : X_{G'} \to X_{H'}$ if and only if there exists a k-block conjugacy $\hat{\Phi}'_\infty : X_{\hat{G}'} \to X_{\hat{H}'}$. Since there exists a k-block conjugacy $\Phi_\infty : X_G^e \to X_H^e$ between edge shifts if and only if there exists a k-block conjugacy $\Phi'_\infty : X_{G'} \to X_{H'}$ between the vertex representations, there exists a 1-block conjugacy $\Phi_\infty : X_G \to X_H$ if and only if there exists a k-block conjugacy $\hat{\Phi} : X_{\hat{G}} \to X_{\hat{H}}$. Thus k-BC^e is 1-BC^e-hard and, in particular, GI-hard. □

3.6 Recognizing shifts of finite type

We now look at a larger class of shifts than shifts of finite type. In particular, we look at sofic shifts, which are the smallest class of shifts containing shifts of finite type which are closed under factors maps (i.e., quotients). We then investigate the problem: Given a sofic shift X, is X a shift of finite type?

Let $G = (V, E)$ be a directed graph. A **labeled graph** is a pair $\mathcal{G} = (G, \mathcal{L})$ where $\mathcal{L}$ is a labeling function $\mathcal{L} : V \to \mathcal{A}$ into some alphabet $\mathcal{A}$. Similarly an **edge-labeled multigraph** is a pair $\mathcal{G}^e = (G, \mathcal{L})$ where G is a multigraph and $\mathcal{L} : E \to \mathcal{A}$ is a labeling function from the edges of G into the alphabet $\mathcal{A}$. A labeled graph (or multigraph) $\mathcal{G} = (G, \mathcal{L})$ is **irreducible** if G is irreducible. Then, as with vertex and edge shifts, consider any point (i.e., bi-infinite walk) $p = \cdots p_{-1}.p_0 p_1 \cdots$ in X_G (or X_G^e). Considering that $\mathcal{L}$ defines a 1-block code $\mathcal{L}_\infty : X_G \to \mathcal{A}^{\mathbb{Z}}$,

the points $\{\mathcal{L}_\infty(x) : x \in X_G\}$ form a shift space [51, Theorem 3.1.4]. When $\mathcal{G}$ is a labeled graph, we denote this shift $X_\mathcal{G}$, and when $\mathcal{G}$ is an edge-labeled multigraph, we denote the shift $X^e_\mathcal{G}$. As another perspective, $X_\mathcal{G}$ (and similarly for $X^e_\mathcal{G}$) is the collection of all bi-infinite walks on $\mathcal{G}$ where the steps are labeled by the labeling $\mathcal{L}$ rather than the vertex (or edge) as was the case in vertex shifts (or edges shifts). Any shift X such that $X = X_\mathcal{G}$ for some labeled graph $\mathcal{G}$ is called a **sofic shift**. Any particular labeled graph $\mathcal{G}$ such that $X = X_\mathcal{G}$ is called a **presentation** of the sofic shift X. Note that, unlike vertex and edge shifts, there are many presentations of the same sofic shift. (Note that every SFT presented by a vertex shift has $|\mathcal{A}|$ vertices and every SFT presented by an edge shift has $|\mathcal{A}|$ edges. See Theorem 3.3.2 to see that all vertex shifts presenting the same SFT are isomorphic and Theorem 3.5.3 to see that all edge shifts presenting the same SFT are isomorphic.) See Figure 3.9 for an example of different presentations of the same sofic shift.

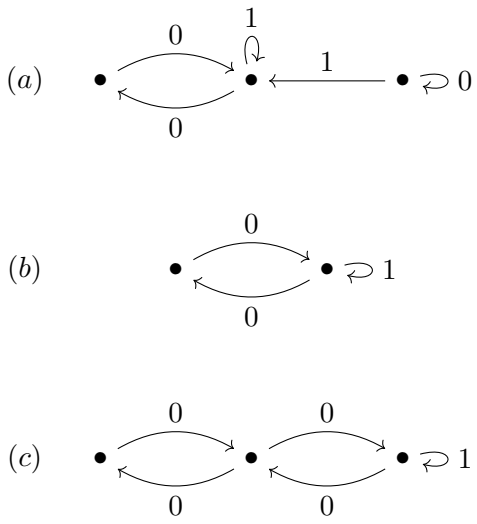

Figure 3.9: (a) A reducible presentation of the even shift which is right-resolving. (b) An irreducible presentation of the even shift which is right-resolving. (c) An irreducible presentation of the even shift which is not right-resolving.

Example 3.6.1. In particular, consider the subshift $X_\mathcal{F}$ of $\{0,1\}^\mathbb{Z}$ where the list of forbidden words is $\mathcal{F} = \{10^k1 : k \text{ is odd}\}$. This shift contains the points of $\{0,1\}^\mathbb{Z}$ where the length of every sequence of consecutive zeroes is either even or infinite. The shift $X_\mathcal{F}$ is called the **even shift**. There is no finite list of forbidden words describing the even shift, so the even shift is a sofic shift

which is not a shift of finite type. Figure 3.9 gives three presentations of the even shift.

Considering irreducible shifts as defined in Definition 2.3.1, we have in graph theoretic terms for sofic shifts that X is **irreducible** if there exists an irreducible presentation $\mathcal{G}$ for X [51, Proposition 3.3.11]. For an example, note that (b) and (c) in Figure 3.9 are irreducible graphs, so the even shift is an irreducible sofic shift, even though Figure 3.9(a) is not irreducible. To describe the situation of Figure 3.9, we say $(b), (c)$ are irreducible presentations of an irreducible shift, while (a) is a reducible presentation of an irreducible shift. Given a labeled graph $\mathcal{G}$, $\mathcal{G}$ is **right-resolving** (i.e., has a deterministic labeling in the forward direction) if for each $v \in V$, the out-neighbors $N^+(v)$ all have distinct labels. Or for edge-labeled multigraphs, $\mathcal{G}$ is **right-resolving** if for each $v \in G$, each edge leaving v has a distinct label. For an example, the presentations (a) and (b) in Figure 3.9 are right-resolving while the presentation (c) is not right-resolving. By [51, Theorem 3.3.2], every sofic shift has a right-resolving presentation which can be obtained via the subset construction; however, given a non-right-resolving presentation $\mathcal{G}$ of a sofic shift, the right-resolving presentation $\mathcal{G}'$ generated by the subset construction need not be polynomial in size compared to $\mathcal{G}$. In particular, $V_{\mathcal{G}'} \subseteq \mathcal{P}(V_{\mathcal{G}}) \setminus \{\emptyset\}$, so the only bound on $|V_{\mathcal{G}'}|$ is $|V_{\mathcal{G}'}| \leq 2^{|V_{\mathcal{G}}|} - 1$. Regardless, we restrict to right-resolving presentations and leave non-right-resolving presentations as an open question.

To recognize which sofic shifts are of finite type, we use a known classification. Unfortunately, this known classification relies on theory which has only been developed for sofic edge shifts. We conjecture the same classification and results hold for sofic vertex shifts, but we do not consider sofic vertex shifts further.

First we note that every irreducible sofic shift has a unique representation as its minimal right-resolving presentation [51, Theorem 3.3.18]. Even more, given any irreducible right-resolving presentation of a sofic shift, we can construct the minimal right-resolving presentation in $O(|V|^2)$ time [51, Theorem 3.4.14].

Consider any word $w \in \mathcal{B}(X_{\mathcal{G}}^e)$ such that every presentation of w in $\mathcal{G}$ terminates at the same

vertex. Such a word is a **synchronizing word** for $X_{\mathcal{G}}^e$. The following result is well-known.

Theorem 3.6.2 ([51, Theorem 3.4.17])**.** *Suppose $\mathcal{G}$ is the minimal right-resolving presentation of an irreducible sofic shifts. Then there exists N such that all words in $\mathcal{B}_N(X_{\mathcal{G}}^e)$ are synchronizing for $\mathcal{G}$ if and only if $X_{\mathcal{G}}^e$ is a shift of finite type.*

We now connect this result to cycles, so we can use the machinery developed in §3.2.

Proposition 3.6.3. *Let $\mathcal{G} = (G, \mathcal{L})$ be an edge-labeled multigraph which is the minimal right-resolving presentation of the the irreducible sofic shift $X_{\mathcal{G}}^e$. Then $\mathcal{G}$ contains two cycles which present the same word if and only if $X_{\mathcal{G}}^e$ is not a shift of finite type.*

Proof. ($\Rightarrow$) Suppose $\mathcal{G}$ contains two cycles c, d which present the same word. Then there is an index i such that $(c_i, c_{i+1}), (d_i, d_{i+1})$ are distinct edges. Then for every length m, the final m characters of $\mathcal{L}(c^\infty c_{[1,i+1]}), \mathcal{L}(d^\infty d_{[1,i+1]})$ form a word which is nonsynchronizing. Thus $X_{\mathcal{G}}^e$ is not of finite type by Theorem 3.6.2.

($\Leftarrow$) Suppose $X_{\mathcal{G}}^e$ is not of finite type. Then by Theorem 3.6.2, there is a nonsynchronizing word of every length. Let $w = w_1 w_2 \cdots w_n$ be a nonsynchronizing word of length $n = \binom{|E|}{2} + 1$. There are walks $a = a_1 a_2 \cdots a_{n+1}$ and $b = b_1 b_2 \cdots b_{n+1}$ such that a and b both present w but $(a_n, a_{n+1}) \neq (b_n, b_{n+1})$. Since $(a_n, a_{n+1}) \neq (b_n, b_{n+1})$ and $\mathcal{G}$ is right-resolving, $(a_i, a_{i+1}) \neq (b_i, b_{i+1})$ for $1 \leq i \leq n$.

Consider all possible pairs (e, e') of edges from $\mathcal{G}$ with $e \neq e'$. There are $\binom{|E|}{2}$ such pairs, so by the pigeonhole principle, there exist indices j, k with $j < k$ such that $(a_j, a_{j+1}) = (a_k, a_{k+1})$ and $(b_j, b_{j+1}) = (b_k, b_{k+1})$. Thus $a_j \cdots a_k$ and $b_j \cdots b_k$ are distinct cycles in $\mathcal{G}$ which present the same word. □

Theorem 3.6.4 (Theorem D)**.** *Let $\mathcal{G} = (G, \mathcal{L})$ be an edge-labeled multigraph which is an irreducible right-resolving presentation of the sofic shift $X_{\mathcal{G}}$. There is a $O(|V|^4)$-time algorithm to decide whether X_G is of finite type.*

	Block size	Verification (G, H, Φ)	Conjugacy (G, H)	Reduction (G, ℓ)
Vertex	$k = 1$	1-BV: P	1-BC: GI-hard, NP	1-BR: NP-complete
	$k > 1$	k-BV: P	k-BC: GI-hard, NP	k-BR: NP-complete??
Edge	$k = 1$	1-BV^e: P	1-BC^e: GI-hard, NP*	1-BR^e: NP-complete??
	$k > 1$	k-BV^e: P	k-BC^e: GI-hard, NP*	k-BR^e: NP-complete??

Table 3.1: Summary of results and open questions, for vertex and edge shifts. Question marks denote conjectures, and BV refers to the verification problem (§3.2). The asterisk (*) denotes a subtlety in edge shift representations: the k-block conjugacy problem is in NP when the the representation size is considered to be the number of edges (i.e., a unary representation), but membership in NP is not clear when the shift is given as an adjacency matrix (i.e., a binary representation).

Proof. By [51, Theorem 3.4.14], we can construct the minimal right-resolving presentation of $X_{\mathcal{G}}^e$ in $O(|V|^2)$ time, so without loss of generality, we can assume the given presentation is the minimal right-resolving presentation.

Let $\mathcal{A} = \{1, 2, \ldots, m\} = \mathcal{L}(E)$ be the alphabet of $X_{\mathcal{G}}^e$. Consider the 1-block code $\mathcal{L}_\infty : X_G^e \to \mathcal{A}^{\mathbb{Z}}$ from the edge shift on the underlying multigraph. By Proposition 3.6.3, $\mathcal{L}_\mathsf{c}$ is injective if and only if $X_{\mathcal{G}}^e$ is a shift of finite type. By Theorem 3.5.1, the injectivity of $\mathcal{L}_\mathsf{c}$ can be determined in $O(|E|^4)$ time. □

3.7 Discussion and future work

We have addressed several variants of the conjugacy problem restricted to k-block codes, with new algorithms to verify a proposed conjugacy, and hardness results for k-block conjugacy and representation reduction via 1-block codes (Table 3.1). Below we discuss subtleties of input representation, followed by applications and open problems.

Representations of SFTs. When considering how to describe a subshift of finite type (SFT), three representations come to mind: a vertex shift, an edge shift, and a list of forbidden words $\mathcal{F}$. As our results pertain to vertex and edge shifts, we now discuss some nuances in these two representations, leaving lists of forbidden words to future work.

Perhaps the central advantage of edge shifts over vertex shifts is their compact representation

size: a shift on n symbols can be represented in size as small as $O(\log n)$ by writing the multi-graph as a integer adjacency matrix, as opposed to $\Omega(n)$ for vertex shifts. This compact representation size can have important implications on the computational complexity. In the verification problem, for example, writing down a k-block code Φ naïvely takes $\Omega(n) = \Omega(|E_G|)$ space, which can be exponential in the size of the graphs G, H. (One can improve this by encoding Φ as a integer $|V_G| \times |V_G| \times |E_H|$ tensor, specifying how many $(u, v) \in E_G$ edges map to a given $e \in E_H$, but this can still be exponential.) Thus, while our algorithm remains polynomial-time, it would not be for cases allowing a compact representation of Φ.

Similarly, for the conjugacy problem, we only know k-BC^e to be in NP if we consider the graphs G, H to be represented in adjacency list form, which takes $\Omega(|E_G|)$ space, rather than the typically more compact integer adjacency matrix form taking $O(|V_G| \log |E_G|)$ space, as the natural certificate is the block map Φ witnessing the conjugacy. For the matrix representation of edge shifts, membership in NP would require a certificate exponentially smaller than the naïve representation of the block map Φ.

Finally, what "size reduction" means for edge shifts depends on the choice of adjacency list or matrix above. For the adjacency list, we have that the problem of reducing the number of vertices in the graph is in NP, but it is less motivated, as the size is dominated by $|E_G|$, not $|V_G|$. On the other hand, while the adjacency matrix representation size is dominated by $|V_G|$, it is not clear whether the problem of reducing the number of vertices is in NP, for the same reason as above.

Motivation from Markov partitions. As noted in [22], variants of the conjugacy problem for vertex shifts have applications in simplifying Markov partitions, a tool to study discrete-time dynamical systems via symbolic dynamics. Briefly, a Markov partition is a collection C of regions of the phase space, satisfying certain properties, which induces a conjugacy to a vertex shift X_G where $G = (C, E)$, i.e., the vertices are labeled with the regions of the phase space. In applications, one can encounter Markov partitions with thousands of regions, thus motivating the problem of simplifying the partition. Without additional information about the dynamical system, essentially the only way to do this while preserving the relevant geometric information is to coarsen the par-

tition, by replacing sets of regions with a single region which is their union. This operation is exactly a 1-block code. Our results therefore give an efficient algorithm to test whether a proposed coarsening (1-block code) is valid (yields a conjugacy). Our results also imply that the problem of minimizing the partition size is NP-complete. (Previous work [22] only showed the latter for the case where the 1-block code was a sequence of amalgamations.)

Open problems. Our work leaves several open problems, such as those implied by Table 3.1: resolving the complexity of the k-block conjugacy problem, and showing NP-hardness of the size reduction problem. The complexity of deciding k-block conjugacy between edge shifts represented as integer matrices is especially interesting, as membership in NP is perhaps unlikely (see above). Regarding the k-block conjugacy problem and resolving where it lies on the spectrum between GI-complete and NP-complete, we conjecture that, similar to the induced subgraph isomorphism problem [70], 1-BC is an NP-complete problem which happens to be GI-complete when $|V_G| = |V_H|$. Considering k-BC^e, it is plausible that k-BC^e is NEXP-complete as edge shifts seem to be a succinct representation of SFTs; this would be similar to results for succinct versions of other NP-complete problems [23, 59]. Beyond these questions, it would be interesting to address the complexity of k-block conjugacy between SFTs given as lists of forbidden words, and the natural variants of the problem for that input (for example, reducing the representation size of the list). And finally, as mentioned in §3.6, the problem of finding an efficient algorithm to recognize when a general presentation of a sofic shift (not necessarily an irreducible right-resolving edge presentation) is also an SFT is left open.

3.8 Algorithms

Function `IsInjective`(G, H, Φ)**:**

Input: irreducible graphs G, H and a 1-block code Φ

Output: true, if $\Phi_{\mathsf{c}} : \bigcup_n C_n(G) \to \bigcup_n C_n(H)$ is injective; false, otherwise

`/* Construct the meta-graph` M `*/`

$V_M \leftarrow V_G \times V_G$;

$E_M \leftarrow \{((v_1, v_2), (u_1, u_2)) : \Phi(v_1) = \Phi(v_2), \Phi(u_1) = \Phi(u_2),$ and $(v_1, u_1), (v_2, u_2) \in E_G\}$;

`/* Decide if` M `has a cycle passing through` (v_1, v_2) `with` $v_1 \neq v_2$ `*/`

$\mathcal{S} \leftarrow$ `GetStronglyConnectedComponents(`M`)` ; `/* Tarjan's */`

foreach subgraph s in $\mathcal{S}$ **do**

  if s is a singleton **then**

    continue;

  end

  foreach vertex (v_1, v_2) in s **do**

    if $v_1 \neq v_2$ **then**

      return true;

    end

  end

end

return false;

Algorithm 3.1: Determine if Φ_{c} is injective

Function `IsConjugacyIrreducible`(G, H, Φ)**:**

Input: irreducible graphs G, H and a 1-block code Φ

Output: true, if Φ_∞ is a conjugacy; false, otherwise

if not `IsInjective`(G, H, Φ) **then**

return false;

end

for $i \in \{1, \ldots, |V_G|\}$ **do**

if $\mathrm{tr}(A(G)^i) \neq \mathrm{tr}(A(H)^i)$ **then**

return false;

end

end

return true;

Algorithm 3.2: Determine if Φ_∞ between irreducible graphs is a conjugacy

Function `AddSinkComponents`(G, H, Φ)**:**

Input: reducible graphs G, H and a 1-block code Φ

Result: (1) alters G, H so each sink component T in H has the property $|V_{\Phi^{-1}(T)}| = 1$, and (2) extends Φ to the new graphs so $\Phi_\infty : X_G \to X_H$ is a conjugacy if and only if the original 1-block code was a conjugacy

$\mathcal{T} \leftarrow$ `GetSinkComponents(`H`)`;

foreach subgraph T in $\mathcal{T}$ **do**

$T' \leftarrow \Phi^{-1}(T)$;

if $|V_{T'}| = 1$ **then continue**;

`/* Find the subgraphs` C `and` C' `*/`

$v \leftarrow$ `GetRandomVertex(`T`)`;

$c \leftarrow$ `GetShortestCycleStartingAt(`v`)`;

$V_C \leftarrow \{u \in V_T : u \in c\}$;

$E_C \leftarrow \{(u, u') : uu' \text{ is a word of length 2 contained in } c^\infty\}$;

$V_{C'} \leftarrow \{u \in V_{T'} : \Phi(u) \in V_C\}$;

$E_{C'} \leftarrow \{(u, u') \in E_{T'} : (\Phi(u), \Phi(u')) \in E_C\}$;

`/* Attach the new vertices` t `and` t' `*/`

V_G`.Add(`t'`)`;

$N^+(t') \leftarrow \{t'\}$; $N^-(t') \leftarrow \{t'\}$;

foreach vertex u in $V_{C'}$ **do**

if $\Phi(u) = v \wedge$ there is a path in C' from u to a cycle **then** $N^-(t')$`.Add(`u`)`;

end

V_H`.Add(`t`)`;

$N^+(t) \leftarrow \{t\}$; $N^-(t) \leftarrow \{t, v\}$;

$\Phi(t') \leftarrow t$;

end

Algorithm 3.3: Turn every sink component into a single vertex

Function `AddSourceComponents`(G, H, Φ)**:**

Input: reducible graphs G, H and a 1-block code Φ

Result: (1) alters G, H so each source component S in H has the property $|V_{\Phi^{-1}(S)}| = 1$, and (2) extends Φ to the new graphs so $\Phi_\infty : X_G \to X_H$ is a conjugacy if and only if the original 1-block code was a conjugacy

G.`ReverseEdges()`;

H.`ReverseEdges()`;

`AddSinkComponents`(G, H, Φ);

G.`ReverseEdges()`;

H.`ReverseEdges()`;

Algorithm 3.4: Turn every source component into a single vertex

Function `IsConjugacyReducible`(G, H, Φ)**:**

Input: reducible graphs G, H and a 1-block code Φ

Output: true, if Φ_∞ is a conjugacy; false, otherwise

`AddSinkComponents`(G, H, Φ);

`AddSourceComponents`(G, H, Φ);

V_G.`Add`(v_G);

$N^-(v_G) \leftarrow$ `GetSinkVertices`(G);

$N^+(v_G) \leftarrow$ `GetSourceVertices`(G);

V_H.`Add`(v_H);

$N^-(v_H) \leftarrow$ `GetSinkVertices`(H);

$N^+(v_H) \leftarrow$ `GetSourceVertices`(H);

$\Phi(v_G) \leftarrow v_H$;

return `IsConjugacyIrreducible`(G, H, Φ);

Algorithm 3.5: Determine if Φ_∞ between reducible graphs is a conjugacy

Chapter 4

Determining isomorphism of quotients of genus 2 groups

We thank Gábor Ivanyos for helpful conversations and references during the work on this chapter.

4.1 Overview

In one view, [49] determines isomorphism of quotients of genus 1 groups via three basic steps.

(1) Classify all possible normal subgroups as either central or containing the commutator subgroup.

(2) Given G which is a quotient of some genus 1 group, recover the smallest genus 1 group, H, such that G is a quotient of H. Then write $G \cong H/N$ as a particular genus 1 group quotiented by a particular normal subgroup N.

(3) Given such quotients $G_1 \cong H/N_1$ and $G_2 \cong H/N_2$, decide if $G_1 \cong G_2$ by determining if there exists $\varphi \in \mathrm{Aut}(H)$ such that $\varphi(N_1) = N_2$.

While we use this idea as motivation, the first two steps fail to generalize to the genus 2 case, so we must find alternative solutions.

4.2 Genus 2 groups

Next we introduce some useful tools for determining isomorphism inside special classes of p-groups of exponent p and class 2, which will allow us to give the technical details behind the

definition of a genus 2 group. Then we apply a known classification of indecomposible genus 2 groups.

4.2.1 Baer's correspondence

Given a commutative ring C and left C-modules U, V, W, the map $b : U \times V \rightarrowtail W$ is **C-bilinear** if for every $u, u' \in U, v, v' \in V, c \in C$,

$$b(cu + u', v) = cb(u, v) + b(u', v),$$

$$b(u, cv + v') = cb(u, v) + b(u, v').$$

To every group, we associate a bilinear map which encodes commutation.

Definition 4.2.1.1. Given any group G, the alternating $\mathbb{Z}$-bilinear map $\mathrm{Bi}(G)$ associated to G is defined by

$$\mathrm{Bi}(G) : G/Z(G) \times G/Z(G) \rightarrowtail G'$$

$$(gZ(G), hZ(G)) \mapsto [g, h].$$

In the case of p-groups of exponent p, $\mathrm{Bi}(G)$ is an alternating $\mathbb{F}_q$-bilinear map for some field extension $\mathbb{F}_q$ of $\mathbb{F}_p$; that is, $\mathrm{Bi}(G)$ is a tensor over $\mathbb{F}_q$ for all the groups we will study in this thesis. Establishing conventions, when referring to $\mathrm{Bi}(G)$ in a coordinate-free way, we will use the notation $\mathrm{Bi}(G)$. We will often need to choose a basis, which in our context will always be over a finite field $\mathbb{F}_q$, at which points we will use the notation $\mathrm{Bi}_{\mathbb{F}_q}(G)$. After choosing a basis, we can refer to individual entries by using the notation $\mathrm{Bi}_{\mathbb{F}_q}(G)_{ijk}$ to represent the element of $\mathbb{F}_q$ stored in the (i, j, k)-entry. Picking an orientation for $\mathrm{Bi}_{\mathbb{F}_q}(G)$, we will write $\mathrm{Bi}_{\mathbb{F}_q}(G)_{**k}$ to be the kth slice in the G' direction. That is, after picking a bases $\{b_1, \ldots, b_m\}$ for $G/Z(G)$ and $\{c_1 \ldots, c_\ell\}$ for G', consider any $u, v \in G/Z(G)$. Then $u(\mathrm{Bi}_{\mathbb{F}_q}(G)_{**k})v^T$ gives the coefficient of c_k in $[u, v]$. While $\mathrm{Bi}_{\mathbb{F}_q}(G)_{i**}$ and $\mathrm{Bi}_{\mathbb{F}_q}(G)_{*j*}$ are also slices of $\mathrm{Bi}_{\mathbb{F}_q}(G)$ in other directions, they will not be used, so the word **slice** will be used only to mean $\mathrm{Bi}_{\mathbb{F}_q}(G)_{**k}$. Going back to individual entries, $\mathrm{Bi}_{\mathbb{F}_q}(G)_{ijk}$ is then the (i, j)-entry of the kth slice of $\mathrm{Bi}_{\mathbb{F}_q}(G)$.

While $\mathrm{Bi}(G)$ doesn't determine G in general, we will soon see that $\mathrm{Bi}(G)$ fully determines G when G has exponent p and $Z(G) = G'$. In the special case that G is a p-group of exponent p, we note that every abelian group of exponent p is isomorphic to the additive group $\mathbb{F}_p \times \cdots \times \mathbb{F}_p$, so we have $\mathrm{Bi}(G)$ is a bilinear map vector spaces over $\mathbb{F}_p$; that is, $\mathrm{Bi}(G)$ can be viewed as

$$\mathrm{Bi}(G) : \mathbb{F}_p^k \times \mathbb{F}_p^k \rightarrowtail \mathbb{F}_p^\ell$$

when G is a p-group of exponent p. Given an homomorphism $\varphi : G \to H$, it should be possible to push φ to a "homomorphism" from $\mathrm{Bi}(G)$ to $\mathrm{Bi}(H)$, and indeed it is. More formally, two bilinear maps $b_1, b_2 : U \times U \rightarrowtail W$ are **pseudo-isometric** if there exist isomorphisms $f : U \to U, \tilde{f} : W \to W$ such that $b_1(f(u), f(v)) = \tilde{f}(b_2(u, v))$; that is, b_1, b_2 are pseudo-isometric if $f, \tilde{f}$ are isomorphisms such that the following diagram commutes.

$$\begin{array}{ccc} U \times U & \xrightarrowtail{b_1} & W \\ {\scriptstyle f}\uparrow \quad {\scriptstyle f}\uparrow & & {\scriptstyle \tilde{f}}\uparrow \\ U \times U & \xrightarrowtail{b_2} & W \end{array}$$

Furthermore, for any $\mathbb{F}_q$-bilinear map $b : U \times U \rightarrowtail W$, we have the group of pseudo-isometries $\Psi\mathrm{Isom}(b)$ defined by

$$\Psi\mathrm{Isom}(b) = \{(f, \tilde{f}) \in \mathrm{GL}_{\mathbb{F}_q}(U) \times \mathrm{GL}_{\mathbb{F}_q}(W) : \text{for each } u, v \in U, b(f(u), f(v)) = \tilde{f}(b(u, v))\}.$$

Leading to a main component of our strategies in this chapter, we would like to compare isomorphism of groups with pseudo-isometry of bilinear maps. First consider any isomorphism $\varphi : G \to H$. Setting $f(gZ(G)) = \varphi(g)Z(G)$ and $\tilde{f}(g) = \varphi(g)$ gives a pseudo-isometry from $\mathrm{Bi}(G)$ to $\mathrm{Bi}(H)$. In picture, we have the following commutative diagram.

$$\begin{array}{ccc} \mathrm{Bi}(H) : H/Z(H) \times H/Z(H) & \longrightarrow & H' \\ \uparrow{\scriptstyle gZ(G) \mapsto \varphi(g)Z(H)} \uparrow & & \uparrow{\scriptstyle \varphi|_{G'}} \\ \mathrm{Bi}(G) : G/Z(G) \times G/Z(G) & \longrightarrow & G' \end{array}$$

In general pseudo-isometry of $\mathrm{Bi}(G), \mathrm{Bi}(H)$ does not imply isomorphism of G, H (in particular, $\mathrm{Bi}(D_8)$ and $\mathrm{Bi}(Q_8)$ are pseudo-isometric), so we now work toward a weakened version of the con-

verse. Reminding ourselves that p is assumed to be an odd prime, consider any alternating $\mathbb{F}_p$-bilinear map $b : U \times U \rightarrowtail W$. We define the **Baer group** $\mathrm{Gp}(b)$ to be the set $U \times W$ equipped with multiplication defined by $(u, w) \cdot (u', w') = (u + u', w + w' + \frac{1}{2}b(u, u'))$.[1] Suppose $(f, \tilde{f})$ is a pseudo-isometry from b_1 to b_2. Then

$$\varphi : \mathrm{Gp}(b_1) \to \mathrm{Gp}(b_2)$$
$$(u, w) \mapsto (f(u), \tilde{f}(v))$$

is an isomorphism, so b_1, b_2 being pseudo-isometric implies $\mathrm{Gp}(b_1), \mathrm{Gp}(b_2)$ are isomorphic. To strengthen this result, we claim that $\mathrm{Gp}(\mathrm{Bi}(G)) \cong G$ whenever G is a p-group of exponent p and with $Z(G) = G'$. To give a feel for this isomorphism, as well as evidence toward this claim, we show G and $\mathrm{Gp}(\mathrm{Bi}(G))$ are both class 2 groups of the same size whose commutation maps match. First the fact that $|G| = |\mathrm{Gp}(\mathrm{Bi}(G))|$ is clear from the assumption that $Z(G) = G'$. Next, noting that $\mathrm{Bi}(G)$ being alternating implies $\mathrm{Bi}(G)(u, u) = 0$ for all $u \in U$, observe that $(u, w)^{-1} = (-u, -w)$ as

$$(u, w) \cdot (-u, -w) = \big(u - u, w - w + \tfrac{1}{2}\mathrm{Bi}(G)(u, -u)\big) = \big(0, \tfrac{-1}{2}\mathrm{Bi}(G)(u, u)\big) = (0, 0).$$

Calculating a generic commutator, we see

$$\begin{aligned}
[(u, w), (u', w')] &= ((-u, -w) \cdot (-u', -w')) \cdot ((u, w) \cdot (u', w')) \\
&= \big(-u - u', -w - w' + \tfrac{1}{2}\mathrm{Bi}(G)(u, u')\big) \cdot \big(u + u', w + w' + \tfrac{1}{2}\mathrm{Bi}(G)(u, u')\big) \\
&= \big(0, \mathrm{Bi}(G)(u, u') + \tfrac{1}{2}\mathrm{Bi}(G)(-(u + u'), u + u')\big) \\
&= (0, \mathrm{Bi}(G)(u, u') - \mathrm{Bi}(G)(u + u', u + u')) \\
&= (0, \mathrm{Bi}(G)(u, u')).
\end{aligned}$$

Thus, $\mathrm{Gp}(\mathrm{Bi}(G))' \leq 0 \times W \leq Z(\mathrm{Gp}(\mathrm{Bi}(G)))$; that is, $\mathrm{Gp}(\mathrm{Bi}(G))$ is nilpotent of class at most 2. Finally, consider any $g, h \in G$. Then $g = \overline{g} \cdot z_1, h = \overline{h} \cdot z_2$ for $\overline{g}, \overline{h} \in G/Z(G)$ and $z_1, z_2 \in Z(G)$, so $[g, h] = b(\overline{g}, \overline{h})$. By the calculation above, $[(\overline{g}, z_1), (\overline{h}, z_2)] = (0, b(\overline{g}, \overline{h}))$, so G and $\mathrm{Gp}(\mathrm{Bi}(G))$ have matching commutation maps. While we have not given the full proof here, the preceding discussion

[1] Note that division by 2 is defined in every field of odd order.

gives the main idea for the following result of Baer, which is fundamental to the remainder of the chapter.

Theorem 4.2.1.2 ([12]). *Two p-groups G_1, G_2 of exponent p with $Z(G_i) = G_i'$ are isomorphic if and only if* $\mathrm{Bi}(G_1)$ *and* $\mathrm{Bi}(G_2)$ *are pseudo-isometric.*

Related, but not necessary for our arguments, is that b and $\mathrm{Bi}(\mathrm{Gp}(b))$ are pseudo-isometric under the following niceness assumption on b. Define an alternating bilinear map $b : V \times V \rightarrowtail W$ to be **nondegenerate** if $u \neq 0$ implies there is some $v \in V$ such that $b(u, v) \neq 0$. Note that this is precisely the analogue of assuming $Z(G) = G'$; if b is nondegenerate, then $G = \mathrm{Gp}(b)$ satisfies $Z(G) = G'$, and conversely, if G satisfies this condition, then $\mathrm{Bi}(G)$ is nondegenerate. We mention for completeness that if $b : V \times V \rightarrowtail W$ is a nondegenerate alternating bilinear map and $b(V, V) = W$, then b and $\mathrm{Bi}(\mathrm{Gp}(b))$ are pseudo-isometric.

4.2.2 Genus of a group

As mentioned in §2.5, we are concerned with p-groups of exponent p for some odd prime p such that $Z(G) = G'$, and the primary way we will approach Gpl is through pseudo-isometry of $\mathrm{Bi}(G)$. Now, as in [20], and using ideas first introduced by Knebelman in [41] for Lie algebras, we associate a notion of genus to every group of nilpotence class 2. Given a bilinear map $b : U \times V \rightarrowtail W$, the **centroid** of b, written $\mathrm{Cent}(b)$, is the largest ring C over which b is C-bilinear. Namely, $\mathrm{Cent}(b)$ is the ring

$$\mathrm{Cent}(b) = \left\{ \begin{array}{c} (f, g, h) \in \mathrm{End}(U) \times \mathrm{End}(V) \times \mathrm{End}(W) : \text{for all } u \in U, v \in V, \\ b(f(u), v) = h(b(u, v)) = b(u, g(v)) \end{array} \right\}.$$

Coming back to groups, it is clear from this explicit description of $\mathrm{Cent}(\mathrm{Bi}(G))$ that $\mathrm{Cent}(\mathrm{Bi}(G))$ always exists and can be found as the solution to a system of linear equations. As also mentioned earlier, it suffices to consider directly indecomposible groups. By [77], if G is directly indecomposible, then $\mathrm{Cent}(\mathrm{Bi}(G))$ is a local ring. Letting $J = \mathrm{Jac}(\mathrm{Cent}(\mathrm{Bi}(G)))$ be the Jacobson radical of the centroid of $\mathrm{Bi}(G)$, we have $G'/G'J$ is a vector space over the field $\mathrm{Cent}(\mathrm{Bi}(G))/J$. We define

the **rank** of G' to be the dimension of $G'/G'J$ over $\mathrm{Cent}(\mathrm{Bi}(G))/J$. For any group G, the group $\mathrm{Gp}(\mathrm{Bi}(G))$ decomposes as $G_1 \times \cdots \times G_m$ where each G_i is directly indecomposible. The **genus** of a class 2 group is defined to be the maximum rank of $[G_i, G_i]$ as a $\mathrm{Cent}(\mathrm{Bi}(G_i))$-module.

4.2.3 Classification of indecomposible genus 2 groups

In an attempt to simplify the problem of testing isomorphism, we wish to subdivide our groups as much as possible. A more general way to split up a group than as a direct product is as a central product.

Definition 4.2.3.1. A **central decomposition** of a group G is a collection of subgroups $\mathcal{H}$ such that:

(1) $\langle \mathcal{H} \rangle = G$,

(2) For each $H \in \mathcal{H}$, $\langle \mathcal{H} \setminus \{H\} \rangle \neq G$,

(3) For each $H \in \mathcal{H}$, $[H, \langle \mathcal{H} \setminus \{H\} \rangle] = 1$.

If $\{G\}$ is the only central decomposition of G, then G is called **centrally indecomposible**.

Note that G being centrally indecomposible implies G is directly indecomposible. Also, we can decompose a group as a central product efficiently [76, 75]. While an isomorphism test which works on directly indecomposible groups immediately gives an isomorphism test for groups which are direct products, the same is not true for central products. Regardless, the case of centrally indecomposible groups is difficult enough; we proceed with the assumption that our groups are centrally indecomposible and the hope that the strategies developed in [20] for central products of genus 2 groups generalize to the case of quotients of central products of genus 2 groups.

Going back to our context of p-groups G of exponent p with $Z(G) = G'$, $\mathrm{Cent}(\mathrm{Bi}(G)) \cong \mathbb{F}_{p^n}$ for some n and the genus of G is the dimension of G' over $\mathbb{F}_{p^n}$. We will be concerned with quotients of genus 2 groups, so we start with the following classification of indecomposible genus 2 groups.

Theorem 4.2.3.2 ([20, Theorem 1.2])**.** *A centrally indecomposible p-group of exponent p and genus 2 over a field $\mathbb{F}_{p^n}$ is isomorphic to one of the following types of groups:*

*(1) (**a genus 2 sloped group**) a quotient $H(R)/N$ of a Heisenberg group,*

$$H(R) = \left\{ \begin{bmatrix} 1 & e & z \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix} : e, f, z \in R = \mathbb{F}_{p^n}[x]/(a(x)^c),\ a(x) \text{ irreducible} \right\},$$

by a central subgroup N, where N is a $\mathbb{F}_{p^n}$-subspace of codimension 2 in H';[2] *or*

*(2) (**a genus 2 flat group**) the matrix group*

$$H_m^\flat(\mathbb{F}_{p^n}) = \left\{ \left[\begin{array}{c|ccccc} & e_1 & \cdots & e_m & 0 & z_1 \\ I_2 & 0 & e_1 & \cdots & e_m & z_2 \\ \hline & & & & & f_0 \\ & & & I_{m+1} & & \vdots \\ & & & & & f_m \\ \hline & & & & & 1 \end{array}\right] : e_i, f_i, z_i \in \mathbb{F}_{p^n} \right\}$$

When the field and m are either understood or unimportant, we will abbreviate $H_m^\flat(\mathbb{F}_{p^n})$ as $H^\flat$.

As determining isomorphism between genus 2 groups is solvable in polynomial time [20], we attempt to generalize the results of [49], which shows isomorphism between quotients of genus 1 groups can be determined in polynomial time, to the case of quotients of genus 2 groups.

4.3 Quotients of genus 2 groups by non-central subgroups

As our desire is to determine isomorphism in quotients of genus 2 groups, we now consider the possible normal subgroups of genus 2 groups. In the genus 1 case, we have the following nice

[2] As written, the idea is correct, but it doesn't actually make sense since $H' \cong \mathbb{Z}/p\mathbb{Z} \times \cdots \times \mathbb{Z}/p\mathbb{Z}$ is not a $\mathbb{F}_{p^n}$-vector space. More correctly, we could say $1 - N$ is a $\mathbb{F}_{p^n}$-subspace of codimension 2 in H' as matrices over $\mathbb{F}_{p^n}$. Alternatively, noting that H' is elementary abelian (and hence, $H' \cong \mathbb{F}_p \times \cdots \times \mathbb{F}_p$), endow a $\mathbb{F}_{p^n}$ scalar multiplication structure from $H(R)$ written over $\mathbb{F}_{p^n}$. Then N is a $\mathbb{F}_{p^n}$-subspace of codimension 2 in H' in this induced $\mathbb{F}_{p^n}$-vector space.

classification of normal subgroups.

Proposition 4.3.1 ([49, Lemma 3.2])**.** *Suppose G is a directly indecomposible genus 1 group with $Z(G) = G'$ and $N \trianglelefteq G$ is any normal subgroup. Then $N \leq Z(G)$ or $G' \leq N$.*

Unfortunately, this result does not generalize to the genus 2 case, as witnessed in the following counterexample.

Example 4.3.2. Consider the subgroup N of $H_m^\flat(\mathbb{F}_q)$ defined by

$$N = \left\{ \left[\begin{array}{c|c|c} I_2 & 0 & \begin{matrix} z_1 \\ 0 \end{matrix} \\ \hline & I_{m+1} & \begin{matrix} f_0 \\ 0 \\ \vdots \\ 0 \end{matrix} \\ \hline & & 1 \end{array}\right] : z_1, f_0 \in \mathbb{F}_q \right\}.$$

Then $N \trianglelefteq H_m^\flat(\mathbb{F}_q)$ but N is neither central nor contains the center.

Supposing $N \trianglelefteq G$ is a normal subgroup of a genus 2 group G there are three possibilities:

(1) $Z(G) \leq N$,

(2) $N \leq Z(G)$,

(3) $N \not\leq Z(G)$ and $Z(G) \not\leq N$.

In the case where $Z(G) = G' \leq N$, the quotient G/N is elementary abelian, so isomorphism is easy to test [35, 39, 64, 72]. The case where $N \leq Z(G)$ will be addressed in §4.4, so we now address the final case of non-abelian quotients by normal subgroups N where N is not central. Supposing G_1, G_2 are two such groups, we will show determining if $G_1 \cong G_2$ can be decided in polynomial time by showing G_i is also a quotient of a genus 1 group and then appealing to [49].

Suppose $G \cong H/N$ is a quotient of a genus 2 group where $N \not\leq Z(H)$ and $H' \not\leq N$. We will show N contains a full 1 dimensional $\mathbb{F}_q$-subspace of H'. Explicitly for the flat genus 2 groups, consider the subgroups

$$Z_1 = \left\{ \left[\begin{array}{c|c|c} I_2 & 0 & \begin{matrix} z_1 \\ 0 \end{matrix} \\ \hline 0 & I_{m+1} & 0 \\ \hline 0 & 0 & 1 \end{array}\right] : z_1 \in \mathbb{F}_q \right\} \text{ and } Z_2 = \left\{ \left[\begin{array}{c|c|c} I_2 & 0 & \begin{matrix} 0 \\ z_2 \end{matrix} \\ \hline 0 & I_{m+1} & 0 \\ \hline 0 & 0 & 1 \end{array}\right] : z_2 \in \mathbb{F}_q \right\} \tag{4.1}$$

of $H_m^\flat(\mathbb{F}_q)$. We will show that any normal subgroup $N \trianglelefteq H_m^\flat(\mathbb{F}_q)$ such that $N \not\leq Z(H_m^\flat(\mathbb{F}_q))$ contains either Z_1 or Z_2.

Lemma 4.3.3. *Suppose $N \trianglelefteq H_m^\flat(\mathbb{F}_q)$ such that $N \not\leq Z(H_m^\flat(\mathbb{F}_q))$. Then N contains a 1 dimensional $\mathbb{F}_q$-subspace of $H_m^\flat(\mathbb{F}_q)'$; in particular, $Z_1 \leq N$ or $Z_2 \leq N$, where Z_1, Z_2 are the subgroups defined in (4.1).*

Proof. Let $A \in N \setminus Z(H_m^\flat(\mathbb{F}_q))$. Writing A as a tuple, we have

$$A = ((e_1, \ldots, e_m), (f_0, \ldots, f_m), z_1, z_2) = (\overline{e}, \overline{f}, z_1, z_2).$$

As N is normal, $[A, B] \in N$ for each $B \in H_m^\flat(\mathbb{F}_q)$. Writing out the multiplication, we get

$$[(\overline{e}, \overline{f}, z_1, z_2), (\overline{g}, \overline{h}, y_1, y_2)] = (\overline{0}, \overline{0}, \overline{e} \cdot \overline{h}_{[0,m-1]} - \overline{g} \cdot \overline{f}_{[0,m-1]}, \overline{e} \cdot \overline{h}_{[1,m]} - \overline{g} \cdot \overline{f}_{[1,m]}) \in N.$$

As $A \notin Z(H_m^\flat(\mathbb{F}_q))$, one of $e_i, f_j \neq 0$.

Case 1: There exists i such that $e_i \neq 0$. Letting $z \in \mathbb{F}_q$ be arbitrary and commutating by $B = (\overline{0}, (0, \ldots, 0, e_i^{-1}z, 0, \ldots, 0), 0, 0))$ where $e_i^{-1}z$ is at index i, we have $[A, B] = (\overline{0}, \overline{0}, 0, z)$ so $Z_2 \leq N$.

Case 2: There exists i such that $f_i \neq 0$.

Subcase 2a: $f_0 = 0$. Letting $z \in \mathbb{F}_q$ be arbitrary and commutating by $B = ((-f_0^{-1}z, 0, \ldots, 0), \overline{0}, 0, 0)$, we have $[A, B] = (\overline{0}, \overline{0}, z, 0)$ so $Z_1 \leq N$.

Subcase 2b: $f_i = 0$ for some $i \neq 0$. Letting $z \in \mathbb{F}_q$ be arbitrary and commutating by $B = ((0, \ldots, 0, -f_i^{-1}z, 0, \ldots, 0), \overline{0}, 0, 0)$ where $-f_i^{-1}z$ is at index i, we have $[A, B] = (\overline{0}, \overline{0}, 0, z)$ so $Z_2 \leq N$. □

While the contained $\mathbb{F}_q$-subspace cannot be written as explicitly for sloped genus 2 groups, the same result still holds.

Lemma 4.3.4. *Suppose $G = H(R)/M$ is a sloped genus 2 group with $R = \mathbb{F}_q[x]/(a(x)^c)$ and $N \trianglelefteq G$ such that $N \not\leq Z(G)$. Then N contains a 1 dimensional $\mathbb{F}_q$-subspace of G'.*

Proof. Suppose $N \trianglelefteq G = H(R)/M$ such that $N \not\leq Z(G)$. Lifting to $H(R)$, there exists $\hat{N} \trianglelefteq H(R)$ such that $M \leq \hat{N}$ and $\hat{N} \not\leq Z(H(R))$. As there exist $\overline{A} \in N \setminus Z(G)$ and $\overline{B} \in G$ such that $[\overline{A}, \overline{B}] \neq 1$, there exist

$$A = \begin{bmatrix} 1 & e & z \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix} = (e, f, z) \in \hat{N} \setminus Z(H(R)) \text{ and } B = (g, h, y) \in H(R)$$

such that $[A, B] = (0, 0, eh - fg) \notin M$. Then for each $\alpha \in \mathbb{F}_q$, the matrix $[A, (\alpha g, \alpha h, z)] = (0, 0, \alpha(eh - fg)) \in \hat{N} \setminus M$. Thus, $\hat{N}$ contains a 1 dimensional $\mathbb{F}_q$-subspace of $H(R)'$ which is not contained in M; that is, N contains a 1 dimensional $\mathbb{F}_q$-subspace of G'. □

Theorem 4.3.5 (Theorem E)**.** *If G is a quotient of a centrally indecomposible group of genus 2 by a non-central subgroup, then G is a quotient of a genus 1 group. In particular, testing isomorphism of such quotients can be done in polynomial time.*

Proof. Suppose $G = H/N$ is a quotient of a centrally indecomposible genus 2 group by a non-central subgroup. By Lemmas 4.3.3 and 4.3.4, N contains a 1 dimensional $\mathbb{F}_q$-subspace of H'. This subspace is a subgroup $L \leq N \leq H$. Then $H/N \cong (H/L)/(N/L)$. As H is centrally indecomposible, $\dim_{\mathbb{F}_q}(H') = 2$, so $\dim_{\mathbb{F}_q}((H/L)') = 1$ and H/L is a genus 1 group. Thus H/N is a quotient of a genus 1 group.

Finally, the fact that testing isomorphism of such groups can be done in polynomial time follows by [49]. □

4.4 Quotients of genus 2 flat groups by central subgroups

Given G which is a quotient of a genus 1 group, a genus 1 group H of which G is also a quotient can be found canonically by looking at the adjoint of $\mathrm{Bi}(G)$.

Definition 4.4.1. Given a bilinear map $b : U \times V \rightarrowtail W$, the **adjoint** of b, written $\mathrm{Adj}(b)$, is the largest ring A such that b factors through the tensor product $\otimes_A : U \times V \to U \otimes_A V$. Or more explicitly, the adjoint of b is

$$\mathrm{Adj}(b) = \{(f, g) \in \mathrm{End}(U) \times \mathrm{End}(V)^{\mathrm{op}} : b(f(u), v) = b(u, g(v))\}.$$

Then G is a quotient of the genus 1 group $H_1(Z(\mathrm{Adj}(\mathrm{Bi}(G))))$ [49], where $H_m(\mathbb{F}_q)$ is a generalized Heisenberg group defined in Example 2.5.2.1.

This result also does not generalize to the case of genus 2 groups as shown by the following example.

Example 4.4.2. Consider $G = H_1^\flat(\mathbb{F}_p[x]/(x^2 + ax - 1))/N$, where a is any constant such that $x^2 + ax - 1$ is irreducible and

$$N = \left\{ \left[\begin{array}{c|c|c} & & 0 \\ I_2 & 0 & \\ & & bx \\ \hline 0 & I_2 & 0 \\ \hline 0 & 0 & 1 \end{array}\right] : b \in \mathbb{F}_p \right\}.$$

A straightforward calculation shows

$$\mathrm{Adj}\left(\mathrm{Bi}(G)\right) \cong \left\{ \left[\begin{array}{c|cc} \alpha & \gamma & M \\ \hline & \beta & \\ & & \beta \end{array}\right] : \alpha, \beta, \gamma \in \mathbb{F}_{p^2} \hookrightarrow M_{2\times 2}(\mathbb{F}_p), M \in M_{2\times 2}(\mathbb{F}_p) \right\},$$

so $Z\left(\mathrm{Adj}\left(\mathrm{Bi}(G)\right)\right) \cong \mathbb{F}_p$.

4.4.1 Recovering the genus 2 group

Similar to the genus 1 case, given a group G which is a quotient of $H_m^\flat(\mathbb{F}_q)$ by a central subgroup, we would like to have a way to recover $H_m^\flat(\mathbb{F}_q)$ using only G. While the idea to use

$Z\left(\mathrm{Adj}\left(\mathrm{Bi}(G)\right)\right)$, as in the genus 1 case, does not work, we can still recover the flat genus 2 group of which G is a quotient.

Consider any indecomposible flat genus 2 group $H_m^\flat(\mathbb{F}_q)$. Then $\mathrm{Bi}_{\mathbb{F}_q}(H_m^\flat(\mathbb{F}_q))$ is a $(2m+1)\times(2m+1)\times 2$-tensor. Slicing $\mathrm{Bi}_{\mathbb{F}_q}(H_m^\flat(\mathbb{F}_q))$ such that there are two slices $(\mathrm{Bi}_{\mathbb{F}_q}(H_m^\flat(\mathbb{F}_q))_{**1}$ and $\mathrm{Bi}_{\mathbb{F}_q}(H_m^\flat(\mathbb{F}_q))_{**2}$ as explained in the conventions below Definition 4.2.1.1), by [20] there are changes of bases in $H_m^\flat(\mathbb{F}_q)/Z(H_m^\flat(\mathbb{F}_q))$ and $Z(H_m^\flat(\mathbb{F}_q))$ such that, respectively, the two slices are

$$\left(\begin{bmatrix} 0 & I_m|0 \\ -(I_m|0)^T & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0|I_m \\ -(0|I_m)^T & 0 \end{bmatrix}\right), \tag{4.2}$$

where the blocks $I_m|0$ and $0|I_m$ are $m\times(m+1)$-blocks consisting of an identity matrix augmented by a column vector of zeros. Writing this tensor over $\mathbb{F}_p$ instead of $\mathbb{F}_{p^n}$ (as explained in §2.4), there are $2n$ slices of size $n(2m+1)\times n(2m+1)$ where any non-zero linear combination of slices has rank $2mn$. We notate this rewritten tensor as $\mathrm{Bi}_{\mathbb{F}_p}(H_m^\flat(\mathbb{F}_{p^n}))$. Explicitly, any linear combination of matrices in $\{\mathrm{Bi}_{\mathbb{F}_p}(H_m^\flat(\mathbb{F}_{p^n}))_{**k}\}_{k\in\{1,\dots,2n\}}$ is

$$M = \left[\begin{array}{cccc|ccccc} & & & & A & B & 0 & \cdots & 0 \\ & & & & 0 & A & B & \cdots & 0 \\ & & 0 & & \vdots & \ddots & \ddots & \ddots & \vdots \\ & & & & 0 & \cdots & 0 & A & B \\ \hline -A^T & 0 & \cdots & 0 & & & & & \\ -B^T & -A^T & \cdots & 0 & & & & & \\ \vdots & \ddots & \ddots & \vdots & & & 0 & & \\ 0 & \cdots & -B^T & -A^T & & & & & \\ 0 & \cdots & 0 & -B^T & & & & & \end{array}\right] \tag{4.3}$$

where $A, B \in \mathrm{GL}_n(\mathbb{F}_p)\cup\{0\}$ and $\mathrm{rank}(M) = 2nm$. Clarifying, we set the convention that the notation $\mathrm{Bi}_{\mathbb{F}_{p^n}}(H_m^\flat(\mathbb{F}_{p^n}))$ will always be the tensor $\mathrm{Bi}(H_m^\flat(\mathbb{F}_{p^n}))$ where the chosen basis gives the slices in (4.2). Also, $\mathrm{Bi}_{\mathbb{F}_p}(H_m^\flat(\mathbb{F}_{p^n}))$ will always be the tensor $\mathrm{Bi}(H_m^\flat(\mathbb{F}_{p^n}))$ where the chosen basis takes the basis of (4.2) and blows the entries up into $\mathrm{GL}_n(\mathbb{F}_p)\cup\{0\}$ using the process outlined in §2.4 to give slices as in (4.3).

Furthermore, given a quotient G of $H_m^\flat(\mathbb{F}_{p^n})$, there is a choice of basis such that for each $k \in \{1, \ldots, \dim_{\mathbb{F}_p}(G')\}$, $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ is in the $\mathbb{F}_p$-linear span of $\{\mathrm{Bi}_{\mathbb{F}_p}(H_m^\flat(\mathbb{F}_{p^n}))_{**k}\}_{k\in\{1,\ldots,2n\}}$. Considering the dimension and rank of M, the following proposition is immediate.

Proposition 4.4.1.1. *Suppose G is a quotient of the genus 2 flat group $H_m^\flat(\mathbb{F}_{p^n})$ such that G is not also a quotient of a genus 1 group. Considering any $M = \mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$, we have*

- $n = \mathrm{rows}(M) - \mathrm{rank}(M)$,
- $m = \dfrac{\mathrm{rank}(M)}{2n}$.

That is, given an indecomposible group G which is a quotient of a genus 2 flat group, a canonical flat group it is a quotient of can be recovered from $\mathrm{Bi}(G)$.

4.4.2 Writing $\mathrm{Bi}(G)$ as potential $\mathbb{F}_q$ blocks

Given G, a centrally indecomposible quotient of a flat genus 2 group by a central subgroup, we first determine what flat genus 2 group G is a quotient of, if indeed it is a quotient of a flat genus 2 group (§4.4.1). Picking any fixed representation of $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[x]/(a(x))$, we want to find a $\mathbb{F}_p$-basis of $\mathrm{Bi}(G)$ such that each slice $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ is a linear combination of slices $\sum_{\ell=1}^{2n} \alpha_\ell \mathrm{Bi}_{\mathbb{F}_p}(H_m^\flat(\mathbb{F}_{p^n}))_{**\ell}$, where each $\mathrm{Bi}_{\mathbb{F}_p}(H_m^\flat(\mathbb{F}_{p^n}))_{**\ell}$ has the structure established above (4.3). As G need only be defined over $\mathbb{F}_p$, and noting the nice block structure of $\mathrm{Bi}_{\mathbb{F}_p}(H_m^\flat(\mathbb{F}_{p^n}))$, we will next attempt to write the slices $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ similarly.

While it is certainly plausible that the following results hold for $m > 1$, the proofs are not immediate generalizations, so we now restrict to the case $m = 1$. In particular, we would like to find a $\mathbb{F}_p$-basis for $\mathrm{Bi}(G)$ such that for every k,

$$\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k} = \begin{bmatrix} 0 & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix}$$

where $A_k, B_k \in \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}$.

Proposition 4.4.2.1. *Given any* $M_\alpha, M_\beta \in \left\{ \sum_{k=1}^{2n} \gamma_k \mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))_{**k} : \gamma \in \mathbb{F}_p^{2n} \setminus 0^{2n} \right\}$, *there is a* $\mathbb{F}_p$*-change of basis in* $H_1^\flat(\mathbb{F}_{p^n})/Z(H_1^\flat(\mathbb{F}_{p^n}))$ *such that either:*

(1) $B_\alpha = B_\beta = 0$ *and* $A_\alpha = I_n$*, or*

(2) $A_\alpha = B_\beta = I_n$ *and* $B_\alpha = A_\beta = 0$.

Furthermore, letting G *be a quotient of* $H_1^\flat(\mathbb{F}_{p^n})$ *by a central subgroup, there is a change of basis in* $G/Z(G)$ *such that for each* $k \in \{1, \ldots, \dim_{\mathbb{F}_p}(G')\}$,

$$\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k} = \begin{bmatrix} 0 & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix}.$$

And if G *is not a quotient of a genus 1 group, then there is a change of basis in* G' *causing* $A_1 = B_2 = I_n$ *and* $B_1 = A_2 = 0$.

Proof. We will construct this basis change by performing only a few actions. First, we will use $M_\alpha \mapsto CM_\alpha C^T, M_\beta \mapsto CM_\alpha C^T$, where

$$C = \begin{bmatrix} C_1 & & \\ & C_2^T & \\ & & C_3^T \end{bmatrix}.$$

Then

$$CM_\alpha C^T = \left[\begin{array}{c|cc} 0 & C_1 A_\alpha C_2 & C_1 B_\alpha C_3 \\ \hline -(C_1 A_\alpha C_2)^T & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ -(C_1 B_\alpha C_3)^T & & \end{array}\right]. \tag{4.4}$$

As this transformation is always performed on both M_α and M_β, we will drop subscripts and write $M \mapsto CMC^T$. Second, we will perform the row/column operations (as explained in §2.4) of swapping, adding one to another, and subtracting one from another. Even more, as we want to preserve the fact that each block is in $\mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}$, we will actually perform them on rows/columns of

blocks rather than individual rows/columns. As such, all references to rows/columns, i.e., "column 2," refer to blocks of rows/columns, i.e., "the 2nd column of blocks."

We first force $B_\alpha = 0$. If $A_\alpha = 0$, swap rows/columns 2 and 3. Then

(1) Perform $M \mapsto CMC^T$ where $C_1 = I_n, C_2 = A_\alpha^{-1}, C_3 = I_n$. After this change of basis, $A_\alpha = I_n$.

(2) If $B_\alpha \neq 0$, perform $M \mapsto CMC^T$ where $C_1 = I_n, C_2 = I_n, C_3 = B_\alpha^{-1}$. After this change of basis, $B_\alpha \in \{0, I_n\}$.

(3) If $B_\alpha \neq 0$, subtract row/column 2 from row/column 3. Then M_α is in the desired form.

Note that these operations also affected A_β and B_β, but did not affect the block structure. If $B_\beta = 0$, we have found a basis change which satisfies the proposition. Assuming $B_\beta \neq 0$, we continue. If $A_\beta \neq 0$, next force $A_\beta = 0$ by

(1) Perform $M \mapsto CMC^T$ where $C_1 = I_n, C_2 = I_n, C_3 = B_\beta^{-1} A_\beta$. After this change of basis B_β becomes the same as A_β while M_α is unaffected.

(2) Subtract column 3 from column 2. After this change of basis, $A_\beta = 0$ and $B_\beta \neq 0$ while M_α is still in the desired form.

Finally scale B_β and force it to become I_n by performing $M \mapsto CMC^T$, where $C_1 = I_n, C_2 = I_n, C_3 = B_\beta^{-1}$. This finishes the proof of the first statement of the proposition.

Now let G be a quotient of $H_1^\flat(\mathbb{F}_{p^n})$ by a central subgroup. Recalling that there is a change of basis in $G/Z(G)$ which causes every $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ to be a linear combination $\sum_{\ell=1}^{2n} \alpha_k \mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))_{**\ell}$ and noting that no shuffles of slices were used in the argument above, the following fact is immediate. For any two $M_1, M_2 \in \{\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k} : 1 \leq k \leq \dim_{\mathbb{F}_p}(G')\}$, there is a change of basis in $G/Z(G)$ such that

$$(M_1, M_2) \in \left\{ \left(\begin{bmatrix} 0 & I_n & 0 \\ -I_n & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & I_n \\ 0 & 0 & 0 \\ -I_n & 0 & 0 \end{bmatrix} \right), \left(\begin{bmatrix} 0 & I_n & 0 \\ -I_n & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & A & 0 \\ -A^T & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \right\}.$$

Even more, we were careful in the argument above to ensure that every operation respected the block structure of every matrix in $\{\mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))_{**k} : 1 \leq k \leq 2n\}$, so we actually get that there is a change of basis in $G/Z(G)$ such that for each $k \in \{1, \ldots, \dim_{\mathbb{F}_p}(G')\}$,

$$\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k} = \begin{bmatrix} 0 & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix},$$

where $A_1 = I_n$ and $B_1 = 0$. If there exists k such that $B_k \neq 0$, there is, by the argument above, a change of basis which preserves $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**1}$ (and the block form of every other slice) which turns $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ into

$$\begin{bmatrix} 0 & 0 & I_n \\ 0 & 0 & 0 \\ -I_n & 0 & 0 \end{bmatrix}.$$

Changing basis in G' by swapping slice 2 with slice k completes the desired change of basis in the second statement of the proposition.

On the other hand, if $B_k = 0$ for all k, then G is a quotient of the genus 1 group (defined over $\mathbb{F}_{p^n}$)

$$\mathrm{Gp}\left(\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) \cong H_1(\mathbb{F}_{p^n}) \times \mathbb{F}_{p^n},$$

where $H_1(\mathbb{F}_{p^n})$ is defined in Example 2.5.2.1. □

Having proved that our desired basis change exists, we now need that the basis change can be found efficiently. After picking any basis, we first transform the basis so the first two slices $\{\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}\}_{k=1,2}$ have $A_1 = B_2 = I_n$ and $B_1 = A_2 = 0$. Then we construct and solve a system of linear equations to find a basis change which forces every slice to have the desired block form while preserving the first two slices. We believe transforming the first two slices into the desired canonical form is always possible via fairly straightforward row/column reduction (see Conjecture 4.4.2.5), so

we next focus on showing the basis change transforming the remaining slices into block form can be found via a system of linear equations.

To transform slices $k = 3, \ldots, \dim_{\mathbb{F}_p}(G')$ into the desired block form, we first calculate the stabilizer of the ordered pair $\left(\mathrm{Bi}_{\mathbb{F}_p}(G)_{**1}, \mathrm{Bi}_{\mathbb{F}_p}(G)_{**2}\right)$. Then we note that only a restricted class of basis changes can affect the block form of the remaining slices. Finally, we use the restricted basis change format to construct a system of linear equations to find an appropriate basis change.

Consider the ordered pair of matrices

$$P = \left(\mathrm{Bi}_{\mathbb{F}_p}(G)_{**1}, \mathrm{Bi}_{\mathbb{F}_p}(G)_{**2}\right) = \left(\begin{bmatrix} 0 & I_n & 0 \\ -I_n & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & I_n \\ 0 & 0 & 0 \\ -I_n & 0 & 0 \end{bmatrix} \right).$$

Proposition 4.4.2.2. *The stabilizer of P under simultaneous change of basis is*

$$\mathrm{Stab}(P) = \left\{ \begin{bmatrix} D & E & F \\ 0 & D^{-T} & 0 \\ 0 & 0 & D^{-T} \end{bmatrix} : D \in \mathrm{GL}_n(\mathbb{F}_p), E, F \in M_{n\times n}(\mathbb{F}_p), DE^T = ED^T, DF^T = FD^T \right\}.$$

Proof. Looking at P acted on by an arbitrary matrix $R \in \mathrm{GL}_{3n}(\mathbb{F}_p)$ by $P \mapsto RPR^T$, we see for the first matrix of P that

$$\begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \begin{bmatrix} 0 & I_n & 0 \\ -I_n & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix}^T =$$
$$\begin{bmatrix} D_{11}D_{12}^T - D_{12}D_{11}^T & D_{11}D_{22}^T - D_{12}D_{21}^T & D_{11}D_{32}^T - D_{12}D_{31}^T \\ D_{21}D_{12}^T - D_{22}D_{11}^T & D_{21}D_{22}^T - D_{22}D_{21}^T & D_{21}D_{32}^T - D_{22}D_{31}^T \\ D_{31}D_{12}^T - D_{32}D_{11}^T & D_{31}D_{22}^T - D_{32}D_{21}^T & D_{31}D_{32}^T - D_{32}D_{31}^T \end{bmatrix}.$$

As we want $R \in \mathrm{Stab}(P)$, we much have

$$D_{i1}D_{i2}^T = D_{i2}D_{i1}^T, \qquad i = 1, 2, 3, \tag{4.5}$$

$$D_{i1}D_{32}^T = D_{i2}D_{31}^T, \qquad i = 1, 2, 3, \tag{4.6}$$

$$D_{11}D_{22}^T = I_n + D_{12}D_{21}^T. \tag{4.7}$$

Similarly, for the other matrix in P, we have

$$\begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \begin{bmatrix} 0 & 0 & I_n \\ 0 & 0 & 0 \\ -I_n & 0 & 0 \end{bmatrix} \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix}^T =$$
$$\begin{bmatrix} D_{11}D_{13}^T - D_{13}D_{11}^T & D_{11}D_{23}^T - D_{13}D_{21}^T & D_{11}D_{33}^T - D_{31}D_{31}^T \\ D_{21}D_{13}^T - D_{23}D_{11}^T & D_{21}D_{23}^T - D_{23}D_{21}^T & D_{21}D_{33}^T - D_{23}D_{31}^T \\ D_{31}D_{13}^T - D_{33}D_{11}^T & D_{31}D_{23}^T - D_{33}D_{21}^T & D_{31}D_{33}^T - D_{33}D_{31}^T \end{bmatrix}.$$

Again we get equations, which are

$$D_{i1}D_{i3}^T = D_{i3}D_{i1}^T, \qquad i = 1, 2, 3, \tag{4.8}$$

$$D_{i1}D_{23}^T = D_{i3}D_{21}^T, \qquad i = 1, 2, 3, \tag{4.9}$$

$$D_{11}D_{33}^T = I_n + D_{13}D_{31}^T. \tag{4.10}$$

Noting that each block is itself a $n \times n$ matrix, where for example

$$D_{ij} = \begin{bmatrix} (d_{ij})_{11} & (d_{ij})_{12} & \cdots & (d_{ij})_{1n} \\ (d_{ij})_{21} & (d_{ij})_{22} & \cdots & (d_{ij})_{2n} \\ \vdots & \vdots & & \vdots \\ (d_{ij})_{n1} & (d_{ij})_{n2} & \cdots & (d_{ij})_{nn} \end{bmatrix},$$

we define the vectors

$$\overline{(d_{ij})_{k\cdot}} := \langle (d_{ij})_{k1}, (d_{ij})_{k2}, \ldots, (d_{ij})_{kn} \rangle,$$

$$\overline{(d_{ij})_{\cdot k}} := \langle (d_{ij})_{1k}, (d_{ij})_{2k}, \ldots, (d_{ij})_{nk} \rangle.$$

With this notation, we can convert our matrix equations into vector equations. In particular, we

get the equations

$$\overline{(d_{i1})_{k\cdot}} \cdot \overline{(d_{i2})_{\ell\cdot}} = \overline{(d_{i1})_{\ell\cdot}} \cdot \overline{(d_{i2})_{k\cdot}}, \qquad i = 1,2,3, \tag{4.11}$$

$$\overline{(d_{i1})_{k\cdot}} \cdot \overline{(d_{32})_{\ell\cdot}} = \overline{(d_{i2})_{k\cdot}} \cdot \overline{(d_{31})_{\ell\cdot}}, \qquad i = 1,2,3, \tag{4.12}$$

$$\overline{(d_{11})_{k\cdot}} \cdot \overline{(d_{22})_{\ell\cdot}} = \overline{(d_{12})_{k\cdot}} \cdot \overline{(d_{21})_{\ell\cdot}} + \delta_{k\ell}, \tag{4.13}$$

$$\overline{(d_{i1})_{k\cdot}} \cdot \overline{(d_{i3})_{\ell\cdot}} = \overline{(d_{i1})_{\ell\cdot}} \cdot \overline{(d_{i3})_{k\cdot}}, \qquad i = 1,2,3, \tag{4.14}$$

$$\overline{(d_{i1})_{k\cdot}} \cdot \overline{(d_{23})_{\ell\cdot}} = \overline{(d_{i3})_{k\cdot}} \cdot \overline{(d_{21})_{\ell\cdot}}, \qquad i = 1,2,3, \tag{4.15}$$

$$\overline{(d_{11})_{k\cdot}} \cdot \overline{(d_{33})_{\ell\cdot}} = \overline{(d_{13})_{k\cdot}} \cdot \overline{(d_{31})_{\ell\cdot}} + \delta_{k\ell}. \tag{4.16}$$

Notating column i by C_i, we show $D_{32} = 0$. In order to show a contradiction, consider that if $(d_{32})_{11} \neq 0$, then we can perform the column operation where we set

$$C_1 \leftarrow (d_{32})_{11}C_1 + (d_{32})_{12}C_2 + \cdots + (d_{32})_{1n}C_n - ((d_{31})_{11}C_{n+1} + (d_{31})_{12}C_{n+2} + \cdots (d_{31})_{1n}C_{2n}).$$

By Equation (4.12), every row of column 1 will be 0. This contradicts the fact that R has full rank, so we have $(d_{32})_{11} = 0$.

This same argument but setting

$$C_k \leftarrow (d_{32})_{11}C_1 + (d_{32})_{12}C_2 + \cdots + (d_{32})_{1n}C_n - ((d_{31})_{11}C_{n+1} + (d_{31})_{12}C_{n+2} + \cdots (d_{31})_{1n}C_{2n})$$

gives $(d_{32})_{1k} = 0$ for $k \in \{1, \ldots, n\}$ and $(d_{31})_{1k} = 0$ for $k \in \{n+1, \ldots, 2n\}$. To show the remaining rows of D_{32}, D_{31} are 0, we use the same argument where the coefficients are from row ℓ. That is, the column operations

$$C_k \leftarrow (d_{32})_{\ell 1}C_1 + (d_{32})_{\ell 2}C_2 + \cdots + (d_{32})_{\ell n}C_n - ((d_{31})_{\ell 1}C_{n+1} + (d_{31})_{\ell 2}C_{n+2} + \cdots (d_{31})_{\ell n}C_{2n})$$

give $(d_{32})_{\ell k} = 0$ for $k \in \{1, \ldots, n\}$ and $(d_{31})_{\ell k} = 0$ for $k \in \{n+1, \ldots, 2n\}$.

Similarly, using Equation 4.15 and the column operations

$$C_k \leftarrow (d_{23})_{\ell 1}C_1 + (d_{23})_{\ell 2}C_2 + \cdots (d_{23})_{\ell n}C_n - ((d_{21})_{\ell 1}C_{n+1} + (d_{21})_{\ell 2} + \cdots + (d_{21})_{\ell n}C_{2n}),$$

we get $D_{21} = D_{23} = 0$.

Substituting these known zero blocks into our matrix equations causes (4.7) to become $D_{11}D_{22}^T = I_n$, so $D_{22} = D_{11}^{-T}$. Similarly, (4.10) becomes $D_{11}D_{33}^T = I_n$, so $D_{33} = D_{11}^{-T}$.

Finally, the added restrictions that $D_{11}D_{12}^T$ and $D_{11}D_{13}^T$ are symmetric come from (4.5) and (4.8) where $i = 1$. Thus setting $D := D_{11}, E := D_{12}, F := D_{13}$, we have

$$\mathrm{Stab}(P) \subseteq \left\{ \begin{bmatrix} D & E & F \\ 0 & D^{-T} & 0 \\ 0 & 0 & D^{-T} \end{bmatrix} : D \in \mathrm{GL}_n(\mathbb{F}_p), E, F \in M_{n\times n}(\mathbb{F}_p), DE^T = ED^T, DF^T = FD^T \right\}.$$

It is easy to verify that any such matrix is, in fact, in $\mathrm{Stab}(P)$, so the subset is an equality. □

Looking at the action of a matrix from $\mathrm{Stab}(P)$ on an arbitrary skew-symmetric matrix, we see

$$\begin{bmatrix} D & E & F \\ 0 & D^{-T} & 0 \\ 0 & 0 & D^{-T} \end{bmatrix} \begin{bmatrix} G & A & B \\ -A^T & H & J \\ -B^T & -J^T & L \end{bmatrix} \begin{bmatrix} D^T & 0 & 0 \\ E^T & D^{-1} & 0 \\ F^T & 0 & D^{-1} \end{bmatrix} =$$

$$\begin{bmatrix} (*) & DAD^{-1} + EHD^{-1} - FJ^TD^{-1} & DBD^{-1} + EJD^{-1} + FLD^{-1} \\ (**) & D^{-T}HD^{-1} & D^{-T}JD^{-1} \\ (**) & -D^{-T}J^TD^{-1} & D^{-T}LD^{-1} \end{bmatrix},$$

where $(**)$ are the necessary entries for the matrix to be skew-symmetric and

$$(*) = DGD^T - EA^TD^T - FB^TD^T + DAE^T + EHE^T - FJ^TE^T + DBF^T + EJF^T + FLF^T.$$

Noting that the $(2,2), (2,3), (3,2), (3,3)$ blocks cannot change between zero and non-zero, if the skew-symmetric matrix is a slice of $\mathrm{Bi}_{\mathbb{F}_p}(G)$ where G is a quotient of $H_1^p(\mathbb{F}_{p^n})$ by a central subgroup, then we must already have $H = J = L = 0$. Acting on a slice $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ by an element of $\mathrm{Stab}(P)$, we have

$$\begin{bmatrix} D & E & F \\ 0 & D^{-T} & 0 \\ 0 & 0 & D^{-T} \end{bmatrix} \begin{bmatrix} G_k & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix} \begin{bmatrix} D^T & 0 & 0 \\ E^T & D^{-1} & 0 \\ F^T & 0 & D^{-1} \end{bmatrix} =$$

$$\begin{bmatrix} DG_kD^T + DA_kE^T - EA_k^TD^T + DB_kF^T - FB_k^TD^T & DA_kD^{-1} & DB_kD^{-1} \\ -D^{-T}A_k^TD^T & 0 & 0 \\ -D^{-T}B_k^TD^T & 0 & 0 \end{bmatrix},$$

We want to restrict the class of basis changes even further, so we can use the action on each G_k to form a system of equations which is linear to build a basis change which makes every $G_k = 0$ while preserving the block structure. First, observe that any basis change in $\mathrm{Stab}(P)$ can be decomposed as

$$\begin{bmatrix} D & E & F \\ 0 & D^{-T} & 0 \\ 0 & 0 & D^{-T} \end{bmatrix} = \begin{bmatrix} D & 0 & 0 \\ 0 & D^{-T} & 0 \\ 0 & 0 & D^{-T} \end{bmatrix} \begin{bmatrix} I_n & D^{-1}E & D^{-1}F \\ 0 & I_n & 0 \\ 0 & 0 & I_n \end{bmatrix}.$$

Observing that the action

$$\begin{bmatrix} D & 0 & 0 \\ 0 & D^{-T} & 0 \\ 0 & 0 & D^{-T} \end{bmatrix} \begin{bmatrix} G_k & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix} \begin{bmatrix} D^T & 0 & 0 \\ 0 & D^{-1} & 0 \\ 0 & 0 & D^{-1} \end{bmatrix} = \begin{bmatrix} DG_kD^T & DA_kD^{-1} & DB_kD^{-1} \\ -D^{-T}A_k^TD^T & 0 & 0 \\ -D^{-T}B_k^TD^T & 0 & 0 \end{bmatrix}$$

cannot affect whether or not G_k is zero, it suffices to consider basis changes of form

$$\left\{ \begin{bmatrix} I_n & E & F \\ 0 & I_n & 0 \\ 0 & 0 & I_n \end{bmatrix} : E, F \in M_{n\times n}(\mathbb{F}_p) \text{ are symmetric} \right\}.$$

Investigating this action, we see

$$\begin{bmatrix} I_n & E & F \\ 0 & I_n & 0 \\ 0 & 0 & I_n \end{bmatrix} \begin{bmatrix} G_k & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix} \begin{bmatrix} I_n & 0 & 0 \\ E^T & I_n & 0 \\ F^T & 0 & I_n \end{bmatrix} =$$
$$\begin{bmatrix} G_k - EA_k^T - FB_k^T + A_kE^T + B_kF^T & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix},$$

so setting

$$0 = G_k - EA_k^T - FB_k^T + A_kE^T + B_kF^T, \tag{4.17}$$

$$0 = E - E^T, \tag{4.18}$$

$$0 = F - F^T \tag{4.19}$$

generates a system of equations which is linear in the entries of E, F. And by Proposition 4.4.2.1, this system of linear equations will have a solution if G is, in fact, a quotient of $H_1^\flat(\mathbb{F}_{p^n})$ by a central subgroup. Even more, note that the actions by

$$\begin{bmatrix} I_n & E & F \\ 0 & I_n & 0 \\ 0 & 0 & I_n \end{bmatrix} \text{ and } \begin{bmatrix} D^T & 0 & 0 \\ 0 & D^{-1} & 0 \\ 0 & 0 & D^{-1} \end{bmatrix}$$

cannot affect whether or not A_k, B_k are full rank, so we must also have $A_k, B_k \in \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}$. This leads to the following proposition.

Proposition 4.4.2.3. *Given* $\mathrm{Bi}_{\mathbb{F}_p}(G)$ *where* $(\mathrm{Bi}_{\mathbb{F}_p}(G)_{**1}, \mathrm{Bi}_{\mathbb{F}_p}(G)_{**2}) = P$*, finding a basis change which preserves the first two slices while transforming the remaining slices to be*

$$\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k} = \begin{bmatrix} 0 & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix}$$

such that $A_k, B_k \in \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}$ *for all* $k > 3$ *can be done in* $O(n^7)$ *time.*

Proof. Using the equations (4.17), (4.18), (4.19) generates $O(n^3)$ equations in $O(n^2)$ variables. A solution to these linear equations can be found in $O(n^7)$ time by Gaussian elimination. □

Remark 4.4.2.4. In theory, finding a basis for the solution space of the equations in Proposition 4.4.2.3 has the same complexity of matrix multiplication [65, 50]. Regardless, we proceed with the slightly weaker result.

Summarizing, we have Algorithm 4.1.

Function `ConvertToBlockForm`($\mathrm{Bi}_{\mathbb{F}_p}(G)$)**:**

Input: $\mathrm{Bi}_{\mathbb{F}_p}(G)$ where any $\mathbb{F}_p$-basis has been chosen

Output: $\mathrm{Bi}_{\mathbb{F}_p}(G)$ with a new basis such that for all k,

$$\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k} = \begin{bmatrix} 0 & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix},$$

where $A_k, B_k \in \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}$ and $A_1 = B_2 = I_n, A_2 = B_1 = 0$.

(1) Pick any slice, say $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**1}$, and use straightforward row/column reduction to put it into block form where $A_1 = I_n$ and $B_1 = 0$.

(2) Find a slice such that the final n columns are not all 0. Swap slices so this slice is $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**2}$.

(3) Use row/column reduction to put $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**2}$ into block form with $A_2 = 0$ and $B_2 = I_n$ while preserving $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**1}$.

(4) Find a change of basis which forces the D_{11} block of every slice $\mathrm{Bi}_{\mathbb{F}_P}(G)_{**k}$ to be 0 by generating and solving a system of $O(n^3)$ linear equations in $O(n^2)$ variables.

Algorithm 4.1: Convert $\mathrm{Bi}_{\mathbb{F}_p}(G)$ to block form

While we have not fully proven that this $O(n^7)$-time algorithm always succeeds, there is only one small missing step, embodied in the following conjecture.

Conjecture 4.4.2.5. *Step (3) of Algorithm 4.1 always succeeds; that is, it always puts* $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**2}$ *into the desired form.*

Walking through the algorithm step by step, we note the following. Step (1) can be done by straightforward row/column operations in $O((3n)^2(2n)) = O(n^3)$ time. Step (2) can be done $O(n(3n)(2n)) = O(n^3)$ time. Note that if no such slice exists and G is a quotient of $H_1^\flat(\mathbb{F}_{p^n})$, then G is also a quotient of a genus 1 group and isomorphism of G can be determined in polynomial time by [49]. While we do not have a proof that our algorithm for step (3) always succeeds (see Conjecture 4.4.2.5), the author implemented a $O(n^4)$-time algorithm in Magma [17] which has run successfully with no exceptions on tens of millions of random examples. Finally, we have by Proposition 4.4.2.3 that step (4) can be performed in $O(n^7)$ time. (See Remark 4.4.2.4 for a possible improvement to this step.)

4.4.3 Realizing G as an explicit quotient $H_m^\flat(\mathbb{F}_q)/N$

Given a centrally indecomposible group G which is a quotient of a flat genus 2 group, we first determine what flat genus 2 group it is a quotient of (see §4.4.1). Since there is a basis for $\mathrm{Bi}(G)$ such that each $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ is a linear combination of slices in $\{\mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))_{**k}\}_{k=1,\ldots,2n}$, we note that each slice $\mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))_{**k}$ has a nice block structure, so we find a basis where each slice $\mathrm{Bi}_{\mathbb{F}_p}(G)$ has the same nice block structure (see §4.4.2). Now fixing any representation of $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[x]/(a(x))$, we find a basis so each slice $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ is a linear combination of the canonical slices $\{\mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))_{**k}\}$.

Given $\mathrm{Bi}_{\mathbb{F}_p}(G)$ written in block form as

$$\left\{ \begin{bmatrix} 0 & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix} : A_k, B_k \in \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}, 1 \le k \le \dim_{\mathbb{F}_p}(G') \right\},$$

we now transform the $\{A_k\} \cup \{B_k\}$ to form an embedding $\mathbb{F}_{p^n} \hookrightarrow \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}$. After performing a similar transformation on $\mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))$, we need only to determine a field isomorphism to write

each $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ as an explicit linear combination $\sum_{\ell=1}^{2n} \alpha_\ell \mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n})_{**\ell})$ at which point writing G as a quotient of $H_1^\flat(\mathbb{F}_{p^n})$ by an explicit central subgroup N is trivial.

Consider the set of nonzero blocks from the slices $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$. Let

$$\mathcal{B}^G = \{B : B \in \{A_k, B_k\} \text{ for some } \mathrm{Bi}_{\mathbb{F}_p}(G)_{**k} \text{ and } B \neq 0\}.$$

Fix any indexing $\mathcal{B}^G = \{B_i^G\}_{i=1,\dots,\ell}$, and define the tensor $T_{\mathcal{B}^G}$ by letting $\mathcal{B}^G_{**k} = B_k^G$. That is, form $T_{\mathcal{B}^G}$ by stacking blocks in $\mathcal{B}^G$ to be the slices of $T_{\mathcal{B}^G}$ over $\mathbb{F}_p$. Define $\mathrm{Adj}(\mathcal{B}^G)$ to be $\mathrm{Adj}(T_{\mathcal{B}^G})$ (see Definition 4.4.1), where the basis of $T_{\mathcal{B}^G}$ has been chosen such that B_k^G is the kth slice of the tensor. As there is a choice of basis of $T_{\mathcal{B}^G}$ such that each slice of $T_{\mathcal{B}^G}$ is a linear combination of slices from $T_{\mathcal{B}^{H_1^\flat(\mathbb{F}_{p^n})}}$, we have $\mathrm{Adj}(\mathcal{B}^{H_1^\flat(\mathbb{F}_{p^n})}) \cong R \subseteq \mathrm{Adj}(\mathcal{B}^G)$. Consider $(\alpha_u, \alpha_v^T) \in R$. By definition of the adjoint, $\alpha_u B_i^G = B_i^G \alpha_v^T$; so in particular, $(B_1^G)^{-1}\alpha_u B_1^G = \alpha_v^T$, and

$$\alpha_u B_i^G = B_i^G \alpha_v^T \tag{4.20}$$

$$\alpha_u B_i^G (B_1^G)^{-1} = B_i^G \alpha_v^T (B_1^G)^{-1} \tag{4.21}$$

$$\alpha_u (B_i^G (B_1^G)^{-1}) = B_i^G ((B_1^G)^{-1} \alpha_u B_1^G) B_1^{G-1} \tag{4.22}$$

$$\alpha_u (B_i^G (B_1^G)^{-1}) = (B_i^G (B_1^G)^{-1}) \alpha_u. \tag{4.23}$$

That is, $B_i^G (B_1^G)^{-1}$ commutes with every $\alpha_u \in \mathrm{Adj}(\mathcal{B}^{H_1^\flat(\mathbb{F}_{p^n})})|_U$, which leads us to the following proposition.

Proposition 4.4.3.1. *Suppose G is a quotient of the genus 2 flat group $H := H_1^\flat(\mathbb{F}_{p^n})$ by a central subgroup and $\mathrm{Bi}_{\mathbb{F}_p}(G)$ is written in block form as*

$$\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k} = \left\{ \begin{bmatrix} 0 & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix} : A_k, B_k \in \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}, 1 \leq k \leq \dim_{\mathbb{F}_p}(G') \right\}.$$

Then $\{B_i^H (B_1^H)^{-1} : B_i^H, B_1^H \in \mathcal{B}^H\}$ generates an embedding of $\mathbb{F}_{p^n}$ in $\mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}$ and $\{B_i^G (B_1^G)^{-1} : B_i^G, B_1^G \in \mathcal{B}^G\}$ generates a subfield of $\mathbb{F}_{p^n}$.

Before proving this proposition, we need the following lemma; we thank Nathaniel Thiem for its proof.

Lemma 4.4.3.2. *Embed $\mathbb{F}_{p^n}^\times$ into $GL := \mathrm{GL}_n(\mathbb{F}_p)$. Let F be the image of the embedding. Then the centralizer $C_{GL}(F) = F$.*

Proof (N. Thiem). Let $\psi : \mathbb{F}_{p^n}^\times \hookrightarrow \mathrm{GL}_n(\mathbb{F}_p)$ be an embedding of $\mathbb{F}_{p^n}$ into $GL := \mathrm{GL}_n(\mathbb{F}_p)$, and let $F := \psi(\mathbb{F}_{p^n})$ be the image of the embedding.

This result follows by [54, II.1.6, IV.2.5–IV.2.7]. To show why these results apply in our situation, we adopt the notation of [54] for this proof only. We have

$$|C_{GL}(F)| = \prod_{f \in \Phi} a_{\mu(f)}(q_f).$$

In our context, $|\Phi| = 1$ as the characteristic polynomial of $\psi(x)$ is irreducible (it is the irreducible polynomial used to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/(\psi(x))$). Let $\lambda := \mu(f)$ for the unique $f \in \Phi$. Then

$$|C_{GL}(L)| = a_\lambda(p^n) = (p^n)^{|\lambda|+2n(\lambda)} \prod_{i \geq 1} \varphi_{m_{i(\lambda)}}(p^{-n}). \tag{4.24}$$

Again since f is irreducible, the partition λ is trivial; thus $|\lambda| = 1$, $n(\lambda) = \sum_{i=1}^{1}(i-1)\lambda_i = (1-1)\lambda_1 = 0$, and $\prod_{i \geq 1} \varphi_{m_{i(\lambda)}}(p^{-n}) = \varphi_1(p^{-n}) = (1 - p^{-n})$. Substituting these values back into (4.24), we get

$$|C_{GL}(L)| = (p^n)^{1+0}(1 - p^{-n}) = p^n - 1.$$

Since $F \subseteq C_{GL}(F)$, we have, by cardinality considerations only, that $F = C_{GL}(F)$. □

Now we are ready to prove Proposition 4.4.3.1.

Proof. First we show $\{B_i^H (B_1^H)^{-1} : B_i^H, B_1^H \in \mathcal{B}^H\}$ generates an embedding of $\mathbb{F}_{p^n}^\times$ in $\mathrm{GL}_n(\mathbb{F}_p)$. Note that there is a shuffle of the canonical slices of $T_{\mathcal{B}^H}$ (over $\mathbb{F}_p$) such that $T_{\mathcal{B}^H}$ forms two copies of the block of $1 \in \mathbb{F}_{p^n}$ embedded into $\mathbb{F}_p^{n \times n \times n}$ via the procedure in §2.4.4. Thus $\mathbb{F}_{p^n}^\times \hookrightarrow \mathrm{Adj}(\mathcal{B}^H)|_U \subseteq \mathrm{GL}_n(\mathbb{F}_p)$. By a calculation identical to that of (4.20)–(4.23) but with blocks from $\mathcal{B}^H$, we have $\{B_i^H (B_1^H)^{-1}\}$ is contained in the centralizer of $\mathrm{Adj}(\mathcal{B}^H)|_U$. By Lemma 4.4.3.2, this centralizer can have cardinality no larger than $p^n - 1$. Note also that the canonical slices of $\mathcal{B}^H$ contain n linearly independent $n \times n$ matrices; let $\{B_1^H, \ldots, B_n^H\}$ be the linearly independent slices. As $B_1^H \in \mathrm{GL}_n(\mathbb{F}_p)$, we also have $\{B_1^H (B_1^H)^{-1}, \ldots, B_n^H (B_1^H)^{-1}\}$ are still linearly independent. As

nonzero linear combinations of n linearly independent matrices generate $p^n - 1$ matrices, the size of the centralizer of $\mathrm{Adj}(\mathcal{B}^H)|_U$ in $\mathrm{GL}_n(\mathbb{F}_p)$ is exactly $p^n - 1$. Thus

$$\langle \mathcal{B}^H (B_1^H)^{-1} \rangle = \langle B_1^H (B_1^H)^{-1}, \ldots, B_n^H (B_1^H)^{-1} \rangle \cong \mathbb{F}_{p^n} \hookrightarrow \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}.$$

Turning now to $\langle \mathcal{B}^G (B_1^G)^{-1} \rangle$, we have by the calculation in (4.20)–(4.23) that $B_i^G (B_1^G)^{-1}$ commutes with every $\alpha_u \in \mathrm{Adj}(\mathcal{B}^H)|_U$ for every i; that is, $\{B_i^G (B_1^G)^{-1}\}$ is contained in the centralizer of $\mathrm{Adj}(\mathcal{B}^H)|_U$ in $\mathrm{GL}_n(\mathbb{F}_p)$ and $\{\mathcal{B}^G (B_1^G)^{-1}\}$ is contained in an embedding of $\mathbb{F}_{p^n}^\times$ in $\mathrm{GL}_n(\mathbb{F}_p)$. Thus, generating $\langle \mathcal{B}^G (B_1^G)^{-1} \rangle \subseteq \mathrm{GL}_n(\mathbb{F}_p) \cup \{0\}$ as a subring (so allowing both linear combinations and powers) generates a subfield of the embedding of $\mathbb{F}_{p^n}$. □

Note that we already have $I_n \in \mathcal{B}^G$, so there is a shuffle of $\mathcal{B}$ such that $I_n = B_1^G$. Thus $\mathcal{B}^G$ already generates a subfield of $\mathbb{F}_{p^n}$. Also, $\langle \mathcal{B}^H (B_1^H)^{-1} \rangle \cong \mathbb{F}_{p^n}$. Solve field isomorphism to embed $\varphi : \langle \mathcal{B}^G \rangle \hookrightarrow \langle \mathcal{B}^H (B_1^H)^{-1} \rangle$, where $\varphi \in \mathrm{GL}_n(\mathbb{F}_p)$. Such an isomorphism can be computed in polynomial time [48]. Noting that

$$\begin{bmatrix} \varphi^{-1} & 0 & 0 \\ 0 & (A_1^H)^T & 0 \\ 0 & 0 & (A_1^H)^T \end{bmatrix} \begin{bmatrix} 0 & A_k & B_k \\ -A_k^T & 0 & 0 \\ -B_k^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \varphi & 0 & 0 \\ 0 & A_1^{-1} & 0 \\ 0 & 0 & A_1^{-1} \end{bmatrix} =$$
$$\begin{bmatrix} 0 & \varphi^{-1}(A_k) \cdot A_1^H & \varphi^{-1}(B_k) \cdot A_1^H \\ -(\varphi^{-1}(A_k) \cdot A_1^H)^T & 0 & 0 \\ -(\varphi^{-1}(B_k) \cdot A_1^H)^T & 0 & 0 \end{bmatrix},$$

we can efficiently transform our already chosen basis of $\mathrm{Bi}(G)$ so each slice $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ is a linear combination of slices from $\{\mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))_{**k}\}_{k=1,\ldots,2n}$. Consider the $1 \times 1 \times 1$-tensor T over $\mathbb{F}_{p^n}$ with entry $T_{111} = 1$, and recall from §2.4 that after writing T over $\mathbb{F}_p$ using the established convention that each slice has a 1 in a unique location of the top row. Thus it is trivial to write each slice $\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}$ as $\sum_{\ell=1}^{2n} \alpha_\ell \mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))$. Extend $\{\mathrm{Bi}_{\mathbb{F}_p}(G)_{**k}\}_{k=1,\ldots,\dim_{\mathbb{F}_p}(G')}$ by sequentially adding linearly independent matrices from $\{\mathrm{Bi}_{\mathbb{F}_p}(H_1^\flat(\mathbb{F}_{p^n}))_{**k}\}_{k=1,\ldots,2n}$ until there are $2n$ matrices. These added matrices form a basis in $Z(H_1^\flat(\mathbb{F}_{p^n}))$ for N where $G \cong H_1^\flat(\mathbb{F}_{p^n})/N$.

4.4.4 Determining if $H_m^\flat(\mathbb{F}_q)/N_1 \cong H_m^\flat(\mathbb{F}_q)/N_2$

Given two groups G_1, G_2 which are quotients of flat genus 2 groups, we first determine a canonical flat genus 2 group of which it is a quotient (§4.4.1). Then if the G_i are quotients of $H_1^\flat(\mathbb{F}_q)$, we write the G_i explicitly as $H_1^\flat(\mathbb{F}_q)/N_i$ (§4.4.3). Given two groups $G_1 \cong H_1^\flat(\mathbb{F}_q)/N_1$, $G_2 \cong H_1^\flat(\mathbb{F}_q)/N_2$ written as explicit quotients by central subgroups, it is possible that $G_1 \cong G_2$ but $N_1 \neq N_2$, so our isomorphism test is not yet complete. As is the case in [49] for groups of genus 1, we conjecture the following.

Conjecture 4.4.4.1. *Let $G_1 \cong H_1^\flat(\mathbb{F}_q)/N_1$ and $G_2 \cong H_1^\flat(\mathbb{F}_q)/N_2$ where the N_i are central subgroups. Then $G_1 \cong G_2$ if and only if there is $\varphi \in \mathrm{Aut}(H_1^\flat(\mathbb{F}_q))$ such that $\varphi(N_1) = N_2$.*

Assuming Conjecture 4.4.4.1, we then need only to find $\mathrm{Aut}(H_1^\flat(\mathbb{F}_{p^n}))$ to finish our isomorphism test for quotients of $H_1^\flat(\mathbb{F}_{p^n})$. To this end, we have the following.

Proposition 4.4.4.2 ([12])**.** $\mathrm{Aut}\left(H_1^\flat(\mathbb{F}_{p^n})\right) \cong \mathrm{Hom}_{\mathbb{F}_p}\left(\mathbb{F}_{p^n}^3, \mathbb{F}_{p^n}^2\right) \rtimes \Psi\mathrm{Isom}\left(\mathrm{Bi}(H_1^\flat(\mathbb{F}_{p^n}))\right)$*, where for $\mu \in \mathrm{Hom}_{\mathbb{F}_p}\left(\mathbb{F}_{p^n}^3, \mathbb{F}_{p^n}^2\right)$ and $(f, \tilde{f}) \in \Psi\mathrm{Isom}\left(\mathrm{Bi}(H_1^\flat(\mathbb{F}_{p^n}))\right)$, $\mu^{(f,\tilde{f})} = f^{-1}\mu\tilde{f}$.*

Conjecture 4.4.4.3. $\Psi\mathrm{Isom}(\mathrm{Bi}(H_1^\flat(\mathbb{F}_{p^n}))) = (\mathrm{Sp}(2, \mathbb{F}_{p^n}) \rtimes \mathrm{GL}_2(\mathbb{F}_p)) \rtimes \mathrm{Gal}(\mathbb{F}_{p^n}/\mathbb{F}_p)$*, so in light of Proposition 4.4.4.2,*

$$\mathrm{Aut}(F_1^\flat(\mathbb{F}_{p^n})) = \mathrm{Hom}_{\mathbb{F}_p}(\mathbb{F}_{p^n}^3, \mathbb{F}_{p^n}^2) \rtimes ((\mathrm{Sp}(2, \mathbb{F}_{p^n}) \rtimes \mathrm{GL}_2(\mathbb{F}_p)) \rtimes \mathrm{Gal}(\mathbb{F}_{p^n}/\mathbb{F}_p)).$$

Regardless of the validity of Conjecture 4.4.4.3, generators for $\Psi\mathrm{Isom}(\mathrm{Bi}(\mathbb{F}_{p^n}))$ can be found in polynomial time.

Theorem 4.4.4.4 ([20, Theorem 6.5])**.** *There is a deterministic algorithm that, given an alternating $\mathbb{F}_q$-bilinear map $b : \mathbb{F}_q^d \times \mathbb{F}_q^d \rightarrowtail \mathbb{F}_q^2$ of genus 2, constructs generators for $\Psi\mathrm{Isom}(b)$. The algorithm is polynomial time if either (1) q is bounded or (2) the number of pairwise pseudo-isometric indecomposible summands of the input of the summands of the input bilinear maps is bounded.*

Hence, we have by combining Theorem 4.4.4.4 and Proposition 4.4.4.2 that there is a deterministic algorithm which constructs generators for $\mathrm{Aut}(H_1^\flat(\mathbb{F}_{P^n}))$. Thus, assuming Conjec-

ture 4.4.4.1, we can test if $H_1^\flat(\mathbb{F}_{P^n})/N_1 \cong H_1^\flat(\mathbb{F}_{P^n})/N_2$ in polynomial time. Combining with Theorem E, we get the following conjecture.

Conjecture 4.4.4.5 (Theorem F)**.** *Given two groups G_1, G_2 which are quotients of the centrally indecomposible genus 2 group $H_1^\flat(\mathbb{F}_q)$, the polynomial time algorithm outlined in this chapter to determine whether $G_1 \cong G_2$ always succeeds.*

Recalling the missing pieces to this conjecture, we remind the reader that the only missing pieces to this conjecture are Conjectures 4.4.2.5 and 4.4.4.1. Conjecture 4.4.2.5 seems to be true based on tens of millions of random examples. Furthermore, we have found no evidence showing Conjecture 4.4.4.1 does not generalize to the more general case of all indecomposible genus 2 groups via a slight generalization of the proof of the genus 1 case in [49].

4.5 Future work

In addition to the conjectures officially stated, we leave a few problems unaddressed. As mentioned before Proposition 4.4.2.1, generalizing the arguments to $H_m^\flat(\mathbb{F}_q)$ for $m > 1$ is not immediate, so we leave the $m > 1$ case as an open problem. It seems to be the case that a partial generalization of Proposition 4.4.2.1 does hold; in particular, there seems to be a basis change such that any two linear combinations $M_\alpha, M_\beta \in \left\{\sum_{k=1}^{2n} \gamma_k \mathrm{Bi}_{\mathbb{F}_p}(H_m^\flat(\mathbb{F}_{p^n}))_{**k} : \gamma \in \mathbb{F}_p^{2n} \setminus 0^{2n}\right\}$ can be written such that $(A_\alpha, B_\alpha) = (I_n, 0)$ and $(A_\beta, B_\beta) \in \{(0, I_n), (A_\beta, 0)\}$ (see (4.3)). However, considering any other M_δ, it is unclear if there is a basis such that M_α, M_β are in the desired form and A_δ is well-defined, i.e., if each of the blocks in the position of A_δ of M_δ is any fixed block.

Continuing the possible generalization, we conjecture the following generalization of Proposition 4.4.2.2. Let

$$L = \begin{bmatrix} I_n & 0 & \cdots & 0 \\ 0 & I_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_n \end{bmatrix}, P = \left\{ \begin{bmatrix} \begin{bmatrix} 0_{nm\times nm} & L & 0_{nm\times n} \\ -L^T & 0_{nm\times nm} & 0_{nm\times n} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} 0_{nm\times nm} & 0_{nm\times n} & L \\ -L^T & 0_{nm\times n} & 0_{nm\times nm} \end{bmatrix} \right\}.$$

Then we conjecture that

$$\mathrm{Stab}(P) = \left\{ \begin{bmatrix} D & & & E_1 & E_2 & \cdots & E_{m+1} \\ & D & & E_2 & E_3 & \cdots & E_{m+2} \\ & & \ddots & \vdots & \iddots & \iddots & \vdots \\ & & & D & E_m & E_{m+1} & \cdots & E_{2m} \\ & & & & D^{-T} & & & \\ & & & & & D^{-T} & & \\ & & & & & & \ddots & \\ & & & & & & & D^{-T} \end{bmatrix} : \begin{array}{c} D \in \mathrm{GL}_n(\mathbb{F}_p), E_i \in M_{n\times n}(\mathbb{F}_p), \\ AE_i^T = E_i A^T \end{array} \right\}.$$

The proof in the case of $m = 2$ is straightforward generalization of the $m = 1$ case; however, it again requires writing the formulas in coordinates. For the case of $m > 2$, it is easy to show these matrices are contained in $\mathrm{Stab}(P)$, but whether or not this containment is strict is not obvious.

Beyond that, the case of quotients of sloped genus 2 groups by central subgroups was left unaddressed. In the sloped case, even the question of whether there exists a nice block form into which any two tensor slices can be put is unclear.

Even given algorithms to solve the cases of quotients of sloped or flat groups by central subgroups, the question of quotients of general genus 2 groups also needs to be addressed. We hope the general case (no restriction to centrally indecomposible groups) follows from the ideas developed in [20, Section 6], but we leave this problem open as well.

Bibliography

[1] Sergei I. Adian. Algorithmic unsolvability of problems of recognition of certain properties of groups. (Russian). Doklady Akademii Nauk SSSR, 103:533–535, 1955.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. Annals of Mathematics, 2:781–793, 2002.

[3] Vikraman Arvind and Jacobo Torán. Solvable group isomorphism is (almost) in NP ∩ coNP. ACM Transactions on Computation Theory, 2(2):1–22, 2011.

[4] László Babai. Trading group theory for randomness. In 17th Annual ACM Symposium on Theory of Computing, pages 421–429. ACM, 1985.

[5] László Babai. Automorphism groups, isomorphism, reconstruction. In Ronald L. Graham, Martin Grötschel, and László Lovász, editors, Handbook of Combinatorics (Vol. 2), pages 1447–1540. MIT Press, Cambridge, MA, USA, 1995.

[6] László Babai. Graph isomophism in quasipolynomial time. arXiv:1512.03547 [cs.DS], 2015.

[7] László Babai, Paolo Codenotti, Joshua A. Grochow, and Youming Qiao. Code equivalence and group isomorphism. In Proceedings of the Twenty-Second Annual ACM–SIAM Symposium on Discrete Algorithms (SODA11), pages 1395–1408, Philadelphia, PA, 2011. SIAM.

[8] László Babai, Paolo Codenotti, and Youming Qiao. Polynomial-time isomorphism test for groups with no abelian normal subgroups. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, Automata, Languages, and Programming, ICALP, pages 51–62, Berlin, Heidelberg, 2012. Springer.

[9] László Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. Journal of Computer and System Sciences, 36(2):254–276, 1988.

[10] László Babai and Youming Qiao. Polynomial-time isomorphism test for groups with abelian Sylow towers. In Christoph Dürr and Thomas Wilke, editors, 29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012), volume 14 of Leibniz International Proceedings in Informatics (LIPIcs), pages 453–464, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[11] László. Babai and Endre Szemerédi. On the complexity of matrix group problems I. In Proceedings of the 25th Annual Symposium on Foundations of Computer Science, SFCS '84, pages 229–240, Washington, DC, 1984. IEEE Computer Society.

[12] Reinhold Baer. Groups with abelian central quotient group. Transactions of the American Mathematical Society, 44(3):357–386, 1938.

[13] Genrich Belitskii, Ruvim Lipyanski, and Vladimir V. Sergeichuk. Problems of classifying associative or Lie algebras and triples of symmetric or skew-symmetric matrices are wild. Linear Algebra and Its Applications, 407:249–262, 2005.

[14] Jon Louis Bentley and Andrew Chi-Chih Yao. An almost optimal algorithm for unbounded searching. Information Processing Letters, 5(3):82–87, 1976.

[15] Simon R. Blackburn, Peter M. Neumann, and Geetha Venkataraman. Enumeration of Finite Groups. Cambridge Tracts in Mathematics. Cambridge University Press, 2007.

[16] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? Information Processing Letters, 25(2):127–132, 1987.

[17] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. Journal of Symbolic Computation, 24(3-4):235–265, 1997.

[18] Mike Boyle. Open problems in symbolic dynamics. Contemporary Mathematics, 469:69–118, 2008.

[19] Mike Boyle. Personal communication, 2018.

[20] Peter Brooksbank, Joshua Maglione, and James B. Wilson. A fast isomorphism test for groups whose Lie algebra has genus 2. Journal of Algebra, 473:545–590, 2017.

[21] Heiko Dietrich and James B. Wilson. Polynomial-time isomorphism testing of groups of most finite orders. arXiv:1806.08872 [math.GR], 2018.

[22] Rafael M. Frongillo. Optimal state amalgamation is NP-hard. Ergodic Theory and Dynamical Systems, 39(7):1857–1869, 2019.

[23] Hana Galperin and Avi Wigderson. Succinct representations of graphs. Information and Control, 56(3):183–198, 1983.

[24] The GAP Group. GAP—Groups, Algorithms, and Programming, Version 4.10.2, 2019. (`https://www.gap-system.org`).

[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. Journal of the ACM, 38(3):690–728, 1991.

[26] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal on Computing, 18(1):186–208, 1989.

[27] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In 18th Annual ACM Symposium on Theory of Computing, pages 56–68. ACM, 1986.

[28] Joshua A. Grochow and Youming Qiao. Polynomial-time isomorphism test of groups that are tame extensions. In 26th International Symposium on Algorithms and Computation (ISAAC), Springer Lecture Notes in Computer Science 9472, pages 578–589, 2015.

[29] Joshua A. Grochow and Youming Qiao. Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions. arXiv:1907.00309 [cs.CC], 2019.

[30] Gustav A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. Mathematical Systems Theory, 3(4):320–375, 1969.

[31] Harald Andrés Helfgott, Jitendra Bajpai, and Daniele Dona. Graph isomorphisms in quasi-polynomial time. arXiv:1710.04574 [math.GR], 2017.

[32] Derek Holt, Bettina Eick, and Eamonn O'Brien. Handbook of Computational Group Theory. Chapman & Hall, Boca Raton, 2005.

[33] Shui-Hung Hou. Classroom note: A simple proof of the Leverrier–Fadeev characteristic polynomial algorithm. SIAM Review, 40(3):706–709, 1998.

[34] David J. A. Howden. Computing Automorphism Groups and Isomorphism Testing in Finite Groups. PhD thesis, The University of Warwick, 2012.

[35] Costas S. Iliopoulos. Computing in general abelian groups is hard. Theoretical Computer Science, 41:81–93, 1985.

[36] I. Martin Isaacs. Finite Group Theory. American Mathematical Society, 2008.

[37] William M. Kantor and Eugene M. Luks. Computing in quotient groups. In Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing, STOC '90, pages 524–534, New York, NY, USA, 1990. ACM.

[38] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, Complexity of Computer Computations, The IBM Research Symposia Series, pages 85–103, Boston, MA, 1972. Springer.

[39] Telikepalli Kavitha. Linear time algorithms for abelian group isomorphism and related problems. Journal of Computer and System Sciences, 73(6):986–996, 2007.

[40] Bruce Kitchens. Symbolic Dynamics: One-sided, Two-sided and Countable State Markov Shifts. Springer, 1998.

[41] Morris S. Knebelman. Classification of Lie algebras. Annals of Mathematics, 36(1):46–56, 1935.

[42] Johannes Köbler, Uwe Schöning, and Jacobo Torán. The Graph Isomorphism Problem: Its Structural Complexity. Birkhauser Verlag, Basel, Switzerland, Switzerland, 1993.

[43] Richard Ladner. On the structure of polynomial time reducibility. Journal of the ACM, 22(1):155–171, 1975.

[44] François Le Gall. Powers of tensors and fast matrix multiplication. In Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM.

[45] François Le Gall. Efficient isomorphism testing for a class of group extensions. In Susanne Albers and Jean-Yves Marion, editors, 26th International Symposium on Theoretical Aspects of Computer Science, volume 3 of Leibniz International Proceedings in Informatics (LIPIcs), pages 625–636, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[46] Urbain Le Verrier. Sur les variations séculaires des éléments des orbites pour les sept planétes principales. Journal de Mathématiques, 5:220–254, 1840.

[47] Charles R. Leedham-Green and Susan McKay. The Structure of Groups of Prime Power Order. London Mathematical Society monographs. Oxford University Press, 2002.

[48] Hendrik W. Lenstra, Jr. Finding isomorphisms between finite fields. Mathematics of Computation, 56(193):329–347, 1991.

[49] Mark Lewis and James B. Wilson. Isomorphism in expanding families of indistinguishable groups. Groups, Complexity, Cryptology, 4:73–110, 2012.

[50] Thomas Lickteig. The computational complexity of division in quadratic extension fields. SIAM Journal on Computing, 16(2):278–311, 1987.

[51] Douglas Lind and Brian Marcus. An Introduction to Symbolic Dynamics and Coding. Cambridge University Press, 1999.

[52] Eugene M. Luks. Computing in solvable matrix groups. In Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, pages 111–120, 1992.

[53] Eugene M. Luks. Permutation groups and polynomial-time computation. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 11, 1993.

[54] Ian G. MacDonald. Symmetric Functions and Hall Polynomials. Oxford mathmatical monographs. Oxford University Press, second edition, 1995.

[55] Charles F. Miller. Decision problems for groups—survey and reflections. In Gilbert Baumslag and Charles F. Miller, editors, Algorithms and Classification in Combinatorial Group Theory, pages 1–59, New York, NY, 1992. Springer New York.

[56] Gary L. Miller. On the $n^{\log n}$ isomorphism technique (a preliminary report). In Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, STOC '78, pages 51–58, New York, NY, USA, 1978. ACM.

[57] Gary L. Miller. Graph isomorphism, general remarks. Journal of Computer and System Sciences, 18(2):128–142, 1979.

[58] Peter Bro Miltersen and Vinodchandran N. Variyam. Derandomizing Arthur-Merlin games using hitting sets. Computational Complexity, 14(3):256–279, 2005.

[59] Christos H. Papadimitriou and Mihalis Yannakakis. A note on succinct representations of graphs. Information and Control, 71:181–185, 1985.

[60] Youming Qiao, Jayalal Sarma M.N., and Bangsheng Tang. On isomorphism testing of groups with normal hall subgroups. In Thomas Schwentick and Christoph Dürr, editors, 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011), volume 9 of Leibniz International Proceedings in Informatics (LIPIcs), pages 567–578, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[61] Michael O. Rabin. Recursive unsolvability of group theoretic problems. Annals of Mathematics, 67(1):172–194, 1958.

[62] David J. Rosenbaum. Bidirectional collision detection and faster deterministic isomorphism testing. arXiv:1304.3935 [cs.DS], 2013.

[63] David J. Rosenbaum. Breaking the $n^{\log n}$ barrier for solvable-group isomorphism. In Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1054–1073, 2013.

[64] Carla Savage. An $O(n^2)$ algorithm for abelian group isomorphism. Technical report, North Carolina State University, Raleigh, NC, 1980.

[65] Arnold Schonhage. Equations solving in terms of computational complexity. In Proceedings of the International Congress of Mathematics, pages 131–153, 1986.

[66] Tyler Schrock and Rafael Frongillo. Computational complexity of k-block conjugacy. arXiv:1909.02627 [math.DS], 2019.

[67] Ákos Seress. Permutation Group Algorithms. Cambridge Tracts in Mathematics. Cambridge University Press, 2003.

[68] Vladimir V. Sergeĭčuk. The classification of metabelian p-groups (Russian). In Matrix Problems, pages 150–160. Akademiia Nauk Ukrainy SSR Institut Matematiki, Kiev, 1977.

[69] Michael Sipser. Introduction to the Theory of Computation. Cengage Learning, 2012.

[70] Maciej M. Sysło. The subgraph isomorphism problem for outerplanar graphs. Theoretical Computer Science, 17(1):91–97, 1982.

[71] Robert Tarjan. Depth-first search and linear graph algorithms. SIAM Journal on Computing, 1(2):146–160, 1972.

[72] Narayan Vikas. An $O(n)$ algorithm for abelian p-group isomorphism and an $O(n \log n)$ algorithm for abelian group isomorphism. Journal of Computer and System Sciences, 53(1):1–9, 1996.

[73] Fabian Wagner. On the complexity of group isomorphism. Technical report TR11-052, Electronic Colloquium on Computational Complexity (ECCC), 2011.

[74] Robert F. Williams. Classification of subshifts of finite type. Annals of Mathematics, 98(1):120–153, 1973.

[75] James B. Wilson. Finding central decompositions of p-groups. Journal of Group Theory, 12:813–830, 2008.

[76] James B. Wilson. Decomposing p-groups via Jordan algebras. Journal of Algebra, 322(8):2642–2679, 2009.

[77] James B. Wilson. Existence, algorithms, and asymptotics of direct product decompositions, I. Groups Complexity Cryptology, 4:33–72, 2012.

[78] James B. Wilson. 2014 conference on *Groups, Computation, and Geometry* at Colorado State University, co-organized by Peter Brooksbank, Alexander Hulpke, Tim Penttila, James B. Wilson, and William M. Kantor. Personal communication, 2014.