THE UNIVERSITY OF CHICAGO

LIST-DECODING HOMOMORPHISM CODES

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF MATHEMATICS
AND
DEPARTMENT OF COMPUTER SCIENCE

BY
ANGELA WUU

CHICAGO, ILLINOIS
JUNE 2018

To those who dragged me up

when I fell down

To those who believed in me

when I didn't myself

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Foremost, I would like to thank my advisor Laci Babai for many hours of support, generosity with ideas, a firehose of advice, and endless enthusiasm in teaching me how write math.

Thank you to Sasha Razborov and Madhur Tulsiani for being on my committee.

Thank you to Madhu Sudan for being the one to introduce us to the subject of list-decoding homomorphism codes.

Thank you to Tim Black for discussions and fun times.

Thank you to Gene Luks for telling me about his unpublished work explained in Section 2.4.5.

It takes a village, and I am grateful for mine.

ABSTRACT

The codewords of the *homomorphism code* $\mathrm{aHom}(G, H)$ are the affine homomorphisms between two finite groups, G and H, generalizing Hadamard codes. Following the work of Goldreich–Levin (1989), Grigorescu et al. (2006), Dinur et al. (2008), and Guo and Sudan (2014), who demonstrated local list-decodability up to minimum distance of homomorphism codes for expanding classes of groups (Boolean, abelian, nilpotent). We further expand the range of groups with this property. In particular, for the first time, we do not require either G or H to be solvable. Specifically, we demonstrate a $\mathrm{poly}(1/\varepsilon)$ bound on the list size, i.e., the number of codewords within distance (mindist-eps) from any received word, when G is an alternating group, and H **is an arbitrary (finite or infinite) group**. We conjecture that a similar bound holds for all finite simple groups as domains; the alternating groups serve as the first test case.

We also have an analogous result for abelian domain, not included in the dissertation.

Our main result is efficient local list-decoding for the permutation representations of alternating groups (i. e., when the codomain is a symmetric group S_m) under the restriction that the domain $G = A_n$ is paired with codomain $H = S_m$ satisfying $m < 2^{n-1}/\sqrt{n}$.

The limitations on the codomain in the latter case reflect a gap between uniquely identifying a homomorphism in $\mathrm{aHom}(A_n, H)$ and determining the homomorphism on generators of the whole group. This phenomenon is new and is sure to appear again for other more general classes of domains. Bridging this gap requires solving the **Homomorphism Extension Problem** (HomExt): given a partial map $\gamma : G \rightharpoonup H$ (the domain of γ is a subset of G) decide whether or not there exists a homomorphism $\varphi : G \to H$ extending γ.

For this reason, we introduce an intermediate algorithmic model we call **Certificate List-Decoding** that avoids the HomExt bottleneck and works in the alternating vs. arbitrary setting.

Our new combinatorial tools allow us to play on the relatively well-understood top layers

of the subgroup lattice of the domain, avoiding the dependence on the codomain, a bottleneck in previous work. We also introduce "mean-list-decoding," a relaxation principle for constraints on the domain, that automatrically upgrades results such as {abelian→abelian} to {arbitrary→abelian}.

While motivated by bridging the mentioned gap in list-decoding, HomExt is also of independent interest, both as a problem in computational group theory and as a new and natural problem in NP of unsettled complexity status.

We consider the case $H = S_m$ (the symmetric group of degree m), i.e., $\gamma : G \rightharpoonup H$ gives a group action by the subgroup generated by the domamin of γ. We assume $G \leq S_n$ is given as a permutation group by a list of generators. We characterize the equivalence classes of extensions in terms of a multidimensional oracle subset-sum problem. From this we infer that for bounded G the HomExt problem can be solved in polynomial time.

We are most concerned with the case $G = A_n$ (the alternating group of degree n) for variable n under the assumption that the index of M in G is bounded by poly(n). We solve this case in polynomial time for all $m < 2^{n-1}/\sqrt{n}$. This is the case required for the main list-decoding result.

The Homomorphism Extension results are solo work; the rest are joint with László Babai and Timothy Black.

CHAPTER 1

INTRODUCTION

1.1 Brief history

Let G and H be finite groups, to be referred to as the *domain* and the *codomain*, respectively. A map $\varphi : G \to H$ is an *affine homomorphism* if it satisfies $\varphi(ab^{-1}c) = \varphi(a)\varphi(b)^{-1}\varphi(c)$ for all $a, b, c \in G$. Affine homomorphisms are the translations of homomorphisms. We write $\mathrm{Hom}(G, H)$ and $\mathrm{aHom}(G, H)$ to denote the set of homomorphisms and affine homomorphisms, respectively. Let H^G denote the set of all functions $f : G \to H$.

We view $\mathrm{aHom}(G, H)$ as a (nonlinear) code within the code space H^G (the space of possible "received words") and refer to this class of codes as *homomorphism codes*.

Homomorphism codes are candidates for efficient *local* list-decoding *up to mimimum distance* ($\mathsf{mindist}$) and in many cases it is known that their minimum distance is (asymptotically) equal to the list-decoding bound (LDB).

This line of work goes back to the celebrated paper by Goldreich and Levin (1989) [GL89] who found local list-decoders for Hadamard codes, i. e., for homomorphism codes with domain $G = \mathbb{Z}^n$ and codomain $H = \mathbb{Z}_2$. This result was extended to homomorphism codes of abelian groups (both the domain and the codomain abelian) by Grigorescu, Kopparty, and Sudan (2006) [GKS06] and Dinur, Grigorescu, Kopparty, and Sudan (2008) [DGKS08] and to the case of supersolvable domain and nilpotent codomain by Guo and Sudan (2014) [GS14], cf. [BGSW18].

While homomorphism codes have low (logarithmic) rates, they tend to have remarkable list-decoding properties that are not expected to hold for denser codes. In particular, in all cases of homomorphism codes studied so far (including the present work), for an *arbitrary* received word $f \in H^G$, and any $\varepsilon > 0$, the number of codewords within radius $(\mathsf{mindist} - \varepsilon)$ is bounded by $\mathrm{poly}(1/\varepsilon)$ (as opposed to some faster-growing function of ε, as permitted in the

theory of list-decoding). This is an essential feature for the complexity-theoretic application (hard-core predicates) by Goldreich and Levin.

We call the poly$(1/\varepsilon)$ bound *economical*, and a homomorphism code permitting such a bound **combinatorially economically list-decodable (CombEcon)**.

By *local* decoding we mean an algorithm that uses only poly$(\log|G|, 1/\varepsilon)$ queries to the received word. By *efficient* local decoding we mean a local algorithm that uses only poly$(\log|G|, \log|H|, 1/\varepsilon)$ additional work. We call a CombEcon code **AlgEcon (algorithmically economically list-decodable)** if it permits efficient decoding in this sense. So the cited results show that homomorphism codes with abelian domain and codomain, and more generally with supersolvable domain and nilpotent codomain, are CombEcon and AlgEcon.

In all work on the subject, the stated efficiency depends on the computational representation of the groups studied, such as special types of presentations in terms of generators and relators, black-box access, or explicit representation as permutation groups. We shall make the required representation (the access model) explicit in all results.

1.2 Contributions

1.2.1 Combinatorial bounds

We further expand the range of groups for which efficient local list-decoding is possible up to the minimum distance. In particular, for the first time, we **do not require either G or H to be solvable**. In fact, in our combinatorial and semi-algorithmic results (see below), **the codomain is an arbitrary (finite of infinite) group**. We say that a class $\mathfrak{G}$ of finite groups is **universally CombEcon** if for all $G \in \mathfrak{G}$ and arbitrary (finite or infinite) H, the code aHom(G, H) is CombEcon. The work in this thesis is the first to demonstrate the existence of significant universally CombEcon classes.

Convention 1.2.1. When speaking of a homomorphism code aHom(G, H), the domain G

will always be a finite group, but the codomain H will, in general, not be restricted to being finite.

Theorem 1.2.2 (Main combinatorial result)**.** *Alternating groups are universally CombEcon.*

We have an analogous result for abelian groups [BBW18], not included in this dissertation. See Section 1.4.

We explain the main combinatorial result in detail. By "distance" in a code we mean normalized Hamming distance.

(Restatement of Theorem 1.2.2.) Let the domain G be a finite alternating group and H an arbitrary (finite or infinite) group. Let $\mathsf{mindist}$ denote the minimum distance of the homomorphism code $\mathrm{aHom}(G, H)$ and let $\varepsilon > 0$. Let $f \in H^G$ be an arbitrary received word. Then the number of codewords within $(\mathsf{mindist} - \varepsilon)$ of f is at most $\mathrm{poly}(1/\varepsilon)$.

We prove that the degree of the $\mathrm{poly}(1/\varepsilon)$ bound at most 9; with additional work, this can be improved to 7.

Our choice of the alternating groups as the domain is our test case of what we believe is a general phenomenon valid for all finite simple groups.

Conjecture 1.2.3. *The class of finite simple groups is universally CombEcon.*

Theorem 1.2.2 also holds for a hierarchy of wider classes of finite groups we call *shallow random generation* groups or "SRG groups." This class includes the alternating groups. The defining feature of these groups is that a bounded number of random elements generate, with extremely high probability, a "shallow" subgroup, i. e., a subgroup at bounded distance from the top of the subgroup lattice.

Our new combinatorial tools allow us to play on the relatively well-understood top layers of the subgroup lattice of the domain, avoiding the dependence on the codomain, a bottleneck in previous work.

1.2.2 Algorithms

We say that a class $\mathfrak{G}$ of finite groups is **universally AlgEcon** if for all $G \in \mathfrak{G}$ and arbitrary finite H, the code $\mathrm{aHom}(G, H)$ is AlgEcon. The validity of such a statement depends also on the algorithmic representation of the domain and codomain.

A *permutation representation of degree* m of a group G is a homomorphism $G \to S_m$, where the codomain is the symmetric group of degree m. We also obtain efficient local list-decoding for the permutation representations of alternating groups under a rather generous restriction on the size of the permutation domain.

Theorem 1.2.4 (Main algorithmic result)**.** *Let $G = A_n$ be the alternating group and $H = S_m$ the symmetric group of degree m. If $m < 2^{n-1}/\sqrt{n}$, then* $\mathrm{aHom}(G, H)$ *is AlgEcon.*

The limitations on the codomain arise from severe technical difficulties encountered.

In contrast to all previous work, in the alternating case the minimum distance does not necessarily correspond to a subgroup of smallest index (modulo the "irrelevant kernel," see Chapter 4, more specifically Section 4.1.2). This necessitates the introduction of the **Homomorphism Extension** (**HomExt**) **Problem**, which remains the principal bottleneck in algorithmic progress. This problem is the focus of the next section, where we state the corresponding HomExt result needed for Theorem 1.2.4.

To bypass the HomExt bottleneck, we introduce a new model we call **Certificate List-Decoding**. This model mirrors that of algorithmic list-decoding, except that it returns an output list of *partial maps*. The list must include, for every affine homomorphism φ within $(\mathsf{mindist} - \varepsilon)$ of the received word, a partial map γ that uniquely extends to φ, i.e., φ is the unique affine homomorphism that satisfies $\varphi|_{\mathrm{dom}\,\gamma} = \gamma$. We say that a homomorphism code is **economically certificate-list-decodable (CertEcon)** if such a list of partial maps can be efficiently generated.

Note that, by definition, AlgEcon $\implies$ CertEcon $\implies$ CombEcon.

We say that a class $\mathfrak{G}$ of finite groups is **universally CertEcon** if for all $G \in \mathfrak{G}$ and arbitrary (finite or infinite) H, the code $\mathrm{aHom}(G, H)$ is CertEcon.

Theorem 1.2.5 (Main semi-algorithmic result)**.** *Alternating groups are universally CertEcon.*

In fact, we show that SRG groups are universally CertEcon.

Finally, we show that certificate list-decoding, combined with a HomExt oracle for the top layers of the subgroup lattice of G, suffices for list-decoding $\mathrm{aHom}(G, H)$. This is the route we take to proving Theorem 1.2.4.

We give more formal statements of these results in Section 3.2.

1.2.3 Homomorphism extension

We define the Homomorphism Extension Problem and state the main result. This problem will be the focus of Chapter 7. Stronger versions of the results stated here are given in Section 7.1.5.

Definition 1.2.6. Homomorphism Extension

Instance: Groups G and H and a partial map $\gamma : G \rightharpoonup H$.

Solution: A homomorphism $\varphi \in \mathrm{Hom}(G, H)$ that extends γ, i.e., $\varphi|_{\mathrm{dom}\,\gamma} = \gamma$.

The **Homomorphism Extension** Decision Problem (HomExt) asks whether a solution exists. The **Homomorphism Extension** Search Problem asks whether a solution exists and, if so, to find one.

The complexity of these problem depends on the representation of the groups involved. We shall assume that G and H are given as permutation groups. In particular, we address the subcase of group actions, i.e., $H = S_m$ is the symmetric group on m elements. We call this problem HomExtSym.

Theorem 1.2.7. HOMEXTSYM *can be solved in* $\mathrm{poly}(n, m)$ *time in any of the following cases. To fix notation, the instance consists of* $G \leq S_n$, $H = S_m$ *and a partial map* $\gamma : G \rightharpoonup H$.

(1) G *has bounded order.*

(2) $G = A_n$, $m < 2^{n-1}/\sqrt{n}$, *and the subgruop generated by* $\mathrm{dom}\,\gamma$ *has* $\mathrm{poly}(n)$ *index in* G.

(3) $m > 2^{1.7^{n^2}}$.

Case (2) is used in the list-decoding result above. Case (1) is a special case of Case (3).

This result is proved by looking at the orbits in $[m]$ of the group generated by the domain of the partial function, then deciding how they may combine to form orbits of G. We reformulate HOMEXTSYM as an exponentially large instance of a generalized Subset Sum Problem to which we have oracle access. The technical assumption $m < 2^{n-1}/\sqrt{n}$ guarantees that the arising generalized Subset Sum instance is tractable. Answering oracle queries amounts to solving certain problems of computational group theory.

The more general **Homomorphism Extension** Threshold-k Enumeration Problem asks to list all solutions unless there are more than k, in which case list k of them. This problem holds significance in the list-decoding context (see Section 6.3.5). Chapter 7 in fact proves a version of Theorem 1.2.7 for this problem with $\mathrm{poly}(n, m, k)$ time.

1.2.4 Relaxation principles and mean-list-decoding

For two groups G and H, the (G, H)-irrelevant kernel is found by intersecting the kernels of all $G \to H$ homomorphisms. If N is the (G, H)-irrelevant kernel, we will see that list-decoding $\mathrm{aHom}(G, H)$ is essentially the same as list-decoding $\mathrm{aHom}(G/N, H)$ (Theorem 4.3.2). From this we infer a domain-relaxation principle (Theorem 4.3.13). For instance, an "{abelian→abelian} is CombEcon" result automatically extends to an "{arbitrary→abelian}

is CombEcon" result, since in this case the commutator subgroup G' is contained in the irrelevant kernel. This type of relaxation applies in the context of CertEcon and AlgEcon as well.

The principle described above is achieved through the introduction and analysis of **mean-list-decoding** (Section 4.2). Instead of list-decoding one received word f, we list-decode a family $\mathscr{F}$ of received words to find the codewords within a given radius on average (over received words in $\mathscr{F}$).

Theorem 1.2.8 (Equivalence of mean-list-decoding)**.** *A code is CombEcon mean-list-decodable if and only if it is CombEcon list-decodable. The corresponding statement is true for CertEcon and AlgEcon as well.*

We remark that there is a similar equivalence between list-decoding codes $\mathrm{aHom}(G, H)$ of affine homomorphisms and list-decoding codes $\mathrm{Hom}(G, H)$ of homomorphisms, in the contexts of CombEcon, CertEcon and AlgEcon (Section 4.3.3). This result and Theorem 1.2.8 are derived using the same combinatorial lemma (Lemma 4.1.1).

1.3 Structure of the thesis

Chapter 2 contains necessary background on group theory, computational representations of groups, and coding theory. We remark in particular that Section 2.4 contains background on permutation group algorithms used only for the Homomorphism Extension algorithms in Chapter 7.

Chapter 3 contains terminology and presentation of results. Section 3.1 establishes terminology for general codes, while Section 3.2 establishes terminology and makes formal statements for homomorphism codes.

Chapter 4 introduces mean-list-decoding and shows that list-decoding and mean-list-decoding are equivalent in the contexts of CombEcon, CertEcon, and AlgEcon. The same

argument, a bipartite covering lemma, shows that list-decoding $\mathrm{Hom}(G, H)$ and list-decoding $\mathrm{aHom}(G, H)$ are equivalent in these contexts.

Chapter 5 addresses our universally CombEcon results. We present our combinatorial tools in Section 5.2 followed by the results for alternating groups in Section 5.3.

Chapter 6 addresses our AlgEcon results. We introduce SRG groups and prove the universal CombEcon and universal CertEcon results in Section 6.2. We discuss currently available methods to generate algorithmic list-decodability results in Section 6.3, focusing particularly on the role of mindist and how to bridge the gap from certificate-list-decoder to list-decoder.

Chapter 7 introduces the Homomorphism Extension Problem and provides efficient solutions in certain cases.

1.4 Sources; representation of joint work

Most of the material in this thesis is based on the two papers, [BBW18] and [Wuu18].

The thesis includes the solo paper [Wuu18] in full.

Regarding the joint paper [BBW18], it is expected that the material in this dissertation will somewhat overlap with the forthcoming dissertation of my coauthor Tim Black. Some of this is inevitable. In order to minimize the overlap, we omit all details of the results with abelian domain, which are expected to be covered in full in his dissertation.

CHAPTER 2

BACKGROUND

We write $\mathbb{N}$ for $\mathbb{N} = \{0, 1, 2, \ldots\}$.

Let G be a set. For any subset $S \subseteq G$, define the **density** of S in G by $\mu_G(S) = \frac{|S|}{|G|}$. We call G the "ambient set" and write $\mu(S) = \mu_G(S)$ when G is understood. The ambient set will generally be a group G.

2.1 Groups

We will denote the class of all groups (finite or infinite) by $\mathfrak{Groups}$. We write $\mathfrak{Abel}$ to denote the class of finite abelian groups and $\mathfrak{Alt}$ for the class of (finite) alternating groups.

Our group theory reference is [Rob95]. We review some definitions and facts.

Let G be a group. We write $H \leq G$ to express that H is a subgroup; we write $H \trianglelefteq G$ if H is a normal subgroup. We refer to cosets of subgroups of G as **subcosets**. For the subcoset aH of G (where $H \leq G$), let $|G : aH| := |G : H|$ denote the index of H in G. For a subset S of a group G, the **subgroup** $\langle S \rangle$ **generated by** S is the smallest subgroup of G containing S. If $\langle S \rangle = G$, then S **generates** G. A subset $K \subseteq G$ is **affine-closed** if $(\forall a, b, c \in K)(ab^{-1}c \in K)$. An affine-closed subset is either empty or it is a subcoset. The intersection of affine-closed subsets is affine-closed. The **affine closure** $\langle S \rangle_{\mathsf{aff}}$, affinely generated by S, is the the smallest affine-closed subset containing S. Note that the affine closure of the empty set is empty. For any $q \in S$, we have that $\langle S \rangle_{\mathsf{aff}} = q \cdot \langle q^{-1}r \mid r \in S \rangle$.

2.1.1 Series and solvable groups

A **subnormal series** for G is a sequence of subgroups

$$1 = G_0 \trianglelefteq G_1 \trianglelefteq \cdots \trianglelefteq G_k = G,$$

where $G_{i-1} \trianglelefteq G_i$ for all $1 \leq i \leq k$ (but $G_i \trianglelefteq G$ is not guaranteed). The **terms** of the series are the groups G_i for $0 \leq i \leq k$, and the **factors** of the series are the groups G_i/G_{i-1} for $1 \leq i \leq k$. A **subnormal cyclic series** is a subnormal series whose factors are cyclic groups. A group is **solvable** if it has a subnormal cyclic series.

A **normal series** is a subnormal series whose terms are all normal in G. A **normal cyclic series** is a normal series whose factors are cyclic groups. A group is **supersolvable** if it has a normal cyclic series.

If a subnormal series is a subsequence of another, the latter is said to be a **refinement** of the former. If both series have the same set of factors, the refinement is **trivial**. A subnormal series with no nontrivial refinement is a **composition series**. A normal series with no nontrivial (normal series) refinement is a **principal series** or **chief series**.

The **commutator** $[h, k]$ of two group elements h and k is defined as $[h, k] := h^{-1}k^{-1}hk$. The **commutator** $[H, K]$ of two groups H and K is the group generated by their commutators $[H, K] := \langle [h, k] \mid h \in H, k \in K \rangle$. The **commutator subgroup** of G is defined as $G' := [G, G]$. The **center** of a group G is defined as $Z(G) := \{z \in G \mid (\forall g \in G)([z, g] = 1)\}$.

The **lower central series** $G_0 \rhd G_1 \rhd G_2 \rhd \cdots$ of a group G is defined recursively by $G_0 = G$ and $G_i = [G_{i-1}, G]$. A group G is **nilpotent** if its lower central series terminates in the identity after a finite number of steps. The **class** of a nilpotent group is the number of such steps required to reach the identity. In particular, a nilpotent group is of *class* 2 if $G' \leq Z(G)$.

2.1.2 Presentations of groups

Let X be a set. Let $\mathcal{F}_X$ denote the free group generated by X, the set of words over $X \cup X^{-1}$ as well as the element $\{e\}$, with the group operation given by concatenation.

Definition 2.1.1. A pair $\langle X \mid \mathcal{R} \rangle$ is a *presentation* for a group G if

1. the *generators* $X = \{g_1, \ldots, g_k\}$ is a subset of G,

2. the *relations* $\mathcal{R}$ is a set of words in $\mathcal{F}_X$, and

3. G is isomorphic to the quotient $\mathcal{F}_X$ by the normal subgroup generated by $\mathcal{R}$.

We will also write $g_{i_1}^{m_1} \cdots g_{i_\ell}^{m_\ell} = g_{j_1}^{n_1} \cdots g_{j_{\ell'}}^{n_{\ell'}}$ to refer to the relation $g_{i_\ell}^{-m_\ell} \cdots g_{i_1}^{-m_1} g_{j_1}^{n_1} \cdots g_{j_{\ell'}}^{n_{\ell'}}$.

Fact 2.1.2. Let G be a group with presentation $\langle X \mid \mathcal{R} \rangle$, and let H be a group. Let $\theta \colon X \to H$. Then θ extends to a homomorphism in $\mathrm{Hom}(G, H)$ if and only if for all $h_1^{n_1} \cdots h_\ell^{n_\ell} \in \mathcal{R}$ we have $\theta(h_1)^{n_1} \cdots \theta(h_\ell)^{n_\ell} = 1$. When θ extends to a homomorphism, it extends uniquely.

Let $\mathcal{F}_X^{\mathsf{ab}}$ denote the free abelian group group generated by X, the set of linear combinations $\sum a_x x$, where $a_x \in \mathbb{Z}$ for all $x \in X$.

Let $\mathsf{ab} : \mathcal{F}_X \to \mathcal{F}_X^{\mathsf{ab}}$ given by $r \mapsto r^{\mathsf{ab}}$ denote the natural abelianization map, where the coefficient a_x of x in $r^{\mathsf{ab}} = \sum a_x x$ is found by summing the exponents of all occurrences of x in the word r. For $\mathcal{R} \subset \mathcal{F}$, denote by $\mathcal{R}^{\mathsf{ab}} := \{r^{\mathsf{ab}} : r \in \mathcal{R}\}$ the image of $\mathcal{R}$ under ab.

Definition 2.1.3. A pair $\langle X | \mathcal{R}^{\mathsf{ab}} \rangle_{\mathsf{ab}}$ is an **abelian presentation** for an abelian group G if

1. the *generators* $X = \{g_1, \ldots, g_k\}$ form a subset of G,

2. the *relations* $\mathcal{R}^{\mathsf{ab}}$ form a set of words in $\mathcal{F}_X^{\mathsf{ab}}$, and

3. G is isomorphic to the quotient $\mathcal{F}_X^{\mathsf{ab}}$ by the subgroup generated by $\mathcal{R}^{\mathsf{ab}}$.

Fact 2.1.4. Let G be a group given by the presentation $\langle X | \mathcal{R} \rangle$. Then, the abelianization G/G' of G has presentation $\langle X | \mathcal{R}^{\mathsf{ab}} \rangle_{\mathsf{ab}}$.

2.1.3 Smith normal form

In this section we briefly review Smith normal form of matrices. For more details see [Nor12]. Smith normal form will be used in the algorithm FindLambda of Section 6.3.6.

Denote by $\mathrm{Diag}^{m \times n}(a_1, \ldots, a_k)$ the $m \times n$ matrix with entries $a_1, \ldots, a_k$ on the diagonal and zeros everywhere else, where $k = \min\{m, n\}$.

Definition 2.1.5. Let A be an $m \times n$ integer matrix. The matrix A is in **Smith normal form** if $A = \text{Diag}^{m\times n}(a_1, \ldots, a_\ell, 0, \ldots, 0)$, where $a_i > 0$ for $i = 1, \ldots, \ell$ and a_i divides a_{i+1} for $i = 1, \ldots, \ell - 1$.

A square integer matrix is **unimodular** if its determinant is ± 1.

Theorem 2.1.6. *For every $m \times n$ integer matrix A, there exists a unimodular $m \times m$ integer matrix S and a unimodular $n \times n$ integer matrix T such that SAT is in Smith normal form.*

This matrix SAT is the **Smith normal form** of A. The nonzero diagonal entries $a_1, \ldots, a_\ell$ of the Smith normal form of A are the **elementary divisors of A.** The Smith normal form of every integer matrix is unique.

Definition 2.1.7 (Converting between integer matrices and abelian presentations)**.** Let $\mathcal{R}^{\mathsf{ab}}$ be a set of abelian relations on the generators $\{g_1, \ldots, g_n\}$. We denote by $A(\mathcal{R}^{\mathsf{ab}}) = \{a_{r,i}\}$ the $|\mathcal{R}^{\mathsf{ab}}| \times n$ integer matrix that encodes the relations $r \in \mathcal{R}^{\mathsf{ab}}$, using the expressions $r = a_{r,1}g_1 + \ldots + a_{r,n}g_n$, where $a_{r,i} \in \mathbb{Z}$ for $i = 1, \ldots, n$.

Let A be an $m \times n$ integer matrix. Let $\mathcal{R}^{\mathsf{ab}}(A) = \{a_{r,1}g_1 + \ldots + a_{r,n}g_n \mid r = 1, \ldots, m\}$ be the set of relations on $\{g_1, \ldots, g_n\}$ with coefficients corresponding to the rows of A.

Observation 2.1.8. *Multiplication by unimodular matrices does not change the abelian group represented. In other words, if A is an $m \times n$ integer matrix, if S is an $m \times m$ unimodular matrix, and if T is an $n \times n$ unimodular matrix, then $\langle X|\mathcal{R}^{ab}(A)\rangle_{ab} \cong \langle X|\mathcal{R}^{ab}(SAT)\rangle_{ab}$.*

Suppose the $m \times n$ matrix $A = A(\mathcal{R}^{\mathsf{ab}})$ encodes the relations $\mathcal{R}^{\mathsf{ab}} = \{r_1, \ldots, r_m\}$ over $\mathbf{g} = \{g_1, \ldots, g_n\}$. Then, SAT encodes the set of relations $S\mathbf{R}$ (where $\mathbf{R} = (r_1, \ldots, r_m)^\top$) over $\mathbf{h} = T^{-1}\mathbf{g}$.

Observation 2.1.9. *Consider an abelian group G given by the presentation $G = \langle X|\mathcal{R}^{ab}\rangle_{ab}$ where $X = \{g_1, \ldots, g_n\}$ and $\mathcal{R}^{ab} = \{b_i g_i | i = 1, \ldots, \ell\}$, where $n \geq \ell$ and $b_i \in \mathbb{Z}^+$ for $i = 1, \ldots, \ell$. Then,*

$$G \cong (\mathbb{Z}/b_1\mathbb{Z} \times \cdots \times \mathbb{Z}/b_\ell\mathbb{Z}) \times \mathbb{Z}^{n-\ell}.$$

Recall that all groups are finite, so $n = \ell$.

Definition 2.1.10. Let G be an abelian group. A **canonical decomposition** of G is a presentation as given in Observation 2.1.9, such that b_i divides b_{i+1}, for $i = 1, \ldots, \ell - 1$.

Definition 2.1.11. Let G be an abelian group. A **prime-power decomposition** of G is a presentation as given in Observation 2.1.9, such that each b_i is a prime-power or 1.

The following corollary follows from Theorem 2.1.6 on Smith normal form.

Corollary 2.1.12 (Converting any presentation of an abelian group into canonical form)**.** *Let G be an abelian group with abelian presentation $G = \langle X | \mathcal{R}^{ab} \rangle_{ab}$. If G is given in the canonical decomposition as*

$$G \cong (\mathbb{Z}/a_1\mathbb{Z} \times \cdots \times \mathbb{Z}/a_\ell\mathbb{Z}) \times \mathbb{Z}^{k-\ell},$$

then $a_1, \ldots, a_\ell$ correspond exactly to the elementary divisors of $B(\mathcal{R}^{ab})$.

In [KB79], Kannan and Bachem proved that the conversion to Smith normal form can be computed in polynomial time.

Theorem 2.1.13 (Kannan and Bachem)**.** *There exists a deterministic algorithm that, given an integer matrix A, will compute the unimodular matrices S and T such that SAT is in Smith normal form, in polynomial time (polynomial in the bit-length of the input).*

Combining the discussion of this section with Theorem 2.1.13, we find that the canonical decomposition of any abelian group can be found from any presentation in polynomial time. We remark that converting from the canonical decomposition to the prime-power decomposition requires prime factorization.

2.1.4 Supersolvable presentations

A **polycyclic series** is a subnormal series $1 = G_0 \trianglelefteq G_1 \trianglelefteq G_2 \trianglelefteq \cdots \trianglelefteq G_k$ such that each factor G_i/G_{i-1} is cyclic. A group G is a **polycyclic group** if there exists a polycyclic series such that $G_k = G$. A finite group is polycyclic if and only if it is solvable.

Let G be a polycyclic group. A sequence of elements $g_1, \ldots, g_k$ is a **polycyclic sequence** for G if there is a polycyclic series $1 = G_0 \trianglelefteq G_1 \trianglelefteq G_2 \trianglelefteq \cdots \trianglelefteq G_k = G$ satisfying $\langle g_i, G_{i-1}\rangle = G_i$.

The **relative order** of $g \in G$ with respect to $N \trianglelefteq G$ is the index $|\langle g, N\rangle : N|$. Let $g_1, \ldots, g_k$ be a polycyclic sequence for G. The **relative order** of g_i is the index $r_i := |G_i : G_{i-1}|$, the relative order of g_i with respect to $G_{i-1} \trianglelefteq G_i$.

For every $g \in G$, there exists a unique sequence $(e_k, \ldots, e_1)$, with $0 \leq e_i \leq r_i - 1$ for $1 \leq i \leq n$, such that $g = g_k^{e_k} \cdots g_1^{e_1}$. This expression $g = g_k^{e_k} \cdots g_1^{e_1}$ is the **normal form** of g with respect to the polycyclic sequence $g_1, \ldots, g_k$.

A **polycyclic presentation** for G is a presentation consisting of an ordered list of **generators** $g_1, \ldots, g_k$ and relations of the form

$$g_i^{m_i} = g_{i-1}^{a_{i;i-1}} \cdots g_1^{a_{i;1}} \qquad \text{for } 1 \leq i \leq k, \tag{2.1}$$

$$g_i^{-1} g_j g_i = g_{i-1}^{b_{i,j;i-1}} \cdots g_1^{b_{i,j;1}} \qquad \text{for } 1 \leq j < i \leq k, \text{ and} \tag{2.2}$$

$$g_i g_j g_i^{-1} = g_{i-1}^{c_{i,j;i-1}} \cdots g_1^{c_{i,j;1}} \qquad \text{for } 1 \leq j < i \leq k, \tag{2.3}$$

with $m_i > 0$ and $a_{i;\ell}, b_{i,j;\ell}, c_{i,j;\ell} \geq 0$ for each i, j, ℓ. The numbers $m_1, \ldots, m_k$ are the **power exponents** of the polycyclic presentation. Notice that $r_i \mid m_i$ for each $1 \leq i \leq k$, or, the relative orders divide the power exponents.

A group has a polycyclic presentation if and only if it is a polycyclic group. A polycyclic presentation is **reduced** if, for all i, j, ℓ, we have $a_{i;\ell}, b_{i,j;\ell}, c_{i,j;\ell} < m_\ell$.

If G is given by a polycyclic presentation with generators $g_1, \ldots, g_k$, then the chain of

groups $1 = G_0 \trianglelefteq G_1 \trianglelefteq G_2 \trianglelefteq \cdots \trianglelefteq G_k = G$, given by $G_i = \langle g_i, G_{i-1} \rangle = \langle g_1, \ldots, g_i \rangle$, forms a polycyclic series. So, the generators for a polycyclic presentation form a polycyclic sequence.

A polycyclic presentation is **confluent** if $m_i = r_i$ for $1 \leq i \leq k$, or, the power exponents $m_1, \ldots, m_k$ of the presentation are identical to the relative orders $r_1, \ldots, r_k$ of the polycyclic sequence $g_1, \ldots, g_k$, or, .

A polycyclic presentation $\langle g_1, \ldots, g_k \mid \mathcal{R} \rangle$ is a **supersolvable presentation** if the last two types of relations have the form:

$$g_i^{-1} g_j g_i = g_j^{b_{i,j;j}} \cdots g_1^{b_{i,j;1}} \qquad \text{for } 1 \leq j < i \leq k \tag{2.4}$$

$$g_i g_j g_i^{-1} = g_j^{c_{i,j;j}} \cdots g_1^{c_{i,j;1}} \qquad \text{for } 1 \leq j < i \leq k \tag{2.5}$$

Definition 2.1.14. We call a supersolvable presentation ***earnest*** if it is confluent and reduced, and the associated power exponents $m_1, \ldots, m_k$ are prime numbers with $m_1 \geq m_2 \geq \cdots \geq m_k$.

Every supersolvable presentation defines a supersolvable group. By Zappa, every supersolvable group has an earnest supersolvable presentation.For reference, see [Rob95].

Proposition 2.1.15 (Zappa)**.** *If G is a (finite) supersolvable group and $|G| = p_1 \cdots p_k$, where $p_1 \geq \cdots \geq p_k$ are primes, then G has a principal series*

$$1 = G_0 \lhd G_1 \lhd \cdots \lhd G_k = G$$

where each $G_i / G_{i-1} \cong \mathbb{Z}_{p_i}$.

Other than the usage of "earnest," the terminology defined above is standard in computational group theory. For further reference, see [HEO05, Chapter 8]. We have defined earnest supersolvable presentations for convenience.

2.2 Homomorphism codes

2.2.1 Affine homomorphisms as codewords

Let G be a finite group and H a group. Denote the set of homomorphisms from G to H by $\mathrm{Hom}(G, H)$.

Let G_1 and H_1 be affine-closed subsets of G and H, respectively.

Definition 2.2.1 (Affine homomorphisms)**.** A function $\varphi\colon G_1 \to H_1$ is an **affine homomorphism** if

$$(\forall a, b, c \in G_1)(\varphi(a)\varphi(b)^{-1}\varphi(c) = \varphi(ab^{-1}c))\,.$$

Definition 2.2.2 (aHom)**.** We write $\mathrm{aHom}(G_1, H_1)$ to denote the set of affine homomorphisms from G_1 to H_1.

Fact 2.2.3. Let $G_1 \leq G$ and $H_1 \leq H$. Let $a \in G$ and $b \in H$. A function $\varphi\colon aG_1 \to bH_1$ is an affine homomorphism if and only if there exists $h \in H$ and $\varphi^0 \in \mathrm{Hom}(G_1, H_1)$ such that

$$\varphi(g) = h \cdot \varphi^0(g) \tag{2.6}$$

for every $g \in G_1$. The element h and the homomorphism φ_0 are unique.

The analogous statement also holds with h on the right of φ_0.

Definition 2.2.4. For sets G, H and functions $f, g\colon G \to H$, the **equalizer** $\mathrm{Eq}(f, g)$ is the subset of G on which f and g agree, i. e.,

$$\mathrm{Eq}(f, g) := \{x \in G \mid f(x) = g(x)\}.$$

More generally, if Φ is a collection of functions from G to H, then the **equalizer** $\mathrm{Eq}(\Phi)$ is the set

$$\mathrm{Eq}(\Phi) := \{x \in G \mid (\forall f, g \in \Phi)(f(x) = g(x))\}.$$

Fact 2.2.5. (a) If $\varphi, \psi \in \mathrm{Hom}(G, H)$ then $\mathrm{Eq}(\varphi, \psi) \leq G$.

(b) If $\varphi, \psi \in \mathrm{aHom}(G, H)$ then $\mathrm{Eq}(\varphi, \psi)$ is affine-closed. Moreover, if $\varphi_0, \psi_0 \in \mathrm{Hom}(G, H)$ are the corresponding homomorphisms (see (2.6)) then either $\mathrm{Eq}(\varphi, \psi)$ is empty or $\mathrm{Eq}(\varphi, \psi) = g \cdot \mathrm{Eq}(\varphi_0, \psi_0)$ for any $g \in \mathrm{Eq}(\varphi, \psi)$.

Remark 2.2.6 (Why affine?)**.** The reader may ask, why we (and all prior work) consider affine homomorphisms rather than homomorphisms. The reason is that affine homomorphisms are simply the more natural objects in this context. To begin with, this object is more homogeneous. For instance, for finite H, under random affine homomorphisms, the images of any element $g \in G$ are uniformly distributed over H. This uniformity also serves as an inductive tool: when extending the domain from a subgroup G_0 to a group G, the action of any homomorphism $\varphi \in \mathrm{Hom}(G, H)$ can be split into actions on the cosets of G_0 in G. Those actions are affine homomorphisms. On the other hand we also note that list-decoding of $\mathrm{Hom}(G, H)$ and $\mathrm{aHom}(G, H)$ are essentially equivalent tasks, see Section 4.3.3.

2.2.2 Agreement parameter Λ

Recall that the (normalized) **agreement** $\mathrm{agr}(f, g)$ between two functions $f, g \colon G \to H$ is given by

$$\mathrm{agr}(f, g) := \frac{|\mathrm{Eq}(f, g)|}{|G|}.$$

Definition 2.2.7 ($\Lambda_{G,H}$)**.** We write

$$\Lambda_{G,H} := \Lambda_{\mathrm{aHom}(G,H)}$$

for the **maximum agreement** of $\mathrm{aHom}(G, H)$. In other words,

$$\Lambda_{G,H} := \max_{\substack{\varphi,\psi \in \mathrm{aHom}(G,H) \\ \varphi \neq \psi}} \mathrm{agr}(\varphi, \psi)$$

If the groups G and H are understood, we often write Λ in place of $\Lambda_{G,H}$. Using this terminology, the minimum distance (Definition 3.1.1) of the homomorphism code $\mathrm{aHom}(G, H)$ is $(1 - \Lambda_{G,H})$.

We present results regarding the value of Λ. We remark that Λ is in general nontrivial to determine.

The following appears in [Guo15, Proposition 3.5].

Proposition 2.2.8 (Guo)**.** *Let G, H be groups. The maximum agreement $\Lambda_{G,H}$ can equivalently be defined with* aHom *replaced by* Hom*, i. e.,*

$$\Lambda_{\mathrm{Hom}(G,H)} = \Lambda_{\mathrm{aHom}(G,H)}.$$

Here we use the convention that the maximum of the empty set (of nonnegative numbers) is zero. Otherwise we would need to make the additional assumption $|\mathrm{Hom}(G, H)| > 1$.

As a corollary we get an upper bound on Λ.

Corollary 2.2.9. *Let G be a finite group and H a group. Then, $\Lambda \leq \max\{\mu(K) \mid K \lneq G\}$, the largest density of a proper subgroup of G.*

Next, we state the main theorem of [Guo15], which characterizes $\Lambda_{G,H}$ when G is solvable or H is nilpotent. The algorithm FindLambda of Section 6.3.6 checks (a stronger version of) this characterization to calculate $\Lambda_{G,H}$.

Theorem 2.2.10 (Guo)**.** *Let G and H be groups. Suppose that G is solvable or H is nilpotent. Then*

$$\Lambda_{G,H} = \frac{1}{p}$$

where p is the smallest prime divisor of $\gcd(|G|, |H|)$ such that G has a normal subgroup of index p. If no such p exists, then $|\mathrm{Hom}(G, H)| = 1$; in particular, $\Lambda_{G,H} = 0$.

The proof of Theorem 2.2.10 uses the next two observations, Proposition 2.2.11 and 2.2.12, which also provide information on Λ.

Proposition 2.2.11. *Suppose G_1, G_2 and H are finite groups. Then,*

$$\Lambda_{G_1 \times G_2, H} = \max\{\Lambda_{G_1,H}, \Lambda_{G_2,H}\}.$$

Proposition 2.2.12. *If G has a normal subgroup of prime index p and p divides $|H|$, then*

$$\Lambda_{G,H} \geq \frac{1}{p}.$$

Below is another tool on determining Λ.

Lemma 2.2.13. *Let G and H be groups and let $\Phi \subseteq \mathrm{aHom}(G, H)$. Then every prime factor of $|G : \mathrm{Eq}(\Phi)|$ divides $|H|$.*

Proof. Let $\Phi = \{\varphi_1, \ldots, \varphi_\ell\}$. Define the "diagonal" $D \leq H^\ell$ by $D = \{(h, \ldots, h) \in H^\ell \mid h \in H\}$. Then $|H^\ell : D|$ divides $|H|^\ell$, so $|(\varphi_1(g), \ldots, \varphi_\ell(g))^{-1}(H^\ell) : (\varphi_1(g), \ldots, \varphi_\ell(g))^{-1}(D)| = |G : \mathrm{Eq}(\Phi)|$ divides $|H|^\ell$. □

We observe that, by definition of Λ, determining a codeword (affine homomorphism) on a subcoset of density $> \Lambda$ will determine the codeword uniquely. This observation is used to "certificate list-decode" (see Section 6.2.5).

Fact 2.2.14. Let G and H be groups and $S \subseteq G$ a subset. If $\varphi, \psi \in \mathrm{aHom}(G, H)$ and $\varphi(x) = \psi(x)$ for all $x \in S$, then $\langle S \rangle_{\mathsf{aff}} \subseteq \mathrm{Eq}(\varphi, \psi)$. □

Corollary 2.2.15. *Let G be a finite group, H a group, and $S \subseteq G$, such that $\mu(\langle S \rangle_{\mathsf{aff}}) > \Lambda_{G,H}$. If $\varphi, \psi \in \mathrm{aHom}(G, H)$ are such that $\varphi(x) = \psi(x)$ for all $x \in S$, then $\varphi = \psi$.*

2.2.3 Notation for lists

We collect notation for lists and list size bounds for homomorphism codes. This will be presented again in generality throughout Chapter 3.

Let G and H be groups. Let $f : G \to H$ be a function. We view f as a *received word* in the list-decoding context. Let $0 \leq \lambda \leq 1$.

Definition 2.2.16 (The list $\mathcal{L}$)**.** We denote by $\mathcal{L}(\mathrm{aHom}(G, H), f, \lambda)$ the list of codewords in $\mathrm{aHom}(G, H)$ with agreement $\geq \lambda$ with f, i.e.,

$$\mathcal{L}(\mathrm{aHom}(G, H), f, \lambda) = \{\varphi \in \mathrm{aHom}(G, H) \mid \mathrm{agr}(f, \varphi) \geq \lambda\}.$$

Let $0 \leq \rho \leq 1$. For the list defined in terms of the complementary notion of distance, we denote by $\mathcal{L}^r(\mathrm{aHom}(G, H), f, \rho)$ the list of codewords in $\mathrm{aHom}(G, H)$ within distance ρ of f, i.e.,

$$\mathcal{L}^r(\mathrm{aHom}(G, H), f, \lambda) = \{\varphi \in \mathrm{aHom}(G, H) \mid \mathrm{dist}(f, \varphi) \leq \rho\}.$$

While usage of distance is standard in error correcting codes, agreement is a more natural quantity to consider for affine homomorphisms.

Definition 2.2.17 (List-size bound)**.** We denote

$$\ell(\mathrm{aHom}(G, H), \lambda) = \max_f |\mathcal{L}(\mathrm{aHom}(G, H), f, \lambda)|.$$

Similarly, we denote $\ell^r(\mathrm{aHom}(G, H), \rho) = \max_f |\mathcal{L}^r(\mathrm{aHom}(G, H), f, \rho)|$.

2.3 Computational representations of groups

In this section we discuss the models of access to groups required by our algorithms. The choice of the model significantly impacts the running time and even the feasibility of an

algorithm.

The models include oracle models (black-box access, black-box groups), generator-relator presentations, and various explicit models such as permutation groups, matrix groups, direct product of cyclic groups of known orders.

Our domain groups are always finite but the codomain may be infinite.

2.3.1 Black-box models

If the codomain is infinite, and even if it is finite but very large, the black-box-group model with its fixed-length encoding [BS84] is not appropriate (see "encoded groups" below). We start with an extension of that model.

Definition 2.3.1 (Black-box access)**.** An **unencoded black-box representation** of a (finite or infinite) group K is an ordered 5-tuple

$$(U, r, \mathsf{mult}, \mathsf{inv}, \mathsf{id})$$

where

- U is a (possibly infinite) set;
- $r\colon U \to K \cup \{*\}$ with $r(U) \supseteq K$;
- $\mathsf{mult}\colon r^{-1}(K) \times r^{-1}(K) \to r^{-1}(K)$ with $r(\mathsf{mult}(x, y)) = r(x)r(y)$ for all $x, y \in r^{-1}(K)$;
- $\mathsf{inv}\colon r^{-1}(K) \to r^{-1}(K)$ with $r(\mathsf{inv}(x)) = r(x)^{-1}$ for $x \in r^{-1}(K)$; and
- $\mathsf{id}\colon r^{-1}(K) \to \{\mathsf{yes}, \mathsf{no}\}$ with $\mathsf{id}(x) = \mathsf{yes}$ if and only if $r(x)$ is the identity in K.

We say that an algorithm has **black-box access** to the group K if the algorithm can store elements of U and query the functions (oracles) $\mathsf{mult}, \mathsf{inv}, \mathsf{id}$. We say that K is given as an **(unencoded) black-box group** if in addition a list of generators of K is given.

Remark 2.3.2. We emphasize that the difference between black-box access to a group G and the group G being given as a black-box group is that in the latter model, a list of generators of H is given, whereas no elements of G may be a priori known in the former.

If $U = \{0,1\}^n$ then we talk about an **encoded group**, of **encoding length** n. This of course implies that K is finite, namely, $|K| \leq 2^n$. (This is the model introduced in [BS84].)

In an abuse of notation, when black-box access to a group K is given, we may refer to elements of $r^{-1}(K)$ by their images under r, we may write gh in place of $\mathsf{mult}(g,h)$, we may write g^{-1} in place of $\mathsf{inv}(g)$, and we may write $g = 1$ in place of $\mathsf{id}(g) = \mathsf{yes}$.

Access to domain and codomain. In general we shall not need generators of the codomain, H, just black-box access. On the other hand, we do need generators of the domain, G; homomorphisms will be defined by their values on a set of generators. So our access to the domain will be assumed to be at least as strong as an (encoded) black-box group.

The black-box unit cost model. The (unencoded) black-box access model is particularly well suited to the **unit-cost model** where we assume that we can copy and store an element of U and query an oracle at unit cost. We shall analyze our algorithms in the unit-cost model for the codomain H. This essentially counts the operations performed in H, so its bit-cost will incur an additional factor of $O(\log |H|)$ (if H is finite and nearly optimally encoded).

Remark 2.3.3. Black-box groups are studied in a substantial body of literature, both in the theory of computing and in computational group theory (see the references in [BBS09]). It is common to make additional access assumptions to a black-box group (assume additional oracles) such as an oracle for the order of the elements.

Given a black-box group H, we cannot determine the order $|H|$ or the order of a given element $h \in H$. In fact, even with an oracle for the order of elements, $\mathbb{Z}_p$ and $\mathbb{Z}_p \times \mathbb{Z}_p$ cannot be distinguished in fewer than $p+1$ black-box queries (see [BS84]). To avoid such obstacles,

it is common to assume additional information beyond black-box access. In finding $\Lambda_{G,H}$ for abelian domain G one needs to decide if a given prime divides $|H|$. To accomplish this, we assume additional information about the group H such as the order $|H|$ or the list of primes dividing $|H|$.

2.3.2 Generator-relator presentation, homomorphism checking

By a "presentation" of a group we mean a *generator-relator presentation.*

For a group given by a presentation, basic questions such as whether the group has order 1 are undecidable. However, special types of presentations, such as polycyclic presentations of finite solvable groups, are often helpful. However, that it is not known how to efficiently perform group operations in a finite solvable group given by a polycyclic presentation, so such presentations cannot answer basic black-box queries.

Any presentation, however, can be used for homomorphism checking, a critical operation in decoding homomorphism codes. See Section 2.4.3 for more details.

2.3.3 Permutation groups

We write S_n for the symmetric group on n elements, $[n] = \{1, \ldots, n\}$.

A group G is a **permutation group** if it is a subgroup of S_n. A group G is given as a permutation group if a set of generators of G is provided and elements of G are encoded as permutations in S_n of length n.

Permutation groups are discussed more extensively in the next section.

2.4 Permutation group algorithms

We discuss permutation groups. Section 2.4.1 defines permutation groups and sets relevant notation. The remainder of this section addresses material in this section will be referenced

only in Chapter 6 on Homomorphism Extension.

2.4.1 Permutation groups: definitions

For a set Ω, $\mathrm{Sym}(\Omega)$ denotes the symmetric group on Ω and $\mathrm{Alt}(\Omega)$ denotes the alternating group on Ω. Often, we write S_n (or A_n) for the symmetric (or alternating) group on $[n] = \{1, \ldots, n\}$.

Definition 2.4.1 (Group actions)**.** A **(permutation) action** of a group G on a set Ω is given by a homomorphism $\varphi : G \to \mathrm{Sym}(\Omega)$, often denoted by $G \overset{\varphi}{\curvearrowright} \Omega$ or $G \curvearrowright \Omega$.

Recall that G is a permutation group if it is a subgroup of a symmetric group, $G \leq \mathrm{Sym}(\Omega)$. Permutation groups naturally define permutation actions.

Let $G \leq \mathrm{Sym}(\Omega)$, $g \in G$, $\omega \in \Omega$, and $\Delta \subset \Omega$.

The image of ω under g is denoted by ω^g. This notation extends to sets. So, $\Delta^g := \{\omega^g : \omega \in \Delta\}$ and $\Delta^G := \{\omega^g : \omega \in \Delta, g \in G\}$. The subset $\Delta \subset \Omega$ is **G-invariant** if $\Delta^G = \Delta$. The **orbit ω^G of ω under action by** G is given by $\omega^G := \{\omega^g : g \in G\}$. The orbits of G are G-invariant and they partition Ω. All G-invariant sets are formed by unions of orbits.

The **point stabilizer G_ω of** ω is the subgroup of G fixing ω, given by $G_\omega = \{g \in G \mid \omega^g = \omega\}$. The **pointwise stabilizer $G_{(\Delta)}$ of** Δ is the subgroup fixing every point in Δ, given by $G_{(\Delta)} = \bigcap_{\omega \in \Delta} G_\omega$. The **setwise stabilizer G_Δ of** Δ is given by $G_\Delta = \{g \in G \mid \Delta^g = \Delta\}$.

Let $\Delta \subseteq \Omega$ be G-invariant. For $g \in G$, denote by g^Δ the restriction of the action of g to Δ. The group $G^\Delta = \{g^\Delta : g \in G\} \leq \mathrm{Sym}(\Delta)$ is the image of the permutation representation of G in its action on Δ. We see that $G^\Delta \cong G/G_{(\Delta)}$.

We state a result that goes back to Jordan. Its modern formulation by Liebeck (see [DM96, Theorem 5.2A]) describes the small index subgroups of A_n. This theorem is used extensively in economical list-decoding results for {alternating→arbitrary} homomorphism codes. It is used also to categorize group actions by A_n in Theorem 7.1.9.

Theorem 2.4.2 (Jordan–Liebeck)**.** *Let $n \geq 10$ and let r be an integer with $1 \leq r < n/2$. Suppose that $K \leq A_n$ has index $|A_n : K| < \binom{n}{r}$. Then, for some $\Delta \subseteq [n]$ with $|\Delta| < r$, we have $(A_n)_{(\Delta)} \leq K \leq (A_n)_{\Delta}$.*

2.4.2 Basic results

We present results we use from the literature on permutation group algorithms. Our main reference is the monograph [Ser03].

Recall that a group G is **given** or **known** when a set of generators for G is given/known. A coset Ga is **given** or **known** if the group G and a coset representative $a' \in Ga$ are given/known. A group (or a coset) is **recognizable** if we have an oracle for membership and **recognizable in time** t if the membership oracle can be implemented in time t.

Proposition 2.4.3. *Membership in a given group $G \leq S_n$ (or coset Ga) can be tested in* poly(n) *time. In other words, a known group (or coset) is polynomial-time recognizable.*

Proof. This is accomplished by the Schreier-Sims algorithm, see [Ser03, Section 3.1 item (b)]. □

Corollary 2.4.4. *If $G_1, \ldots, G_k \leq S_n$ and $a_1, \ldots, a_k \in S_n$ are given, then the intersection $\bigcap_i G_i a_i$ is polynomial-time recognizable.*

Proposition 2.4.5. *Given $G \leq S_n$, the following can be computed in* poly(n)*-time.*

(a) A set of $\leq 2n$ generators of G.

(b) The order of G.

(c) The index $|G : M|$, for a given subgroup $M \leq G$.

(d) The orbits of G.

(e) The point stabilizers of G.

Proof. Most items below are addressed in [Ser03, Section 3.1].

(a) Denote by T the set of given generators of G. Use membership testing to prune T down to a non-redundant set of generators. By [Bab86], the length of subgroup chains in S_n is bounded by $2n$, so $|T| \leq 2n$ after pruning.

(b) See [Ser03, Section 3.1 item (c)].

(c) Compute $|M|$ and $|G|$.

(d) See [Ser03, Section 3.1 item (a)].

(e) See [Ser03, Section 3.1 item (e)].

□

Proposition 2.4.6. *Let $M \leq G$ be a recognizable subgroup of G of index $|G : M| = s$. A set of generators for M and a set of coset representatives for $M\backslash G$ can be found in* $\mathrm{poly}(n, s)$ *time (including calls to the membership oracle).*

Proof. Consider the subgroup chain $G \geq M \geq M_1 \geq M_{(12)} \geq M_{(123)} \geq M_{(12\cdots n)} = 1$ (M is followed by its stabilizer chain). Apply Schreier-Sims to this chain. (This is the "tower of groups" method introduced in [Bab79] and derandomized in [FHL80]. Note that this method only requires the subgroups in this chain to be recognizable.) □

Proposition 2.4.7. *Let $G \leq S_n$ be a given permutation group. Let $M, L \leq G$ be given subgroups. Denote their indices by $s = |G : M|$ and $t = |G : L|$.*

(a) The normalizer $N_G(M)$ can be found in $\mathrm{poly}(n, s)$*-time.*

(b) The number of conjugates of M in G can be computed in $\mathrm{poly}(n, s)$*-time.*

(c) The conjugacy of L and M in G can be decided and a conjugating element $g \in G$ such that $g^{-1}Lg = M$ can be found if it exists, in $\mathrm{poly}(n, s)$ *time.*

Proof. (a) Let S be the given set of generators of M. Take a set of coset representatives for $M\backslash G$, found by Proposition 2.4.6. Remove the coset representatives g that do not satisfy $g^{-1}Sg \subseteq M$. This is accomplished through membership testing. The remaining coset representatives, along with S, generate $N_G(M)$.

(b) The number of conjugates of M in G is the index $|G : N_G(M)|$.

(c) Check if $|L| = |M|$ by Proposition 2.4.5 (b). If not, they are not conjugate. Otherwise, let S be the set of given generators of M. Now, L and M are conjugate if and only if there exists a coset representative g for $N_G(M)\backslash G$ that satisfies $g^{-1}Sg \subseteq L$.

□

Proposition 2.4.8. *Let $G \leq S_n$ be a given permutation group. Let $M, L \leq G$ be given subgroups. Denote their indices by $s = |G : M|$ and $t = |G : L|$.*

(a) Given two elements $g, h \in G$, membership of h in the double coset LgM can be decided in $\text{poly}(n, \min\{s, t\})$-time.

(b) A set of double coset representatives for $L\backslash G/M$ can be found in $\text{poly}(n, \min\{s, t\})$-time.

Proof. (a) Without loss of generality assume that $s \leq t$. Notice that

$$h \in LgM \iff Lh \cap gM \neq \emptyset \iff g^{-1}Lh \cap M \neq \emptyset \iff (g^{-1}Lg) \cap Mh^{-1}g \neq \emptyset.$$

So, deciding whether $h \in LgM$ is equivalent to deciding whether the subgroup $L^* = g^{-1}Lg$ and coset Mg^* have non-empty intersection, where $g^* = h^{-1}g$. This intersection, $L^* \cap Mg^*$, is either empty or a right coset of $L^* \cap M$ in L^*. In what remains we check whether a coset of $L^* \cap M$ is contained in $L^* \cap Mg^*$.

Notice that $|L^* : L^* \cap M| \leq |G : M| = s$. Find a set R of coset representatives of $L^* \cap M$

in L^* using Proposition 2.4.6, noting that $L^* \cap M$ is recognizable (Corollary 2.4.4). For each representative $r \in R$, check whether $r \in L^* \cap Mg^*$ (Corollary 2.4.4).

(b) A list of t coset representatives of M in G is a redundant set of double coset representatives for $L\backslash G/M$. This can be pared down to a set of non-redundant double coset representatives by $\binom{t}{2}$ comparisons using part (a).

□

2.4.3 Generators, relations, and homomorphism checking

For a permutation group G, checking whether a $G \rightharpoonup H$ partial map that is defined on a set of generators $S \subseteq G$ extends to a homomorphism can be verified efficiently.

Definition 2.4.9 (Straight-line program)**.** Let G be a group and $S = \{s_1, \ldots, s_\ell\}$ a list of elements of G. A **straight-line program** in G from S to $g \in G$ is a sequence $P = (x_1, \ldots, x_m)$ of elements of G such that each x_k is either a member of S or a product of the form $x_i x_j$ for some $i, j < k$ or x_i^{-1} for some $i < k$. We say that the element x_m is given in terms of S by the straight-line program P.

The following is well known.

Proposition 2.4.10. *Let $G \leq S_n$ be a permutation group and S a set of generators of G. Given S, a presentation of G in terms of S can be computed in* poly(n) *time, where the relators returned are represented as straight-line programs.*

Proposition 2.4.11 (Homomorphism checking)**.** *Let $S \subseteq G$ be a list of generators of G. Assume a presentation of G is given in terms of S. Let $\varphi : S \to H$ be a function. Then φ extends to a homomorphism $\widetilde{\varphi} : G \to H$ if and only if the list $(\varphi(s) \mid s \in S)$ satisfies the relations of the presentation.*

Remark 2.4.12. Note that Proposition 2.4.11 gives an efficient way to check whether φ extends to a homomorphism assuming the relators are short or are given as short straight-line programs.

Corollary 2.4.13. *Let $G \leq S_n$ and $H \leq S_m$ be permutation groups. Let $S = \{a_1, \ldots, a_s\}$ be a set of generators of G and $f : S \to H$ a function. Whether f extends to a $G \to H$ homomorphism is testable in* $\mathrm{poly}(n, m)$ *time.*

2.4.4 Centralizers in S_n

Proposition 2.4.14. *Given $G \leq S_n$, its centralizer $C_{S_n}(G)$ in the full symmetric group can be found in polynomial time.*

Proof. Let $T = \{t_i\}_i$ denote the given set of generators for G. Without loss of generality, we may assume $|T| \leq 2n$ by Proposition 2.4.5 (a).

Construct the permutation graph $X = (V, E)$ of G, a colored graph on vertex set $V = [n]$ and edge set $E = \bigcup_{t \in T} E_t$, where $E_t = \{(i, i^t) : i \in [n]\}$ for each color $t \in T$. The edge set colored by $t \in T$ describes the permutation action of t on $[n]$. We see that $C_{S_n}(G) = \mathrm{Aut}(X)$, where automorphisms preserve color by definition.

If G is transitive (X is connected), then $C_{S_n}(G)$ is semiregular (all point stabilizers are the identity). For $i, j \in [n]$, it is possible in $\mathrm{poly}(n)$ time to decide whether there exists a permutation $\sigma \in \mathrm{Aut}(G) = C_{S_n}(G)$ satisfying $i^\sigma = j$ (takes i to j), then find the unique σ if it exists. To see this, build the permutation σ by setting $i^\sigma = j$, then following all colored edges from i and j in pairs to assign σ. If this is a well-defined assignment, then the permutation $\sigma \in \mathrm{Aut}(X)$ satisfying $i^\sigma = j$ exists.

In fact, if $X_1 = (V_1, E_1)$ and $X_2 = (V_2, E_2)$ are connected, whether then a graph isomorphism taking $i \in V_1$ to $j \in V_2$ can be found in $\mathrm{poly}(|V_1|)$ time if one exists.

If X is disconnected, collect the connected components of X by isomorphism type, so that there are m_i copies of the connected graph X_i in X, where $i = 1 \ldots \ell$ numbers the

isomorphism types. The components and multiplicities can be found in poly(n) time by finding the components of X (or, orbits of G, by Proposition 2.4.5 (d)) and pairwise checking for isomorphism. The automorphism group of X is

$$\mathrm{Aut}(X) = \mathrm{Aut}(X_1) \wr S_{m_1} \times \cdots \times \mathrm{Aut}(X_\ell) \wr S_{m_\ell}.$$

Each X_i is connected, so $\mathrm{Aut}(X_i)$ can be found as above. □

2.4.5 Blaha-Luks: enumerating coset representatives

We sketch the proof of the unpublished result by Blaha and Luks (Theorem 7.6.5), restated here for convenience. Below, by "coset" we mean "right coset."

Theorem 2.4.15 (Blaha–Luks)**.** *Given subgroups $K \leq L \leq S_n$, one can efficiently enumerate (at* poly(n) *cost per item) a representative of each coset of K in L.*

Let MoveCoset($M\sigma, i, j$) be a routine that decides whether there exists a permutation $\pi \in M\sigma$ satisfying $i^\pi = j$, and if so, finds one.

Proposition 2.4.16. MoveCoset *can be implemented in polynomial time.*

Proof. Answering MoveCoset is equivalent to finding $\pi \in M$ satisfying $i^\pi = j^{\sigma^{-1}}$ if one exists. This is the same as finding the orbits of M (Proposition 2.4.5 (d)). □

Definition 2.4.17 (Lexicographic ordering of S_n)**.** Let us encode the permutation $\pi \in S_n$ by the string $\pi(1)\pi(2)\cdots\pi(n)$ of length n over the alphabet $[n]$. Order permutations lexicographically by this code.

Note that the identity is the lex-first permutation in S_n.

Lemma 2.4.18. *Let $\sigma \in S_n$ and $K \leq S_n$. The algorithm* LexFirst *(below) finds the lex-first element of the subcoset $K\sigma \subseteq S_n$ in polynomial time.*

Algorithm 1 LexFirst within Subcoset

1: **procedure** LexFirst(subcoset $K\sigma$)
2: **for** $i \in [n]$ **do** $i^\pi \leftarrow$ `Null` ▶ `Initialize` $\pi : [n] \to [n] \cup \{$`Null`$\}$
3: **for** $s \in [n]$ **do** ▶ `Find smallest image of 1 under action by` $K\sigma$`, then iterate.`
4: **for** $t \in [n]$ **do** ▶ `Find smallest` s^π `possible by checking` $[n]$ `in order`
5: **if** MoveCoset$(K\sigma, s, t) =$ `True` **break**
6: **end for**
7: $s^\pi \leftarrow t$
8: $\tau \leftarrow$ MoveCoset$(K\sigma, s, t)$ ▶ `Restrict subcoset to elements moving` s `to` t
9: **end for**
10: **return** π
11: **end procedure**

It is straightforward to verify the correctness and efficiency of LexFirst. □

Proof of Theorem 2.4.15. Let $K \leq L \leq S_n$. Let S be a set of generators of L. The **Schreier graph** $\Gamma = \Gamma(K\backslash L, S)$ is the permutation graph of the L-action on the coset space $K\backslash L$, with respect to the set S of generators. Γ is a directed graph with vertex set $V = K\backslash L$ and edge set $E = \{(i, i^\pi) : i \in [n], \pi \in S\}$.

To prove Theorem 2.4.15, we may assume $|S| \leq 2n$, by Proposition 2.4.5(a). Use breadth-first search on Γ, constructing Γ along the way. Represent each vertex (a coset) by its lexicographic leader. Then, store the discovered vertices, ordered lexicographically, in a balanced dynamic search tree such as a red-black tree. Note that the tree will have $O(\log(n!)) = O(n \log n)$ depth and every vertex of Γ has at most $2n$ out-neighbors. Hence, the incremental cost is poly(n). □

CHAPTER 3

LIST-DECODING, GENERAL CONCEPTS

3.1 Terminology for general codes

3.1.1 List-decoding

We introduce some terminology that applies to codes in general and not just homomorphism codes. There will be some overlap with Section 2.2.3.

Let Σ be an alphabet and Ω a set we think of as the set of positions. We view Σ^Ω, the set of $\Omega \to \Sigma$ functions, as our code space; we call its elements "words." We write $\mathrm{dist}(u, w)$ for the **normalized Hamming distance** between two words $u, w \in \Sigma^\Omega$ (so $0 \le \mathrm{dist}(u, w) \le 1$) and refer to it simply as "distance." Let $\mathcal{C} \subset \Sigma^\Omega$ be a code; we call its elements "codewords."

Definition 3.1.1 (Minimum distance)**.** We write $\mathsf{mindist}(\mathcal{C})$ (or simply $\mathsf{mindist}$) for the minimum distance between distinct codewords in $\mathcal{C}$.

Words we wish to decode are referred to in the literature as "received words."

Definition 3.1.2 (The list $\mathcal{L}^r(\mathcal{C}, f, \rho)$)**.** We refer to the set of codewords within a specified distance ρ of a received word $f \in \Sigma^\Omega$ as **"the list"** and denote it by $\mathcal{L}^r = \mathcal{L}^r(\mathcal{C}, f, \rho)$.

Definition 3.1.3 (List-size bound)**.** We write $\ell^r(\mathcal{C}, \rho) := \max_f |\mathcal{L}^r(\mathcal{C}, f, \rho)|$.

We will later define lists $\mathcal{L}$ and list-size bounds ℓ in terms of agreement instead of distance.

3.1.2 Combinatorial list-decoding

The list-decoding problem splits into a combinatorial and an algorithmic part.

The combinatorial problem, to which we refer as *combinatorial list-decoding*, asks to bound the size of the list. Typically, we take $\rho = (\mathsf{mindist} - \varepsilon)$ and we wish to obtain a bound $\ell(\mathcal{C}, \rho) \le c(\varepsilon)$, that depends only on ε and the code $\mathcal{C}$.

We say that $\mathcal{C}$ is a **CombEcon** ("combinatorially economically list-decodable") code if $c(\varepsilon) = \text{poly}(1/\varepsilon)$. (Naturally, this concept applies to classes of codes, not a single code.)

3.1.3 Algorithmic list-decoding

A **list-decoder** is an algorithm that, given the received word $f \in \Sigma^\Omega$ and the distance ρ, lists a superset $\widetilde{\mathcal{L}}$ of the list $\mathcal{L} = \mathcal{L}^r(\mathcal{C}, f, \rho)$. Typically, we take $\rho = (\mathsf{mindist} - \varepsilon)$ and we wish to produce a list of size $|\widetilde{\mathcal{L}}| \leq \widetilde{c}(\varepsilon)$.

A **local algorithm** is a probabilistic algorithm that has only oracle access to the received word f.

We say that $\mathcal{C}$ is an **AlgEcon** ("algorithmically economically list-decodable") code if there exists a local list-decoder that shows the following features.

Input: $\mathsf{mindist}$, $\epsilon > 0$, oracle access to $f \in \Sigma^\Omega$.

Notation: $\mathcal{L} = \mathcal{L}^r(\mathcal{C}, f, \mathsf{mindist} - \varepsilon)$.

Output: A list $\widetilde{\mathcal{L}}$ of codewords in $\mathcal{C}$ of length $|\widetilde{\mathcal{L}}| = \text{poly}(1/\varepsilon)$.

Guarantee: With probability $\geq 3/4$, we have $\widetilde{\mathcal{L}} \supseteq \mathcal{L}$.

Cost:

(i) $\text{poly}(\log|\Omega|, 1/\varepsilon)$ queries to the received word f.

(ii) $\text{poly}(\log|\Omega|, \log|\Sigma|, 1/\varepsilon)$ amount of work.

Access: The meaning of this definition depends also on the access model to Σ and Ω. We shall clarify this in each application.

Strong AlgEcon

In the **unit cost model** for Σ, we charge unit cost to name an element of Σ.

We say that $\mathcal{C}$ is a **strong AlgEcon** code if there exists a list-decoder satisfying the conditions of AlgEcon, except with (ii) replaced by the following.

(ii') $\mathrm{poly}(\log|\Omega|, 1/\varepsilon)$ amount of work in the unit cost model for Σ.

Typically, elements of Σ are encoded by strings of length $\log|\Sigma|$ and therefore (ii') implies (ii) with linear dependence on $\log|\Sigma|$. The AlgEcon results proved in prior work [DGKS08, GS14, BGSW18] are actually strong AlgEcon results for those classes of pairs of groups. Our AlgEcon result for alternating domain does not meet the "strong" requirement.

3.1.4 Certificate list-decoding

In the light of technical difficulties arising from algorithmic list-decoding, we introduce a new type of list-decoding that is intermediate between the combinatorial and algorithmic. We call it "certificate list-decoding." We shall refer to results of this type as "semi-algorithmic."

A **partial map** γ from Ω to Σ, denoted $\gamma\colon \Omega \rightharpoonup \Sigma$, is a map of a subset of Ω to Σ. In particular, $\operatorname{dom}\gamma \subseteq \Omega$.

Definition 3.1.4 (Certificate)**.** We say that a partial map $\gamma\colon \Omega \rightharpoonup \Sigma$ is a **certificate for the codeword** $\varphi \in \mathcal{C}$ if $\gamma = \varphi_{|\operatorname{dom}\gamma}$ and φ is the unique codeword that extends γ. A **certificate** for the code $\mathcal{C}$ is a certificate for some codeword in $\mathcal{C}$.

Definition 3.1.5 (Certificate-list)**.** We say that a list Γ of $\Omega \rightharpoonup \Sigma$ partial maps is a **certificate-list** for the set $\mathcal{K} \subset \mathcal{C}$ of codewords if Γ contains a certificate for each codeword in $\mathcal{K}$. A **certificate-list** for $\mathcal{C}$ **up to distance** ρ of the received word $f : \Omega \to \Sigma$ is a certificate-list for the list $\mathcal{L} = \mathcal{L}^r(\mathcal{C}, f, \rho)$.

Remark 3.1.6. Note that we permit the certificate-list Γ to contain redundancies (more than one certificate for the same codeword) and irrelevant items (partial functions that are not certificates of any codeword in $\mathcal{K}$, or not even certificates of any codeword at all).

Definition 3.1.7. A **certificate-list-decoder** is an algorithm that, given the received word $f \in \Sigma^\Omega$ and the distance ρ, constructs a certificate-list of $\mathcal{C}$ up to distance ρ of f.

Definition 3.1.8. We say that $\mathcal{C}$ is a **CertEcon** ("certificate-econonomically list-decodable") code if there exists a local certificate-list-decoder that shows the following features.

Input: $\varepsilon > 0$, oracle access to $f \in \Sigma^{\Omega}$.

Notation: Again, let $\mathcal{L} = \mathcal{L}^r(\mathcal{C}, f, \mathsf{mindist} - \varepsilon)$.

Output: A list Γ of $\Omega \rightharpoonup \Sigma$ partial maps of length $|\Gamma| = \mathrm{poly}(1/\varepsilon)$.

Guarantee: With probability $\geq 3/4$, we have that Γ is a certificate-list for $\mathcal{L}$.

Cost:

(i) $\mathrm{poly}(\log|\Omega|, 1/\varepsilon)$ queries to the received word f.

(ii) $\mathrm{poly}(\log|\Omega|, \log|\Sigma|, 1/\varepsilon)$ amount of work.

Access: The meaning of this definition depends also on the access model to Σ and Ω. We shall clarify this in each application.

Remark 3.1.9. Note that mindist is not part of the input. We are likely to find a certificate of $\mathcal{C}$ up to distance $(\mathsf{mindist} - \varepsilon)$ of the received word f, regardless of the actual value of mindist.

Remark 3.1.10. CertEcon is intermediate between AlgEcon and CombEcon. Indeed, CertEcon implies CombEcon, by the length bound of the Output and the Guarantee. Moreover, AlgEcon implies CertEcon, as the AlgEcon Output $\widetilde{\mathcal{L}}$ satisfies the definition of a certificate, under the same Guarantee and Cost bound.

Strong CertEcon

Definition 3.1.11. We say that $\mathcal{C}$ is a **strong CertEcon** code if there exists a certificate-list-decoder satisfying the conditions of CertEcon, except with (ii) replaced by the following.

(ii') $\mathrm{poly}(\log|\Omega|, 1/\varepsilon)$ amount of work in the unit cost model for Σ.

All CertEcon results in this thesis are actually strong CertEcon results.

Remark 3.1.12. Strong CertEcon does not follow from AlgEcon, though it does follow from strong AlgEcon.

Remark 3.1.13. As in the AlgEcon context, the unit cost model can also be used in the case of infinite Σ. In fact, all our CertEcon results hold for infinite codomain in the unit cost model, i.e., they satisfy (ii").

3.1.5 Subword extension

In this section we formalize our strategy to advance from certificate-list-decoding to algorithmic list-decoding (Observation 3.1.15 below).

Definition 3.1.14 (Subword extension problem)**.** Let $\mathcal{C}$ be a code. The **subword extension problem** asks, given a partial map $\gamma : \Omega \rightharpoonup \Sigma$, whether γ extends to a codeword in $\mathcal{C}$.

A **subword extender** is an algorithm that answers this question and returns a codeword in $\mathcal{C}$ extending γ, if one exists.

Observation 3.1.15. *A certificate-list-decoder and a subword extender combine to a list-decoder.*

Remark 3.1.16. This observation describes our two-phase plan to prove algorithmic list-decodability results for homomorphism codes with alternating domains. In the case of homomorphism codes, the subword extension problem corresponds to the *homomorphism extension problem* (see Section 3.2.5). The algorithmic difficulty of the homomorphism extension problem is a major bottleneck to further progress.

In fact, the plan suggested by this observation is too ambitious. We have no hope in solving the subword extension problem in cases of interest for *all* subwords. Therefore, we relax the subword extender concept; correspondingly, we strengthen the notion of certificates required.

Definition 3.1.17 ($\mathcal{W}$-subword extender)**.** The **$\mathcal{W}$-subword extension problem** asks to solve the subword extension problem on inputs from $\mathcal{W}$. A **$\mathcal{W}$-subword extender** is an algorithm $\mathscr{A}$ that takes as input any partial map $\gamma : \Omega \rightharpoonup \Sigma$ and returns a yes/no answer; and in the case of a "yes" answer, it also returns a codeword $\mathscr{A}(\gamma) \in \mathcal{C}$, such that

- if $\gamma \in \mathcal{W}$ then the answer is "yes" if and only if γ extends to a codeword, and in this case, $\mathscr{A}(\gamma)$ is a codeword that extends γ.

Remark 3.1.18. Note that $\mathscr{A}$ is not required to decide whether $\gamma \in \mathcal{W}$. $\mathcal{A}$ must correctly decide extendability of γ for all $\gamma \in \mathcal{W}$; in case $\gamma \notin \mathcal{W}$, the algorithm may return an arbitrary answer.

Let $\mathcal{W}$ be a set of $\Omega \rightharpoonup \Sigma$ partial maps.

Definition 3.1.19 ($\mathcal{W}$-certificate)**.** A **$\mathcal{W}$-certificate** is a certificate that belongs to $\mathcal{W}$.

Definition 3.1.20 ($\mathcal{W}$-certificate-list)**.** We say that a list Γ of $\Omega \rightharpoonup \Sigma$ partial maps is a **$\mathcal{W}$–certificate-list** for the set $\mathcal{K} \subset \mathcal{C}$ of codewords if Γ contains a $\mathcal{W}$-certificate for each codeword in $\mathcal{K}$. A **$\mathcal{W}$-certificate-list** for $\mathcal{C}$ **up to distance** ρ of the received word $f : \Omega \to \Sigma$ is a $\mathcal{W}$-certificate-list for the list $\mathcal{L} = \mathcal{L}(\mathcal{C}, f, \rho)$.

Remark 3.1.21. Note that, as mentioned in Remark 3.1.6, we permit the $\mathcal{W}$-certificate-list Γ to contain redundancies and irrelevant items, including partial functions γ that do not belong to $\mathcal{W}$.

Definition 3.1.22. A $\mathcal{W}$-certificate-list-decoder is an algorithm that, given the received word $f \in \Sigma^{\Omega}$ and the distance ρ, constructs a $\mathcal{W}$-certificate-list of $\mathcal{C}$ up to distance ρ of f.

Our overall strategy for the case when G is "far from abelian" is summarized in the following observation.

Observation 3.1.23. *For any set $\mathcal{W}$ of $\Omega \rightharpoonup \Sigma$ partial maps, a $\mathcal{W}$-certificate-list-decoder and a $\mathcal{W}$-subword extender combine to a list-decoder.*

Definition 3.1.24. We say that $\mathcal{C}$ is a **$\mathcal{W}$-CertEcon** ("$\mathcal{W}$-certificate-econonomically list-decodable") code if there exists a local $\mathcal{W}$-certificate-list-decoder that shows the features listed in Definition 3.1.8.

Definition 3.1.25. We say that $\mathcal{C}$ is a **strong $\mathcal{W}$-CertEcon** code if there exists a strong $\mathcal{W}$-certificate-list-decoder, i. e., a $\mathcal{W}$-certificate-list-decoder that is a strong certificate-list-decoder (see Definition 3.1.11).

3.1.6 Minimum distance versus maximum agreement

Recall that our code space is Σ^{Ω}, the set of $\Omega \to \Sigma$ functions. In the theory of error-correcting codes, the usual measure of distance between two functions (strings) is the (relative) Hamming distance, the fraction of symbols on which they differ. Following [GKS06], we find it convenient to consider the measure complementary to normalized Hamming distance, the (relative) **agreement**,

$$\operatorname{agr}(\varphi, \psi) := \frac{1}{|\Omega|} |\{\omega \in \Omega \mid \varphi(\omega) = \psi(\omega)\}|, \tag{3.1}$$

the fraction of positions on which the two functions $\varphi, \psi : \Omega \to \Sigma$ agree.

Definition 3.1.26. The **maximum agreement** of the code $\mathcal{C}$ is given by

$$\Lambda_{\mathcal{C}} := \max_{\substack{\varphi,\psi \in \mathcal{C} \\ \varphi \neq \psi}} \operatorname{agr}(\varphi, \psi).$$

Fact 3.1.27. The minimum distance is the complement of the maximum agreement, i.e.,

$$\mathsf{mindist} = 1 - \Lambda_{\mathcal{C}}.$$

So, the codewords within distance $(\mathsf{mindist} - \varepsilon)$ of a received word f are the same as the codewords with agreement at least $\Lambda_{\mathcal{C}} + \varepsilon$ with f. For large classes of homomorphism

codes, we will provide evidence for the infeasibility of list-decoding outside this range (see Section 5.3.3), i. e., the list-decoding radius is mindist for those classes.

For the remainder of this thesis, we will use the notation $\mathcal{L}$ for lists and ℓ for list-size bounds, defined in terms of agreement instead of distance.

Definition 3.1.28 (The list $\mathcal{L}(\mathcal{C}, f, \lambda)$)**.** The set of codewords that have specified agreement λ with the received word $f \in \Sigma^\Omega$ is denoted by $\mathcal{L} = \mathcal{L}(\mathcal{C}, f, \lambda)$.

Definition 3.1.29 (List-size bound)**.** We write $\ell(\mathcal{C}, \lambda) := \max_f |\mathcal{L}(\mathcal{C}, f, \lambda)|$.

We observe that $\mathcal{L}(\mathcal{C}, f, \lambda) = \mathcal{L}^r(\mathcal{C}, f, 1-\lambda)$ and $\ell(\mathcal{C}, \lambda) = \ell^r(\mathcal{C}, 1-\lambda)$.

3.2 Formal statements for homomorphism codes

3.2.1 List-decoding homomorphism codes

In the previous section we defined concepts for (local) list-decoding codes in general. Our main objects of study for list-decoding are homomorphism codes $\mathcal{C} = \mathrm{aHom}(G, H)$. In this context, the domain Ω is a finite group G and the codomain Σ is a group H. The code space Σ^Ω is H^G. The maximum agreement will be denoted by $\Lambda_{G,H} = \Lambda_{\mathrm{aHom}(G,H)}$, i.e.,

$$\Lambda_{G,H} := \max_{\substack{\varphi,\psi \in \mathrm{aHom}(G,H) \\ \varphi \neq \psi}} \mathrm{agr}(\varphi, \psi). \tag{3.2}$$

We note that $\Lambda_{G,H} = \Lambda_{\mathrm{Hom}(G,H)}$ whenever $|\mathrm{Hom}(G, H)| > 1$ [Guo15] (see Proposition 2.2.8). In other words, the maximum agreement between affine homomorphisms is the same as the maximum agreement between homomorphisms.

Let $\mathfrak{D}$ be a class of pairs (G, H) of groups. We say that $\mathfrak{D}$ is CombEcon if the class $\{\mathrm{aHom}(G, H) \mid (G, H) \in \mathfrak{D}\}$ of codes is CombEcon. We define CertEcon and AlgEcon classes of homomorphism codes analogously.

Denote by $\mathfrak{Groups}$ the class of all groups, finite or infinite. We say that a class $\mathfrak{G}$ of finite groups is **universally CombEcon** if $\mathfrak{G} \times \mathfrak{Groups}$ is CombEcon. We define universally CertEcon and universally AlgEcon analogously, under access models to be specified.

A common feature of prior work [GL89, GKS06, DGKS08, GS14] (reviewed in Section 1.1) is the CombEcon and AlgEcon list-decodability of the homomorphism codes considered.

All previously existing results put structural restrictions both on the domain and codomain. In particular, they were restricted to subclasses of the solvable groups. In this thesis we extend the economical list-decodability (both combinatorial and algorithmic) in the following three directions.

1. We give a general principle for removing certain types of constraints on the domain (see Section 4.1.2). It will follow that the previously known results extend to arbitrary domains.

2. We find universally economically list-decodable classes of groups (i.e., arbitrary codomain). Specifically, alternating groups are universally CombEcon. Moreover, alternating groups are universally CertEcon, under modest assumptions.

3. We exhibit the first (nontrivial) examples where the domain is not solvable.

We note that no CombEcon bounds appear to be known for the much-studied classical linear codes (Reed–Solomon, Reed–Muller, BCH) (cf., e.g., [BL15]). The poly$(1/\varepsilon)$ CombEcon bound for Hadamard codes is quadratic [GL89]. For abelian and nilpotent groups, it currently has degree 105 [DGKS08, GS14].

3.2.2 Shallow random generation and list-decodability

We shall consider groups with the property that a bounded number of random elements tend to generate a subgroup of bounded depth (see Definitions 3.2.4 and 3.2.5 below). This class

includes the alternating groups. We show that groups in this class are CombEcon, and under minimal assumptions on access they are also CertEcon.

It will be useful to consider an H-independent lower bound on the quantity $\Lambda_{G,H}$.

Definition 3.2.1. We define $\Lambda^*_G = \min\{\Lambda_{G,H} : \Lambda_{G,H} \neq 0, H \in \mathfrak{Groups}\}$.

Observation 3.2.2. *For simple groups the following three quantities are equal: (a) Λ^*_G, (b) $\Lambda_{G,G}$, and (c) the largest fraction of elements of G fixed by an automorphism.*

Observation 3.2.3. *For $G = A_n$, $n \geq 5$, we have $\Lambda^*_G = 1/\binom{n}{2}$.*

The **depth** of a subgroup M in a group G is the length d of the longest subgroup chain $M = M_0 < M_1 < \cdots < M_d = G$. We say that a subgroup is "shallow" if its depth is bounded. It follows from a result of [Bab89] that already a pair of elements in A_n generates a subgroup of depth at most 6. This is the property that we generalize.

Definition 3.2.4 (Shallow random generation)**.** Let $k, d \in \mathbb{N}$. We say that a finite group G is (k,d)**-shallow generating** if

$$\Pr_{g_1,\ldots,g_k \in G}[\mathrm{depth}(\langle g_1, \ldots, g_k \rangle) > d] < (\Lambda^*_G)^k. \tag{3.3}$$

Definition 3.2.5 (SRG groups)**.** We say that a class $\mathfrak{G}$ of finite groups has **shallow random generation** ($\mathfrak{G}$ is SRG) if there exist $k, d \in \mathbb{N}$ such that all $G \in \mathfrak{G}$ are (k,d)-shallow generating.

Lemma 3.2.6. *The alternating groups are SRG groups. In particular, for sufficiently large n, the alternating group A_n is $(2,6)$-shallow generating.*

We prove this lemma in Section 6.2.1. We note that certain classes of Lie type simple groups are also SRG. We shall elaborate on this in a separate paper.

Now we can state one of our main results.

Theorem 3.2.7. *If G is an SRG group, then G is universally CombEcon list-decodable.*

For the case of alternating groups, we show that the degree of the poly$(1/\epsilon)$ list-size bound is at most 9; with further work this can be reduced to 7.

Theorem 3.2.8. *If G is an SRG group, then G is universally strong CertEcon list-decodable.*

In fact, SRG groups are universally strong $\mathcal{W}_{\Lambda_{G,H}}$-CertEcon list-decodable (see Section 3.1.5 for the definition of $\mathcal{W}$-certificates). This restriction on the type of certificates we obtain is necessary for extensions to AlgEcon results (cf. comment before Definition 3.1.17). Section 3.2.3 discusses $\mathcal{W}$-certificates in the context of homomorphism codes. A formal statement of the $\mathcal{W}_{\Lambda_{G,H}}$-CertEcon result is given in Section 3.2.4.

Access model. For the CertEcon results, we assume access to (nearly) uniform random elements of the domain. We do not multiply elements of the domain, so we do not need black-box access to the domain. However, representing the domain as a black-box group suffices for random generation [Bab91].

We need no access to the codomain.

Pointers. We prove the CombEcon result in Section 6.2.4 and the CertEcon result in Section 6.2.5. For alternating groups we also give another, non-algorithmic, proof of the CombEcon result in Section 5.3. That proof relies on a generic sphere packing argument to split the sphere into more tractable bins (see Lemma 5.2.5 and Section 5.2.4).

3.2.3 Certificate list-decoding for homomorphism codes

First we translate the concepts associated with certificate list-decoding (Section 3.1.4) to the context of homomorphism codes. A **certificate** γ is a $G \rightharpoonup H$ partial map that extends uniquely to an affine homomorphism $\varphi \in \mathrm{aHom}(G, H)$.

A **subword extender** is an algorithm that extends a $G \rightharpoonup H$ partial map to a full homomorphism if possible.

For a subset $S \subseteq G$, we denote by $\mu_G(S) := |S|/|G|$ the **density** of S in G. We often write $\mu(S)$ if G is understood. For notational simplicity, we write Λ for $\Lambda_{G,H}$.

Notation 3.2.9. Let $\mathcal{W}_\lambda$ be the set of $G \rightharpoonup H$ partial maps γ such that $\mu(\langle \operatorname{dom} \gamma \rangle) > \lambda$, i.e., the subgroup generated by the domain of γ has density greater than λ.

Recall that we have introduced certificate list-decoding as an intermediate step towards algorithmic list-decoding, to address technical difficulties that arise in algorithmic list-decoding in the alternating case. Our plan is to apply the subword extension strategy of Section 3.1.5 (specifically, Observation 3.1.23 with $\mathcal{W} = \mathcal{W}_\Lambda$).

Observation 3.2.10. *If a partial map $\gamma \colon G \rightharpoonup H$ belongs to $\mathcal{W}_\Lambda$, then γ extends to at most one affine homomorphism in* $\operatorname{aHom}(G, H)$.

We will find $\mathcal{W}_\Lambda$-certificate-list-decoders for a large class of homomorphism codes, and we wish to find corresponding $\mathcal{W}_\Lambda$-subword-extenders.

Let γ be a $G \rightharpoonup H$ partial map. We present three conditions on γ, then discuss their relationships with each other as well as to list-decoding.

(1) If γ extends to an affine homomorphism in $\operatorname{aHom}(G, H)$, then the extension is unique, i.e., γ is a certificate for some affine homomorphism.

(2) $\mu(\langle \operatorname{dom} \gamma \rangle) > \Lambda$.

(3) The domain $\operatorname{dom} \gamma$ generates G.

Clearly, Condition (3) implies Condition (2), which implies Condition (1). Implications in the other direction do not hold in general. In particular, neither reverse implication holds for the alternating groups.

Algorithmic list-decoding requires a list of full homomorphisms, typically represented as partial maps satisfying Condition (3).

Certificate list-decoding requires the list of partial maps to satisfy Condition (1). Our CertEcon algorithms actually return certificates satisfying Condition (2), i.e., they are $\mathcal{W}$-certificate-list-decoders where $\mathcal{W}$ contains certificates satisfying Condition (2).

In the case of abelian G, Condition (3) is equivalent to Condition (1) if the irrevelant kernel is trivial (see Definition 4.3.1). So, in this case $\mathcal{W}$-certificate list-decoding and algorithmic list-decoding are equivalent. We introduced the mean-list-decoding machinery to address the case of nontrivial irrelevant kernel (see Theorems 4.2.2 and 4.3.9).

3.2.4 Certificate list-decoding: SRG → arbitrary

Recall that, in the context of list-decoding $\mathrm{aHom}(G, H)$, $\mathcal{W}_\Lambda$ denotes the set of $G \rightharpoonup H$ partial maps γ such that $\mu(\langle \mathrm{dom}\, \gamma \rangle) > \Lambda$, where $\Lambda = \Lambda_{G,H}$. We state the promised strengthening of Theorem 3.2.8.

Theorem 3.2.11 (SRG certificate, abridged)**.** *If G is an SRG group, then G is universally strong $\mathcal{W}_\Lambda$-CertEcon list-decodable.*

Access model. We assume access to (nearly) uniform random elements of the domain. We do not multiply elements of the domain. We remark that representing the domain as a black-box group would suffice for random generation [Bab91].

We need no access to the codomain. We get ahold of elements of the codomain by querying the received word. We shall not perform any group operations in the codomain.

Actually our result is much stronger than what would be implied by our definition of CertEcon.

Theorem 3.2.12 (SRG certificate, unabridged)**.** *Let G be a (k, d)-shallow generating group and H an arbitrary group. We have a local algorithm that shows the following features.*

Input: *Access to G. Values $\varepsilon, \eta > 0$.*

Output: *A set $\Pi \subset G^{k+d+1}$ of $(k+d+1)$-tuples in G, where*

$$|\Pi| = \left\lceil \frac{1}{\varepsilon^{k+d+1}} \ln\left(\frac{1}{\eta\varepsilon^{k+d+1}}\right) \right\rceil.$$

Cost: $\mathrm{poly}(1/\varepsilon, \ln(1/\eta))$ *amount of work.*

Performance guarantee: *For every received word $f \in H^G$, with probability at least $(1-\eta)$, the set $\Gamma := \{f|_R : R \in \Pi\}$ is $\mathcal{W}_{\Lambda_{G,H}}$-certificate-list for $\mathrm{aHom}(G,H)$ up to distance $(\mathsf{mindist} - \varepsilon)$ of f.*

Access model. Same as in Theorem 3.2.11.

Pointer. The proof of Theorems 3.2.11 and 3.2.12 can be found in Sections 6.2.5.

Remark 3.2.13. Given that A_n is $(2,6)$-shallow generating (Lemma 3.2.6), Theorems 3.2.11 and 3.2.12 applies to A_n with $k+d+1 = 9$. We think of A_n as being given in its natural permutation representation. We note that a representation of A_n as a black-box group would suffice, because the natural permutation representation of an alternating group can be efficiently extracted from a black-box group representation [BLGN$^+$05].

3.2.5 Algorithmic list-decoding: alternating → symmetric, restricted cases

We follow the strategy of Section 3.1.5 to combine a subword extension result with the universally CertEcon result (Theorem 3.2.11) to achieve a universally AlgEcon result.

We will define the Homomorphism Extension Problem, which is exactly the subword extension problem for homomorphisms $\mathrm{Hom}(G,H)$. (It can also be used to solve subword extension for $\mathrm{aHom}(G,H)$, see Section 6.3.3.) Cases of Homomorphism Extension will be solved in Chapter 7. The most pertinent result is as follows (see Chapter 7 for more details).

Theorem 3.2.14. *Let $G = A_n$ and $H = S_m$, both represented as permutation groups. If (1) $m < 2^{n-1}/\sqrt{n}$ and (2) $\gamma : G \rightharpoonup H$ satisfies $|G : \langle \mathrm{dom}\,\gamma \rangle| = \mathrm{poly}(n)$, then Homomorphism Extension Search can be solved in $\mathrm{poly}(n, m)$ time.*

We need one more ingredient before we can prove our main algorithmic result.

Lemma 3.2.15. *Let $n \geq 10$. Let $G = A_n$ and let H be a group. If $|\mathrm{Hom}(G, H)| > 1$, then $\Lambda_{G,H} \geq 1/\binom{n}{2}$. In particular, either $\Lambda_{G,H} = 1/\binom{n}{2}$ or $\Lambda_{G,H} = 1/n$.*

Proof. Since $\mathrm{Hom}(G, H)$ is nontrivial and A_n is simple, H contains an isomorphic copy of A_n. An automorphism of A_n fixes a subgroup of index at least $\binom{n}{2}$, so $\Lambda \geq \Lambda_{G,G} = 1/\binom{n}{2}$. By Fact 2.2.5, $1/\Lambda$ must be the index of a proper subgroup. By the Jordan-Liebeck Theorem (Theorem 2.4.2), the only candidates are n and $\binom{n}{2}$, so $\Lambda = 1/\binom{n}{2}$ or $1/n$. □

We have now stated all the ingredients needed for the AlgEcon result for alternating domains.

Theorem 3.2.16 (Alternating algorithmic result)**.** *If G is an alternating group A_n and H is a symmetric group S_m, then* $\mathrm{aHom}(G, H)$ *is AlgEcon, assuming $m < 2^{n-1}/\sqrt{n}$.*

Access model. We assume both A_n and S_m are given in their natural permutation representations.

Proof. The proof follows the "CertEcon with HomExt implies AlgEcon" approach discussed in Section 3.1.5.

Lemma 3.2.6 shows that alternating groups are SRG groups, which are universally $\mathcal{W}_\Lambda$-CertEcon by Theorem 3.2.11. Theorem 6.3.10 shows that $\textsc{HomExt}_{1/\binom{n}{2}}(G, H)$ can be solved in $\mathrm{poly}(n, m)$ time, under the assumed restrictions on the codomain H. But, $1/\binom{n}{2} \leq \Lambda_{G,H}$ by Lemma 3.2.15. Corollary 6.3.9 turns this into a $\mathrm{poly}(n, m)$-time subword extender, which combines with the $\mathcal{W}_\Lambda$-CertEcon claim (Observation 3.1.23) to give the AlgEcon claim. □

For more details, see Section 6.3.2.

CHAPTER 4

RELAXATION PRINCIPLES AND MEAN-LIST-DECODING

4.1 Introduction

We introduce a new "add-on" method to list-decoding, which extends the classes of codes for which we conclude economical list-decodability. The extension allows for repeated codes, or, translated homomorphism codes.

To be more precise, this method will show the equivalence of the "economical" concepts for a general class of codes $\mathcal{C}$ and the corresponding class of **repeated codes** $\mathcal{C} * r$ for every $r \in \mathbb{N}$ (see Notation 4.2.6 and Theorem 4.2.2). We will identify repeated codes with **mean-lists** (see Definition 4.2.1), lists required to possess certain agreement with a *family* of received words, instead of one received word. Additionally, this method will show, specifically for homomorphism codes $\mathrm{Hom}(G, H)$, the equivalence of "economical" for **translated codes** $\mathrm{aHom}(G, H)$.

The most notable consequence of this method is a relaxation principle on the domain.

Towards proving this relaxation principle, we will define for groups G and H the (G, H)-**irrelevant normal kernel** N as the intersection of all kernels of $G \to H$ homomorphisms. We will see that list-decoding $\mathrm{aHom}(G, H)$ is equivalent to mean-list-decoding $\mathrm{aHom}(G/N, H)$.

4.1.1 Structure of chapter

In Section 4.3.1 we contextualize the domain relaxation principle. In Section 4.1.3, we state and prove the Bipartite Covering Lemma, a combinatorial tool central to the results in this chapter.

Section 4.2 presents mean-list-decoding principles on general codes. Section 4.2.1 defines mean-list-decoding and relevant economical concepts. Section 4.2.2 proves that economical

mean-list-decoding is not much harder than economical list-decoding.

Section 4.3 discusses the implications of mean-list-decoding and the Bipartite Covering Lemma specifically in the context of homomorphism codes. Section 4.3.1 defines the irrelevant kernel and explains how the irrelevant kernel is irrelevant in the context of economical list-decoding. Section 4.3.2 proves this by identifying lists in $\mathrm{aHom}(G, H)$ with mean-lists in $\mathrm{aHom}(G/N, H)$, where N is the (G, H)-irrelevant kernel. Section 4.3.3 applies the Bipartite Covering Lemma to show that economically list-decoding aHom is not much harder than Hom.

4.1.2 Domain relaxation principle

We discuss context and motivation for the domain relaxation principle.

In all prior work [DGKS08, GS14], both the domain and the codomain were abelian or close to abelian (nilpotent or supersolvable). It is natural to ask how to further relax the structural constraints on the groups involved.

We point out that structural constraints such as nilpotence or solvability (or any other hereditary property) play a very different role if imposed on the domain versus the codomain. For instance, a combinatorial list-decoding bound on {abelian $\to$ abelian} homomorphism codes implies the same bound for {arbitrary $\to$ abelian} homomorphism codes. This is shown by reducing the question on $\mathrm{aHom}(G, H)$ for arbitrary G and abelian H to $\mathrm{aHom}(G/G', H)$, where G' is the commutator subgroup of G, so G/G' is the largest abelian quotient of G. A similar argument extends the bounds for {nilpotent $\to$ nilpotent} homomorphism codes to {arbitrary $\to$ nilpotent}, working through the largest nilpotent quotient of G. Similar results hold for certificate and algorithmic list-decoding.

In general, we can replace G by its *relevant quotient* G/N, where N is the (G, H)-irrelevant kernel (see Definition 4.3.1).

While this observation extends the reach of the results of Dinur et al. [DGKS08] and

Guo and Sudan [GS14], it also shows that, in a sense, the gains by extending the class of groups serving as the domains, without relaxing the structural constraints on the codomains, is *virtual*, and the main impediment to extending these results to wider classes of pairs of groups is the structural constraints on the *codomain*.

4.1.3 Bipartite covering lemma

We describe a simple combinatorial lemma (Lemma 4.1.1) that will be used in two contexts in this chapter (mean-list-decoding with application to the domain relaxation principle; equivalence of efficiency of list-decoding Hom and aHom). It describes how we cover the list in the extension code by few lists in the original code. Moreover, it describes how to find a list-decoder for an extension class, using few calls to the list-decoder for the original class.

To prove the mean-list result (Theorem 4.2.2), the extension code remains the same, but the extension lists are mean-lists (see Definition 4.2.1) taking input a family of received words instead of only one received word. For aHom versus Hom lists (Corollary 4.3.19), the extension code is $\mathrm{aHom}(G, H)$, the original code is $\mathrm{Hom}(G, H)$, and the received word is the same.

We write $X = (V, W; E)$ to denote a bipartite graph with given vertex partition (V, W) (all edges go between V and W). We denote the set of neighbors of a vertex u by $N(u)$.

Lemma 4.1.1 (Bipartite covering lemma)**.** *Let $\eta, \delta > 0$. Let $X = (V, W; E)$ be a bipartite graph. Suppose that $\deg(v) \leq L$ for all $v \in V$ and $\deg(w) \geq \delta|V|$ for all $w \in W$. Then the following hold.*

(1) $|W| \leq L/\delta$.

(2) Set $s = \left\lceil \frac{4}{3\delta}(\ln(L/(\eta\delta)) \right\rceil$. *Choose a sequence* $(u_1, \ldots, u_s) \in V^s$ *uniformly at random. Create a set* $U \subseteq V$ *by independently including each* u_i *with probability* $3/4$. *Then with probability* $\geq (1 - \eta)$, *we have* $W = \bigcup_{u \in U} N(u)$.

Conclusion (1) will be used to prove CombEcon results, whereas (2) will be used to prove CertEcon and AlgEcon results.

Proof. (1) Count edges two ways.

$$L \cdot |V| \geq \sum_{v \in V} \deg(v) = \sum_{w \in W} \deg(w) \geq \delta |W||V|.$$

So, $W \leq L/\delta$.

(2) Let $s = \lceil \frac{4}{3\delta}(\ln(L) + \ln(1/\eta\delta)) \rceil$. Choose $u_1, \ldots, u_s \in V$ independently and uniformly at random. Choose $\hat{u}_1, \ldots, \hat{u}_s \in V \cup \{\star\}$ independently as follows. For each $i = 1, \ldots, s$, let $\hat{u}_i$ be u_i with probability $3/4$ and $\star$ otherwise. For notational consistency, define the neighbor set $N(\star)$ of $\star$ by $N(\star) = \emptyset$.

Fix $w \in W$. We have $\Pr_{v \in V}(w \in N(v)) \geq \delta$ by assumption. So, for each i, $\Pr_{\hat{u}_i}(w \in N(\hat{u}_i)) = \frac{3}{4} \cdot \Pr_{u_i}(w \in N(\hat{u}_i)) \geq \frac{3}{4}\delta$. Since the $\hat{u}_i$ were chosen independently,

$$\Pr\left(w \notin \bigcup_{i=1}^{s} N(\hat{u}_i)\right) \leq \left(1 - \frac{3}{4}\delta\right)^s.$$

Taking the union bound over $w \in W$, we find that

$$\Pr\left(W \not\subseteq \bigcup_{i=1}^{s} N(\hat{u}_i)\right) \leq |W| \left(1 - \frac{3}{4}\delta\right)^s \leq \frac{L}{\delta}\left(1 - \frac{3}{4}\delta\right)^s \leq \eta.$$

□

4.2 Mean-list-decoding and principles for general codes

The results of this section apply to all codes, not just homomorphism codes.

4.2.1 Mean-list-decoding, definitions and results for general codes

Let $\mathscr{F} = \{f_i : i \in I\}$ be a family of received words $f_i \in \Sigma^\Omega$. By the **size** $|\mathscr{F}|$ of $\mathscr{F}$ we mean the size $|I|$ of the index set I. The **average distance** $\mathrm{dist}(w, \mathscr{F})$ of a word $w \in \Sigma^\Omega$ to $\mathscr{F}$ is the average distance of w to elements of $\mathscr{F}$, given by $\mathrm{dist}(w, \mathscr{F}) = \mathbb{E}_{i \in I}[\mathrm{dist}(w, f_i)]$. Similarly, the **average agreement** $\mathrm{agr}(w, \mathscr{F})$ of a word w with $\mathscr{F}$ is the average agreement of w with elements of $\mathscr{F}$, given by $\mathrm{agr}(w, \mathscr{F}) = \mathbb{E}_{i \in I}[\mathrm{agr}(w, f_i)] = 1 - \mathrm{dist}(w, \mathscr{F})$. (The expectation $\mathbb{E}$ is taken with respect to the uniform distribution over I.)

Definition 4.2.1 (Mean-lists)**.** We define the **mean-list** $\mathcal{L}$ as the set of codewords that have agreement at least λ with the received words $\mathscr{F}$, i.e.,

$$\mathcal{L} = \mathcal{L}(\mathcal{C}, \mathscr{F}, \lambda) := \{w \in \mathcal{C} : \mathrm{dist}(w, \mathscr{F}) \leq \lambda\}. \tag{4.1}$$

We will write $\mathfrak{m}\ell(\mathcal{C}, \lambda) := \max_{\mathscr{F}} \mathcal{L}(\mathcal{C}, \mathscr{F}, \lambda)$ for the maximum mean-list size for a given agreement λ. See Definition 4.2.5.

As we shall see, the mean-list-decoding concept helps expand the scope of our results, without making them more difficult to prove. We adapt the terminology of Section 3.1 to the context of mean-list-decoding.

Combinatorial. We wish to bound mean-list size by $|\mathcal{L}(\mathcal{C}, \mathscr{F}, \rho)| \leq c'(\varepsilon)$. We say that $\mathcal{C}$ is a **CombEconM** ("combinatorially economically mean-list-decodable") code if $c'(\varepsilon) = \mathrm{poly}(1/\varepsilon)$.

Algorithmic. We say that $\mathcal{C}$ is an **AlgEconM** ("algorithmically economically mean-list-decodable") code if it satisfies the definition of AlgEcon codes, with the following modifications.

The received word f is replaced by a family $\mathscr{F}$ of received words and the list $\mathcal{L}$ becomes $\mathcal{L} = \mathcal{L}(\mathcal{C}, \mathscr{F}, \rho)$. Oracle access to $\mathscr{F}$ means that, given $i \in I$ and $\omega \in \Omega$, the oracle returns

$f_i(\omega)$. Condition (ii) is replaced by the following.

(ii-M) $\mathrm{poly}(\log|\Omega|, \log|\Sigma|, \log|\mathscr{F}|, 1/\varepsilon)$ amount of work.

Note that the number of queries to the family $\mathscr{F}$ remains $\mathrm{poly}(\log|\Omega|, 1/\varepsilon)$.

Certificate. We say that $\mathcal{C}$ is a **CertEconM** ("certificate economically mean-list-decodable") code if it satisfies the definition of CertEcon codes, with the same modifications as AlgEconM.

The next result shows that economical mean-list-decoding is no harder than economical list-decoding.

Theorem 4.2.2 (Mean-list-decoding, main)**.** *For a class $\mathscr{C}$ of codes, we have the following.*

(i) $\mathscr{C}$ is CombEconM if and only if it is CombEcon.

(ii) $\mathscr{C}$ is AlgEconM if and only if it is AlgEcon.

(iii) $\mathscr{C}$ is CertEconM if and only if it is CertEcon.

This is proved in the next section as Corollary 4.2.11 and Theorem 4.2.13.

Remark 4.2.3 (Significance of mean-list-decoding)**.** Dinur et al. show the CombEcon and AlgEcon list-decodability of {abelian→abelian} homomorphism codes [DGKS08]. We shall see that Theorem 4.2.2 quickly leads to the conclusion of CombEcon list-decodability of {arbitrary→abelian} homomorphism codes. The same inference can be made about AlgEcon list-decodability, assuming natural conditions about representation of the domain group. This is the subject of Section 4.3.2.

Strong mean-list-decoding

We say that $\mathcal{C}$ is a **strong AlgEconM** code if it satisfies the definition of AlgEconM,

except with (ii-M) replaced by (ii'-M) below. Similarly, we say that $\mathcal{C}$ is a **strong CertEconM** code if it satisfies the definition of CertEconM, except with (ii-M) replaced by (ii'-M) below.

(ii'-M) poly$(\log|\Omega|, 1/\varepsilon)$ amount of work in the unit cost model for Σ and unit sampling cost model for $\mathscr{F}$.

In the **unit sampling cost model** for $\mathscr{F} = \{f_i : i \in I\}$, we charge unit cost for naming any $i \in I$ and for generating a uniform random $i \in I$.

Remark 4.2.4. Theorem 4.2.2 still holds if all mentions of economical list-decoding are replaced by strong economical list-decoding.

4.2.2 List size versus mean-list size

The goal of this section is to prove Theorem 4.2.2, using the Bipartite Covering Lemma (Lemma 4.1.1).

Lemma 4.2.8, shows that mean-lists are contained in a small number of random lists, with a slight degradation of the parameters. That mean-list size is bounded by list-size is shown by item (i) of Lemma 4.2.8). It follows immediately that the concepts of CombEconM and CombEcon are equivalent (Corollary 4.2.11). Lemma 4.2.8 item (ii) shows the equivalence of AlgEconM with AlgEcon and CertEconM with CertEcon, completing the proof of Theorem 4.2.2.

Further consequences of Lemma 4.2.8 will follow in Section 4.3.2, leading to the generic constraint relaxation principle in the domain.

Recall that $\ell(\mathcal{C}, \lambda)$ denotes the maximum list size for $\mathcal{C}$ with agreement λ. Now we define the analogous quantities for mean-lists. We denote by $\mathcal{C}$ a code, r and s natural numbers, and $\lambda, \delta > 0$.

Definition 4.2.5 (Mean-list-size)**.** The **maximum r-mean-list size** for $\mathcal{C}$ with agreement λ, denoted $\mathfrak{m}_r\ell(\mathcal{C},\lambda)$, is the maximum size of the mean-lists $\mathcal{L}(\mathcal{C},\mathscr{F},\lambda)$ over all families $\mathscr{F}$ of r received words, i.e.,

$$\mathfrak{m}_r\ell(\mathcal{C},\lambda) = \max\{|\mathcal{L}(\mathcal{C},\mathscr{F},\lambda)| : |\mathscr{F}| = r\}.$$

The **maximum mean-list size** for $\mathcal{C}$ with agreement λ is the maximum over the r-mean-list sizes for $\mathcal{C}$ with agreement λ, i.e.,

$$\mathfrak{m}\ell(\mathcal{C},\lambda) = \max_r \mathfrak{m}_r\ell(\mathcal{C},\lambda).$$

Note that $\mathfrak{m}_1\ell(\mathcal{C},\lambda) = \ell(\mathcal{C},\lambda)$.

From the definitions it follows that $\mathrm{aHom}(G,H)$ is CombEconM if and only if

$$\mathfrak{m}\ell(\mathrm{aHom}(G,H), \Lambda_{G,H}+\varepsilon) = \mathrm{poly}(1/\varepsilon)\,.$$

Notation 4.2.6. For a word w, we denote by $w * r = (\overbrace{w \dots w}^{r})$ the word found by concatenating r copies of w. For a set $\mathcal{S}$ of words, we write $\mathcal{S} * r := \{w * r : w \in \mathcal{S}\}$.

Remark 4.2.7 (Mean-list-decoding versus repeated codes)**.** Let $\mathscr{F} = \{f_i : i \in [r]\}$ be a family of r received words. Notice that $\mathcal{L}(\mathcal{C} * r, (f_1,\dots,f_r), \lambda)$ is the r-fold repetition of $\mathcal{L}(\mathcal{C},\mathscr{F},\lambda)$, i.e.,

$$\mathcal{L}(\mathcal{C},\mathscr{F},\lambda) * r = \mathcal{L}(\mathcal{C} * r, (f_1,\dots,f_r), \lambda).$$

It follows that $\mathfrak{m}_r\ell(\mathcal{C},\lambda) = \ell(\mathcal{C} * r, \lambda)$. In this way, mean-list-decoding can be viewed as list-decoding repeated codes.

Next we state the central result of this section, that every mean-list is covered by a small number of lists.

Lemma 4.2.8 (Concentration of mean-lists)**.** *Let $\mathcal{C}$ be a code and $\lambda, \delta > 0$. Let $\mathscr{F} = \{f_i : i \in I\}$ be a family of received words. Let $\mathcal{L} = \mathcal{L}(\mathcal{C}, \mathscr{F}, \lambda + \delta)$. We conclude the following.*

(i) $|\mathcal{L}| \leq \ell(\mathcal{C}, \lambda)/\delta$.

(ii) Set $s = \left\lceil \frac{4}{3\delta}(\ln \ell(\mathcal{C}, \lambda) + \ln(1/\eta\delta) \right\rceil$. *Choose a sequence* $(j_1, \ldots, j_s) \in I^s$ *uniformly at random. For each* i $(1 \leq i \leq s)$ *independently apply the list-decoder to the received word* f_{j_i} *with agreement threshold* λ. *Let* $\mathcal{L}_i$ *denote the output list. Then, with probability* $\geq 1 - \eta$, *we have* $\mathcal{L} \subseteq \bigcup_{i \in S} \mathcal{L}_i$.

Not only does this give combinatorial bounds for mean-lists in terms of lists, it will be used to provide a (certificate-)mean-list-decoder from a (certificate-)list-decoder. Below we will let $\mathcal{L}_i$ be the output list on input function f_i, and the success probabilities of $\mathcal{L}_i$ are set up exactly to account for the 1/4 failure probability.

The proof will follow from the Bipartite Covering Lemma (Lemma 4.1.1) together with the following observation.

Lemma 4.2.9 (Markov degredation)**.** *Fix a codeword φ. Let $\mathscr{F} = \{f_i : i \in I\}$ be a family of received words in the codespace. Assume $\mathbb{E}_i(\mathrm{agr}(\varphi, f_i)) \geq \lambda + \delta$. Then $\Pr_i(\mathrm{agr}(\varphi, f_i) > \lambda) > \delta$.*

Proof. Let $x_i = \mathrm{dist}(\varphi, f_i) = 1 - \mathrm{agr}(\varphi, f_i)$. Then $\mathbb{E}_i(x_i) \leq 1 - \lambda - \delta$. Therefore, by Markov's inequality, $\Pr(\mathrm{agr}(\varphi, f_i) \leq \lambda) = \Pr(\mathrm{dist}(\varphi, f_i) \geq 1 - \lambda) \leq \frac{1-\lambda-\delta}{1-\lambda} = 1 - \frac{\delta}{1-\lambda} < 1 - \delta$. □

Proof of Lemma 4.2.8. We apply Lemma 4.1.1 to the bipartite graph $X = (V, W; E)$, where the edge set E consists of the pairs $(i, \varphi) \in I \times \mathcal{L}$ satisfying $\mathrm{agr}(f_i, \varphi) > \lambda$. Then, $\deg(f) \leq \ell(\mathcal{C}, \lambda)$ by definition of max list size ℓ and $\deg(\varphi) \geq \delta|\mathscr{F}|$ by Lemma 4.2.9.

The decoder succeeds with probability $\geq 3/4$, so $\mathcal{L}_i \supseteq \mathcal{L}(\mathcal{C}, f_i, \lambda)$ happens with probability $\geq 3/4$ independently over $i = 1, \ldots, s$. The lemma follows from Lemma 4.1.1. □

Corollary 4.2.10. *For $\mathcal{C}$ a code, r a natural number, and $\lambda, \delta > 0$, we have*

$$\mathfrak{m}\ell(\mathcal{C}, \lambda + \delta) \leq \frac{1}{\delta}\mathfrak{m}_r\ell(\mathcal{C}, \lambda). \tag{4.2}$$

Proof. Let s be a natural number. By definition of $\mathfrak{m}\ell$, it suffices to show that $\mathfrak{m}_s\ell(\mathcal{C}, \lambda+\delta) \leq \frac{1}{\delta}\mathfrak{m}_r\ell(\mathcal{C}, \lambda)$.

But, we find that $\mathfrak{m}_s\ell(\mathcal{C}, \lambda + \delta) \leq \mathfrak{m}_{sr}\ell(\mathcal{C}, \lambda + \delta)$ by [GS14, Lemma 3.3] (though their lemma is stated in terms of repeated codes). By Lemma 4.2.8, we find that $\mathfrak{m}_{sr}\ell(\mathcal{C}, \lambda+\epsilon) \leq \frac{1}{\delta}\mathfrak{m}_r\ell(\mathcal{C}, \lambda)$. □

The following is now immediate.

Corollary 4.2.11. *For $\mathcal{C}$ a code and $\varepsilon > 0$, we have*

$$\mathfrak{m}\ell(\mathcal{C}, 1 - \mathit{mindist} + \varepsilon) \leq \frac{2}{\varepsilon}\ell(\mathcal{C}, 1 - \mathit{mindist} + \varepsilon/2).$$

In other words, if a class of codes is CombEcon with degree c, then it is CombEcon with degree $c + 1$.

Next, we derive the algorithmic versions of this result. We shall make the following assumption on access to our family $\mathscr{F} = \{f_i : i \in I\}$ of received words.

Access 4.2.12. An oracle provides uniform random elements of the index set I of $\mathscr{F}$.

Theorem 4.2.13. *Under Access 4.2.12, if a class $\mathscr{C}$ of codes is AlgEcon then it is AlgEconM. Under the same assumptions, if $\mathscr{C}$ is CertEcon then it is CertEconM.*

Remark 4.2.14. The bounds on cost in the result above deteriorate as follows.

- A $\frac{2}{\varepsilon}$ multiplicative factor in list size.
- An $O(\frac{1}{\varepsilon}\ln(1/\varepsilon))$ multiplicative factor in queries to the received word f.

- An $O(\frac{1}{\varepsilon}\ln(1/\varepsilon))$ multiplicative factor in amount of work.

We show that, to (certificate-)mean-list-decode a family $\mathscr{F}$ of functions, (certificate-)list-decoding a small random subset of the functions in $\mathscr{F}$ suffices. Lemma 4.1.1 already contains the machinery to guarantee the necessary probability of success.

Proof. We first prove the claim for AlgEcon. Let DECODE be a list-decoder for the class $\mathscr{C}$ satisfying AlgEcon assumptions. We denote by $\textsc{Decode}(\mathcal{C}, f, 1-\mathsf{mindist}+\varepsilon)$ the output of DECODE on the input f and $\varepsilon > 0$, where f is a received word in the code space of a code $\mathcal{C} \in \mathscr{C}$.

We describe here a mean-list-decoder that satisfies AlgEconM. It takes as input $\mathscr{F}$ and $\varepsilon > 0$, where $\mathscr{F}$ is the family of received words in the code space of the code $\mathcal{C} \in \mathscr{C}$.

Denote by I the index set of $\mathscr{F}$. Via the provided oracle, generate a subset $S \subset I$ by picking s elements of I independently and uniformly. The value of s will be determined later. Return the list given by

$$\bigcup_{i \in S} \textsc{Decode}(\mathcal{C}, f_i, 1-\mathsf{mindist}+\epsilon/2). \tag{4.3}$$

We show that this is a list-decoder satisfying the conditions of AlgEconM through direct application of Lemma 4.2.8.

The output $\textsc{Decode}(\mathcal{C}, f, 1-\mathsf{mindist}+\varepsilon/2)$ contains $\mathcal{L}(\mathcal{C}, f, 1-\mathsf{mindist}+\varepsilon/2)$ with probability 3/4 by the definition of list-decoder. If s is set as in Lemma 4.2.8 (ii) with $\eta = 1/4$ and $\delta = \varepsilon/2$, we find that the the desired mean-list is returned with probability at least 3/4.

Notice that the list-decoder DECODE is called $s = \left\lceil \frac{8}{3\varepsilon}\left(\ln(\ell(\mathcal{C}, \lambda) + \ln(8/\varepsilon)\right)\right\rceil$ times as a subroutine, where $\lambda = 1-\mathsf{mindist}+\varepsilon/2$. Very little processing is done outside of these calls. Moreover, since $\mathcal{C}$ is AlgEcon, it is CombEcon, so $\ell(\mathcal{C}, 1-\mathsf{mindist}+\varepsilon/2) = \text{poly}(\varepsilon/2)$ and DECODE is called $O(\frac{1}{\varepsilon}\ln(1/\varepsilon))$ times.

The proof for CertEcon is similar, found mainly by replacing the occurrences of "list-decoder" with "certificate-list-decoder." The mean-certificate-list-decoder returns the union of s output lists by DECODE, which would denote the assumed certificate-list-decoder. □

The same conclusions follow for the strong versions of these concepts.

Remark 4.2.15. Knowledge of mindist is not needed in the conversion from CertEcon to CertEconM. Even in the AlgEcon case, mindist is only needed if required by the list-decoder DECODE. The crucial knowledge for this conversion is ε, so that the deterioration factor (denoted δ above) can be controlled. This deterioration factor is set to $\delta = \varepsilon/2$ in our proofs.

4.3 Applications of Bipartite Covering Lemma and mean-list-decoding in homomorphism codes

4.3.1 Extending the domain: the irrelevant kernel and consequences

We define the irrelevant kernel, state relevant results, then illustrate how it allows a relaxation on the class of groups allowed in the domain.

Definition 4.3.1 (Irrelevant kernel)**.** Let G and H be groups. The (G, H)**-irrelevant kernel** (or "irrelevant kernel" if G and H are clear) is the intersection of the kernels of all $G \to H$ homomorphisms, i.e.,

$$\bigcap_{\varphi \in \mathrm{Hom}(G,H)} \ker(\varphi). \tag{4.4}$$

We call elements and subgroups of the irrelevant kernel **irrelevant**.

For instance, if H is abelian, then the commutator subgroup G' is irrelevant.

We shall find that extending G/N to G retains economical list-decodability, as long as N is (G, H)-irrelevant.

Theorem 4.3.2. *Let N be an irrelevant normal subgroup of G. Then, $\Lambda_{G/N,H} = \Lambda_{G,H}$. Moreover,*

(i) if $\mathrm{aHom}(G/N, H)$ *is CombEcon then* $\mathrm{aHom}(G, H)$ *is CombEcon;*

(ii) if $\mathrm{aHom}(G/N, H)$ *is CertEcon then* $\mathrm{aHom}(G, H)$ *is CertEcon;*

(iii) if $\mathrm{aHom}(G/N, H)$ *is AlgEcon then* $\mathrm{aHom}(G, H)$ *is AlgEcon.*

For items (ii) and (iii) we need to make suitable assumptions on access to the domain.

This is proved in the next section, relying on results from mean-list-decoding (Theorem 4.2.2).

Corollary 4.3.3. *The code* $\mathrm{aHom}(G, H)$ *is AlgEcon for any finite group G and any finite nilpotent group H.*

Proof. Combine Theorem 4.3.2 and the main result of [GS14], i.e., {nilpotent→nilpotent} homomorphism codes are AlgEcon. For abelian H, use instead the main result of [DGKS08], i.e., {abelian→abelian} homomorphism codes are AlgEcon. □

4.3.2 Repeated codes: irrelevant normal subgroups and mean-lists

We state formally state and prove our claims regarding relaxing the domain. Its implications were discussed in Section 4.3.1.

We first identify the code $\mathrm{aHom}(G, H)$ with a repeated code found from $\mathrm{aHom}(G/N, H)$. This hinges on N being an irrelevant normal subgroup. Recall that N is (G, H)-irrelevant if N is contained in the kernel of every $G \to H$ homomorphism (see Definition 4.3.1).

For groups K and H, an enumeration $K = \{k_1, \ldots, k_{|K|}\}$ induces a bijection between the set of functions H^K and the set of words $H^{|K|}$ by $f \mapsto (f(k_1), \ldots, f(k_{|K|}))$.

Observation 4.3.4 (Identification of $\mathrm{aHom}(G, H)$ lists with $\mathrm{aHom}(G/N, H)$ mean-lists)**.** *Let G, H and N be groups such that N is a (G, H)-irrelevant normal subgroup. Let $f : G \to$*

H. There are enumerations of G and G/N, and a family $\mathscr{F}$ of functions $G/N \to H$ such that

$$\mathcal{L}(\mathrm{aHom}(G,H), f, \lambda) = \mathcal{L}(\mathrm{aHom}(G/N,H), \mathscr{F}, \lambda) * |N|.$$

Proof. The enumeration can be found by filling elements of G in a matrix where each row corresponds to a coset of N in G. Elements of G are then ordered by column. We make this more precise.

We identify $\mathrm{aHom}(G,H)$ with $\mathrm{aHom}(G/N,H) * |N|$, then represent $f : G \to H$ as a set $\mathscr{F}$ of $|N|$ functions that map $G/N \to H$.

Let $\tau : G/N \to G$ be a transversal, i.e., an injection mapping each coset to one coset representative. Denote by $S = \tau(G/N)$ the image of τ, which forms a set of of left coset representatives for G/N. Notice that $G = SN$. Let $\pi : G \to G/N$ be the projection onto cosets. (We have that $\pi \circ \tau : G/N \to G/N$ is the identity.) Enumerate the elements of N as $g_1, \ldots, g_{|N|}$.

First, note $\mathrm{aHom}(G,H) = \{\varphi \circ \pi : \varphi \in \mathrm{aHom}(G/N,H)\}$, since N is a (G,H)-irrelevant subgroup. So, $\mathrm{aHom}(G,H)$ can be viewed as $\mathrm{aHom}(G/N,H) * |N|$. In other words, the codeword $\varphi \circ \pi$ is a concatenation $(\varphi, \ldots, \varphi)$ of $|N|$ copies of the word φ (under the correct ordering of G).

We will make this precise for a function in general. Consider a function $f : G \to H$. For $i = 1, \ldots, |N|$, define the function $f_i : G/N \to H$ by $f_i(sN)) = f(\tau(sN)g_i)$ for every coset sN.

Define $\mathscr{F} = \{f_1, \ldots, f_{|N|}\}$. If we view $f : G \to H$ as a word of length $|G|$, it is the concatenation $f = (f_1, \ldots, f_{|N|})$ of the functions in $\mathscr{F}$ (under the correct ordering of G). Notice that $f_i(sN) = \varphi(sN)$ exactly if $f(sg_i) = \varphi(sN) = (\varphi \circ \pi)(sg_i)$. So, for any

$\varphi \in \text{aHom}(G/N, H)$, we have

$$\begin{aligned}
\text{agr}(f, \varphi \circ \pi) &= \frac{1}{|G|}|\{g : f(g) = (\varphi \circ \pi)(g)\}| \\
&= \frac{1}{|N|}\frac{1}{|G/N|}\sum_{i=1}^{|N|}|\{s \in G/N : f(sg_i) = (\varphi \circ \pi)(sg_i)\}| \\
&= \frac{1}{|N|}\sum_{i=1}^{|N|}\text{agr}(f_i, \varphi) \\
&= \mathop{\mathbb{E}}_{i}[\text{agr}(f_i, \varphi)].
\end{aligned}$$

This shows that $\varphi \in \mathcal{L}(\text{aHom}(G/N, H), \mathscr{F}, \lambda)$ exactly if $\varphi \circ \pi \in \mathcal{L}(\text{aHom}(G, H), f, \lambda)$. We have illustrated the claim that

$$\begin{aligned}
\mathcal{L}(\text{aHom}(G, H), f, \lambda) &= \{\varphi \circ \pi : \varphi \in \mathcal{L}(\text{aHom}(G/N, H), \mathscr{F}, \lambda)\} \\
&= \mathcal{L}(\text{aHom}(G/N, H), \mathscr{F}, \lambda) * |N|.
\end{aligned}$$

□

Remark 4.3.5. The proof of Observation 4.3.4 shows more: There is a bijection between functions $f \in H^G$ and families $\mathscr{F}$ of $|N|$ functions so that the equations holds.

Corollary 4.3.6. *If G, H and N are groups such that N is a (G, H)-irrelevant normal subgroup of G, then*

$$\ell(\text{aHom}(G, H), \lambda) = \mathfrak{m}_{|N|}\ell(\text{aHom}(G/N, H), \lambda).$$

We first illustrate the relaxation principle combinatorially, through bounds on list-size. The goal would be to conclude that, if $\mathfrak{G} \times \mathfrak{G}$ is CombEconM for a class $\mathfrak{G}$ of groups, then $\mathfrak{Groups} \times \mathfrak{G}$ is CombEcon. This principle will generalize to CertEcon and AlgEcon as well.

Remark 4.3.7. If N is a (G, H)-irrelevant normal subgroup, then $\Lambda_{G/N,H} = \Lambda_{G,H}$.

Lemma 4.3.8 (Irrelevant normal subgroup lemma)**.** *Let G, H and N be groups such that N is a (G, H)-irrelevant normal subgroup. Then,*

$$\ell(\mathrm{aHom}(G, H), \Lambda_{G,H} + \varepsilon) \leq \frac{2}{\varepsilon} \cdot \ell(\mathrm{aHom}(G/N, H), \Lambda_{G,H} + \varepsilon/2).$$

Proof. Calculate

$$\begin{aligned}
\ell(\mathrm{aHom}(G, H), \Lambda + \varepsilon) &= \mathfrak{m}_{|N|}\ell(\mathrm{aHom}(G/N, H), \Lambda + \varepsilon) && \text{Corollary 4.3.6} \\
&\leq \mathfrak{m}\ell(\mathrm{aHom}(G/N, H), \lambda) && \text{definition of } \mathfrak{m}\ell \\
&\leq \frac{2}{\varepsilon} \cdot \mathfrak{m}_1\ell(\mathrm{aHom}(G/N, H) && \text{Corollary 4.2.10 with } r = 1 \\
&= \frac{2}{\varepsilon} \cdot \ell(\mathrm{aHom}(G/N, H), \Lambda + \varepsilon/2).
\end{aligned}$$

□

This implies that, if $\mathrm{aHom}(G/N, H)$ is CombEconM, then $\mathrm{aHom}(G, H)$ is CombEcon. This principle holds for CertEcon and AlgEcon as well.

Theorem 4.3.9. *Let G, H and N be groups such that N is a (G, H)-irrelevant normal subgroup of G.*

(i) If $\mathrm{aHom}(G/N, H)$ *is CombEcon, then* $\mathrm{aHom}(G, H)$ *is CombEcon.*

(ii) Under suitable access assumptions (Access 4.3.10 (ii)), if $\mathrm{aHom}(G/N, H)$ *is CertEcon, then* $\mathrm{aHom}(G, H)$ *is CertEcon.*

(iii) Under suitable access assumptions (Access 4.3.10 (iii)), if $\mathrm{aHom}(G/N, H)$ *is AlgEcon, then* $\mathrm{aHom}(G, H)$ *is AlgEcon.*

The deterioration in cost is as described in Remark 4.2.14.

Access 4.3.10. (ii) (a) Elements of N can be generated uniformly. (b) A transversal, i.e., an injection $G/N \to G$ that assigns a representative element to each coset, is

given. (c) G/N is known well enough to satisfy the CertEcon access assumptions on $\mathrm{aHom}(G/N, H)$.

(iii) (a') Elements of N can be generated uniformly and generators for N are given. (b') Same as (b). (c') Same as (c), for AlgEcon.

Remark 4.3.11. If the access assumptions on N are at least as strong as having N as a black-box group, then generating (nearly) uniform elements and being given a set of generators are equivalent. If N is a black-box group, generators are given by definition. Nearly uniform random elements in black-box groups can be generated in polynomial time (polynomial in the encoding length of groups elements).

Remark 4.3.12. For the proof of this theorem, we will actually need the –EconM versions of the assumptions, which we may assume as a consequence of Theorem 4.2.2.

Proof of Theorem 4.3.9. Set $\Lambda = \Lambda_{G,H} = \Lambda_{G/N,H}$. Let $\pi : G \to G/N$ be the projection onto cosets.

(i) $\mathrm{aHom}(G/N, H)$ is CombEconM, so $\mathrm{aHom}(G, H)$ is CombEcon by Corollary 4.3.6.

(ii) By assumption, a certificate-list-decoder satisfying the conditions of CertEconM exists for $\mathrm{aHom}(G/N, H)$. Its output list Γ is a certificate list for $\mathcal{L}(\mathrm{aHom}(G/N, H), \mathscr{F}, \Lambda + \varepsilon)$, where $\varepsilon > 0$ and $\mathscr{F} = \{f_1, \ldots, f_{|N|}\}$ is constructed from f as in Observation 4.3.4. We construct a certificate list $\widetilde{\Gamma}$ for $\mathcal{L}(\mathrm{aHom}(G, H), f, \Lambda+\varepsilon)$, by replacing each $G/N \rightharpoonup H$ partial map $\gamma \in \Gamma$ with a $G \rightharpoonup H$ partial map $\widetilde{\gamma}$ defined as follows.

Denote by $\tau : G/N \to G$ the injection guaranteed by the assumption. Let $\widetilde{\gamma}$ have domain $\mathrm{dom}(\widetilde{\gamma}) = \tau(\mathrm{dom}(\gamma))$, and define $\widetilde{\gamma}(g) = \gamma(\pi(g))$ for each $g \in \mathrm{dom}(\widetilde{\gamma})$. If γ is a certificate for $\varphi \in \mathrm{aHom}(G/N, H)$, then $\widetilde{\gamma}$ is a certificate for $\varphi \circ \pi \in \mathrm{aHom}(G, H)$.

(iii) A list-decoder satisfying the conditions of AlgEconM exists for $\mathrm{aHom}(G/N, H)$. By Observation 4.3.4, it suffices to, given the list $\mathcal{L} = \mathcal{L}(\mathrm{aHom}(G/N, H), \mathscr{F}, \Lambda + \varepsilon)$, return the list $\widetilde{\mathcal{L}} = \{\varphi \circ \pi : \varphi \in \mathcal{L}\}$. We address algorithmic issues of defining $\varphi \circ \pi$ from φ.

Denote by X the given set of generators of N and by $\tau : G/N \to G$ the given injective map. Each homomorphism φ is represented by its values on a set Y of generators of G/N. The set $X \cup \tau(Y)$ is a set of generators for G. Define $\widetilde{\varphi}$ on this set by the following.

$$\widetilde{\varphi}(g) = \begin{cases} 1 & g \in X \\ (\varphi \circ \pi)(g) & g \in \tau(Y) \end{cases}.$$

□

A class $\mathfrak{G}$ of finite groups is a **quasivariety** if it is closed under subgroups and direct products. The classes of abelian, nilpotent, and solvable groups are examples of quasivarieties.

Theorem 4.3.13. *Let $\mathfrak{G}$ be a quasivariety of finite groups.*

(i) Suppose $\mathrm{aHom}(G, H)$ *is CombEcon for every $G, H \in \mathfrak{G}$. Then,* $\mathrm{aHom}(G, H)$ *is CombEcon for $H \in \mathfrak{G}$ and arbitrary G.*

(ii) Under suitable access assumptions (Access 4.3.15 (ii)), if $\mathrm{aHom}(G, H)$ *is CertEcon for every $G, H \in \mathfrak{G}$, then* $\mathrm{aHom}(G, H)$ *is CertEcon for $H \in \mathfrak{G}$ and arbitrary G.*

(iii) Under suitable access assumptions (Access 4.3.15 (ii)), if $\mathrm{aHom}(G, H)$ *is AlgEcon for every $G, H \in \mathfrak{G}$, then* $\mathrm{aHom}(G, H)$ *is AlgEcon for $H \in \mathfrak{G}$ and arbitrary G.*

The deterioration in cost is as described in Remark 4.2.14.

Remark 4.3.14. In fact, the class $\mathfrak{G}$ need only be closed under subdirect products.

The access assumptions mirror those of Access 4.3.10 of Theorem 4.3.9.

Access 4.3.15. For every $G \in \mathfrak{Groups}$ and $H \in \mathfrak{G}$, denote by N the (G, H)-irrelevant kernel and assume we have access to N and G/N as follows.

(ii) (a) Random elements of N can be generated uniformly. (b) A transversal of G/N in G can be found. (c) G/N can be found well enough to satisfy the access model of the assumed CertEcon list-decodability of pairs in $\mathfrak{G} \times \mathfrak{G}$.

(iii) (a') Random elements of N can be generated uniformly and a set of generators for N can be found. (b') Same as (b). (c') Same as (c) but for AlgEcon list-decodability.

Proof. Fix $H \in \mathfrak{G}$ and $G \in \mathfrak{Groups}$. Let N be the (G, H)-irrelevant kernel. By Theorem 4.3.9, all desired conclusions will follow if we show that $G/N \in \mathfrak{G}$.

Let

$$\widetilde{H} = \prod_{\varphi \in \mathrm{aHom}(G,H)} H.$$

Define the map $\tau : G \to \widetilde{H}$ given by $\tau(g) = (\varphi(g))_\varphi$. Notice that $\tau(G)$ is subgroup of $\widetilde{H}$ and is thus a subdirect product of copies of the group $H \in \mathfrak{G}$. Since $\mathfrak{G}$ is closed under subdirect products, it follows that $\tau(G) \in \mathfrak{G}$.

Since $\ker(\tau) = N$, we have $\tau(G) \cong G/N$, so $G/N \in \mathfrak{G}$.

□

The AlgEcon list-decodability of {abelian→abelian} and {nilpotent→nilpotent} homomorphism codes is shown in [DGKS08] and [GS14], respectively. As the class of abelian groups and the class of nilpotent groups both form quasivarieties, we conclude the following using the mentioned results and Theorem 4.3.13.

Corollary 4.3.16. *If G is a group and H an abelian group (or, more generally, nilpotent), then we have AlgEcon (and therefore CombEcon) list-decoding of* $\mathrm{aHom}(G, H)$.

4.3.3 Translated codes: Hom *versus* aHom

We show that the code $\mathrm{Hom}(G, H)$ is CombEcon if and only if $\mathrm{aHom}(G, H)$ is CombEcon, and similarly for CertEcon. This is true also for AlgEcon under modest assumptions of the

representations of the group. This reflects a similar phenomenon to our results on mean-list-decoding.

From here on, we make no distinction between proving results for the two classes of codes.

We fix terminology for this section. For an affine homomorphism $\varphi \in \mathrm{Hom}(G, H)$, we denote its **base homomorphism** by $\varphi^0 \in \mathrm{Hom}(G, H)$ (the unique homomorphism satisfying $\varphi = h\varphi^0$ for $h \in H$). For an element $a \in G$ and function $f : G \to H$, we denote by $f^a : G \to H$ the function $f^a(g) = f(a)^{-1} f(ag)$.

We state the central result of this section, that every aHom list is contained within a small number of translated Hom lists. It is similar in spirit to Lemma 4.2.8, and it is similarly proved by the Bipartite Covering Lemma (Lemma 4.1.1). The deterioration in list size and cost will be addressed in Remark 4.3.20 below.

Lemma 4.3.17 (Concentration of aHom lists)**.** *Let G and H be groups, $f : G \to H$ a received word, and $0 < \lambda \leq 1$. Let $\mathcal{L} = \mathcal{L}(\mathrm{aHom}(G, H), f, \lambda)$. We conclude the following.*

(i) $|\mathcal{L}| \leq \frac{1}{\lambda}\ell(\mathrm{Hom}(G, H), \lambda)$.

(ii) Let S be a subset of G formed by choosing $\left\lceil \frac{4}{3\lambda}(\ln|\mathcal{L}| + \ln(1/\eta\lambda)) \right\rceil$ elements in G independently and uniformly. Suppose that, independently for each $a \in G$, the subset D_a of $\mathcal{L}$ contains $\mathcal{L}(\mathrm{Hom}(G, H), f^a, \lambda)$ with probability $\geq 3/4$. We denote $\widetilde{D_a} = \{f(a)\psi(a^{-1})\psi \in \mathrm{aHom}(G, H) : \psi \in D_a\}$. Then, with probability $\geq 1 - \eta$, we have

$$\mathcal{L} \subseteq \bigcup_{a \in S} \widetilde{D_a}.$$

We remark that $\widetilde{D_a} \subset \mathrm{aHom}(G, H)$ is found by translating elements of $D_a \subset \mathrm{Hom}(G, H)$, but not all by the same element.

We defer the proof to state the main result of this section, which is an immediate consequence.

Access 4.3.18. An oracle provides uniform random elements of G.

Corollary 4.3.19 (Hom versus aHom)**.** *Let G and H be groups. If* $\mathrm{Hom}(G, H)$ *is CombEcon, then* $\mathrm{aHom}(G, H)$ *is CombEcon. Under Access 4.3.18, if* $\mathrm{Hom}(G, H)$ *is CertEcon, then* $\mathrm{aHom}(G, H)$ *is CertEcon. Under Access 4.3.18, if* $\mathrm{Hom}(G, H)$ *is AlgEcon, then* $\mathrm{aHom}(G, H)$ *is AlgEcon.*

Remark 4.3.20 (Deterioration)**.** The bounds on cost in the result above deteriorate as follows.

- A $\frac{1}{\lambda}$ multiplicative factor in list size.
- An $O(\frac{1}{\lambda}\ln(1/\lambda))$ multiplicative factor in queries to the received word f.
- An $O(\frac{1}{\lambda}\ln(1/\lambda))$ multiplicative factor in amount of work.

Towards proving Lemma 4.3.17, we state a few facts relating affine homomorphisms to their base homomorphisms.

Observation 4.3.21. *Let G and H be groups and $\varphi \in \mathrm{aHom}(G, H)$. Then,*

$$\varphi(a)^{-1}\varphi(ag) = \varphi^0(g) \quad \forall a, g \in G.$$

Corollary 4.3.22. *Let G and H be groups, $f : G \to H$, and $\varphi \in \mathrm{aHom}(G, H)$. If $f(a) = \varphi(a)$, then*

$$f(ag) = \varphi(ag) \iff f(a)^{-1}f(ag) = \varphi^0(g) \iff f^a(g) = \varphi^0(g).$$

It follows that $\mathrm{agr}(f, \varphi) = \mathrm{agr}(f^a, \varphi^0)$.

We can now prove our concentration lemma.

Proof of Lemma 4.3.17. Fix the function $f : G \to H$. Consider the bipartite graph with vertices $V = G$ and $W = \mathcal{L}(\mathrm{aHom}(G, H), f, \lambda)$, where the edge set contains $(a, \varphi) \in G \times \mathrm{aHom}(G, H)$ if $f(a) = \varphi(a)$. We will apply Lemma 4.1.1, so first check the conditions are satisfied.

For every $\varphi \in W$, we have $\mathrm{agr}(\varphi, f) > \lambda$ by definition, so $\deg(\varphi) > \lambda$.

We show that $\deg(a) \leq \ell(\mathrm{Hom}(G, H), \lambda)$, by showing that the map $\varphi \mapsto \varphi^0$ is an injection from $N(a) = \{\varphi \in W : f(a) = \varphi(a)\}$ to $\mathcal{L}(\mathrm{Hom}(G, H), f^a, \lambda)$. This map is well-defined since $f(a) = \varphi(a)$ implies $\mathrm{agr}(f, \varphi) = \mathrm{agr}(f^a, \varphi^0)$, by Corollary 4.3.22. We show this map is injective. If $\varphi_1, \varphi_2 \in \mathrm{aHom}(G, H)$ satisfy $\varphi_1(a) = f(a) = \varphi_2(a)$ and $\varphi_1^0 = \varphi_2^0$, then $\varphi_1(g) = \varphi_1(a)\varphi^0(a^{-1}g) = \varphi_2(a)\varphi^0(a^{-1}g) = \varphi_2(g)$ for all $g \in G$, by Observation 4.3.21.

Apply the two parts of Lemma 4.1.1 to find the following.

(i) We conclude that $|\mathcal{L}(\mathrm{aHom}(G, H), f, \lambda)| \leq \frac{1}{\lambda}\ell(\mathrm{Hom}(G, H), \lambda)$.

(ii) The subset U is the chosen subset S of G. The subset $\hat{U}$ contains the element $a \in U \subseteq G$ if list-decoding for $\mathcal{L}(\mathrm{Hom}(G, H), f^a, \lambda)$ succeeds, which happens with probability $\geq 3/4$ independently over a.

It remains to show that if $D_a = \mathcal{L}(\mathrm{Hom}(G, H), f^a, \lambda)$, then $\widetilde{D_a} \supseteq N(a)$. But, if $\varphi \in N(a)$, we have already established that $\varphi^0 \in \mathcal{L}(\mathrm{Hom}(G, H), f^a, \lambda)$. Moreover, since $f(a) = \varphi(a)$, we have $\varphi(g) = f(a)\varphi^0(a^{-1}g)$ for all $g \in G$, by Observation 4.3.21. So, $\widetilde{D_a} \supseteq N(a)$ by the definition of $\widetilde{D_a}$.

□

CHAPTER 5

COMBINATORIAL LIST-DECODABILITY

5.1 Introduction

This chapter is concerned with the combinatorial economical list-decodability of homomorphism codes with alternating domain. To prove this, we present a set of combinatorial tools that may apply to more general classes of homomorphism codes.

5.1.1 Structure of chapter

In the rest of this introduction section, we contrast our new combinatorial tools with those in the literature.

Section 5.2.4 formally presents our combinatorial strategy. Section 5.2.1 states the useful "strong negative correlation" bound. While it is a consequence of the Johnson bound, we provide a simple proof. Section 5.2.2 shows that we may assume ε is small in terms of Λ, a consequence of strong negative correlation. Sections 5.2.3– 5.2.5 describe our three-step strategy to bound list size: split the list into buckets (spheres centered instead around a homomorphism), split into sub-buckets (the agreement with the homomorphism is localized to a large subgroup), then bound sub-buckets in terms of subgroup depth.

Section 5.3 uses the presented strategy to prove that {alternating→arbitrary} homomorphism codes are CombEcon. Section 5.3.1 presents background results for alternating groups. Section 5.3.2 gives the proof. Section 5.3.3 gives an example of list-size blowup at agreement Λ, providing evidence that $(1 - \Lambda)$ is the list-decoding radius.

5.1.2 Previous strategies

Previous approaches [GKS06, DGKS08, GS14] to proving CombEcon list-decodability relied on generalizing an interpretation of the Johnson bound, which bounds the size of q-ary error

correcting codes in terms of the code length, distance, and q. This bound is shown to hold for agreement sets of homomorphisms in the list, by an inductive argument through a subgroup chain in the codomain. With this approach, structural constraints on the codomain are unavoidable.

A "base case" weighted p-ary Johnson bound is stated below.

Lemma 5.1.1 (p-ary Johnson bound)**.** *Let $f, \varphi_1, \ldots, \varphi_\ell : [n] \to [q]$ be functions satisfying*

(1) $\mathrm{agr}(f, \varphi_i) = 1/q + \alpha_i$ *for* $\alpha_i \geq 0$*, and*

(2) $\mathrm{agr}(\varphi_i, \varphi_j) \leq 1/q$ *for every* $i \neq j$*.*

Then,

$$\sum_{i=1}^{\ell} \alpha_i^2 \leq 1. \tag{5.1}$$

The breakthrough in [DGKS08] showed that *special intersecting families* of sets satisfy the inequality of Equation (5.1), with some constant C instead of 2. This bounds the size of the family. An inductive argument showed that agreement sets between the received word and homomorphisms in the list formed special intersecting families.

Special intersecting families are sets with small intersection, much like Conditions (1) and (2) of Lemma 5.1.1. In addition, they must already satisfy a base inequality such as Equation (5.1) and a "Helly"-like condition. We state the definition and the central lemma of special intersecting families from [DGKS08] below.

Definition 5.1.2 (Special intersecting family)**.** A collection $S_1, \ldots, S_\ell \subseteq X$ of subsets is a (ρ, τ, c)**-special intersecting family** if the following hold:

1. $\mu(S_i) \geq \rho$ for each i;
2. $\mu(S_i \cap S_j) \leq \rho$ whenever $i \neq j$;
3. $\sum_{i=1}^{\ell} (\mu(S_i) - \rho)^c \leq 1$;

4. If $J \subseteq I \subseteq [\ell]$, $|J| \geq 2$, and $\mu(S_I) > \tau$, then $S_I = S_J$, where $S_K = \cap_{i \in K} S_i$ for any $K \subseteq [\ell]$;

Theorem 5.1.3 (Special intersecting theorem)**.** *For every $c < \infty$, there exists $C = C(c) < \infty$, the* ***special intersecting number for*** *c, such that the following holds: if $S_1, \ldots, S_\ell$ form a (ρ, ρ^2, c)-special intersecting family, with $\mu(S_i) = \rho + \alpha_i$ and $\mu(\bigcup_i S_i) = \rho + \alpha$, then*

$$\alpha^C \geq \sum_{i=1}^{\ell} \alpha_i^C. \tag{5.2}$$

In fact, one can take $C = 2c \cdot (c+1)(4 + (c+1)\log_2 3)$.

Theorem 5.1.3 is key to an inductive argument that the inequality of Equation 5.1, with 2 replaced by C, holds for agreement sets.

For the case of abelian (and nilpotent) codomain, the codomain has a normal series with prime cyclic factors. By quotienting out successively smaller subgroups of the codomain, the strategy of [DGKS08] shows that the agreement sets of homomorphisms with the received word form special intersecting families.

5.1.3 Our strategies

Our new tools initialize via a Johnson-like argument on agreement sets, to break the desired list into "buckets." Further bucket splitting and arguments through the subgroup lattice of the *domain* give us our combinatorial bounds.

We remark that these arguments work only in the domain. This allows us to drop *all structural constraints* on the codomain.

We give a brief overview of the proof that alternating groups are universally CombEcon.

First, we introduce a tool called "strong negative correlation," which similarly bounds the number of sets with certain intersection properties. While this tool does follow from

the Johnson bound, a simple proof with a slightly improved constant will be presented in Section 5.2.1.

Strong negative correlation will, similarly to prior approaches, be applied to agreement sets between the received word and homomorphisms in the list. The assumptions are fulfilled by *all* lists up to minimum distance. So, this tool is more widely applicable than special intersecting families but lacks the same inductive power.

Strong negative correlation is used to split the list of homomorphisms into "buckets" (Section 5.2.1), where each bucket consists of a ball around a representative homomorphism (homomorphisms in the have high agreement with a representative homomorphism). The buckets, centered around a homomorphism, are split further into "sub-buckets" (Section 5.2.4), where the agreement with the representative homomorphism is localized to a large subgroup. This splitting is described in Remark 5.2.11.

A "lattice-climbing" argument (Section 5.2.5) bounds the size any set of homomorphisms that agree on a subgroup, by a polynomial with degree the depth of the subgroup. In particular, sub-buckets have polynomially bounded size.

5.2 Tools

5.2.1 Strong negative correlation

We define strongly negatively correlated families of subsets and give a simple proof for a bound on their sizes. We apply this bound via a sphere-packing argument to divide lists into "few" buckets. Another consequence is that we may without loss of generality assume that $\varepsilon < \sqrt{2\Lambda}$.

Definition 5.2.1 (Strong negative correlation)**.** Let $0 < \rho \leq 1$ and $\tau > 0$. Let X be a finite set and let $S_1, \ldots, S_r \subseteq X$. We say that $S_1, \ldots, S_r$ are (ρ, τ)-**strongly negatively correlated** if

(1) $\mu(S_i) \geq \rho$ for all i, and

(2) $\mu(S_i \cap S_j) \leq \rho^2 - \tau$ for all $i \neq j$.

We will apply this with S_i as the agreement set $\mathrm{agr}(\varphi_i, f)$ between a homomorphism φ_i in the list and the received word f. Notice the similarity between the definition of strong negative correlation and the conditions of Lemma 5.1.1 (p-ary Johnson bound) and Definition 5.1.2 (special intersecting families). We give a simple proof of a Johnson-type bound on the number of strongly negatively correlated subsets.

Lemma 5.2.2 (Strong negative correlation bound)**.** *Let $0 < \rho < 1$ and $\tau > 0$. Let X be a finite set and let $S_1, \ldots, S_r \subseteq X$ be (ρ, τ)-strongly negatively correlated subsets. Then, $r \leq \frac{1}{4\tau} + 1$.*

Proof. Choose x uniformly from X. For $1 \leq i \leq r$, let $Z_i(x) = \chi[x \in S_i]$ be the indicator random variable for the event that $x \in S_i$. Notice that $\mathrm{Var}(Z_i) = \mu(S_i)(1 - \mu(S_i)) \leq \frac{1}{4}$.

For $i \neq j$,

$$\mathrm{Cov}(Z_i, Z_j) = \mathbb{E}[Z_i Z_j] - \mathbb{E}[Z_i]\,\mathbb{E}[Z_j] \leq (\rho^2 - \tau) - \rho^2 = -\tau.$$

So,

$$0 \leq \mathrm{Var}\left(\sum_i Z_i\right) = \sum_i \mathrm{Var}(Z_i) + \sum_{i \neq j} \mathrm{Cov}(Z_i, Z_j) \leq \frac{r}{4} + r(r-1)(-\tau).$$

Solving for r gives the bound as claimed. □

5.2.2 Small ε assumption

The first consequence of strong negative correlation (Lemma 5.2.2) is that we may assume ε is "small," i.e., $\varepsilon < \sqrt{2\Lambda}$. So, to show CombEcon it suffices to show a list-size bound of $\mathrm{poly}(1/\varepsilon, 1/\Lambda)$, instead of $\mathrm{poly}(1/\varepsilon)$.

Lemma 5.2.3 (Small ε lemma)**.** *Let G be a finite group and H a group. Suppose that $\Lambda \leq \frac{1}{2}\varepsilon^2$. Then, $\ell(\mathrm{aHom}(G,H), \Lambda+\varepsilon) \leq \frac{1}{2\varepsilon^2}+1$. In particular, $\mathrm{aHom}(G,H)$ is combinatorially $(\Lambda+\varepsilon, \mathrm{poly}(1/\varepsilon))$-list-decodable.*

Proof. Let $\mathcal{L} = \mathcal{L}(\mathrm{aHom}(G,H), \Lambda+\varepsilon)$. We show that the sets $\{\mathrm{Eq}(f,\varphi)\}_{\varphi\in\mathcal{L}}$ are $(\Lambda+\varepsilon, \varepsilon^2/2)$-strongly negatively correlated. But, $\mu(\mathrm{Eq}(f,\varphi)) \geq \Lambda+\varepsilon$ for all $\varphi \in \mathcal{L}$ by definition of $\mathcal{L}$. Also, $\mathrm{Eq}(f,\varphi_1) \cap \mathrm{Eq}(f,\varphi_2) \subseteq \mathrm{Eq}(\varphi_1,\varphi_2)$ so $\mu(\mathrm{Eq}(f,\varphi_1)\cap\mathrm{Eq}(f,\varphi_2)) \leq \Lambda$ for all distinct $\varphi_1,\varphi_2 \in \mathcal{L}$. By Lemma 5.2.2, we find that $|\Phi| \leq \frac{1}{2\varepsilon^2}+1$. □

Corollary 5.2.4. *Let $\mathfrak{D}$ be a class of pairs of groups. If $\ell(\mathrm{aHom}(G,H), f, \Lambda_{G,H}+\varepsilon) = \mathrm{poly}(1/\Lambda, 1/\varepsilon)$ for all $(G,H) \in \mathfrak{D}$, $f\colon G \to H$, and $\varepsilon > 0$, then $\mathfrak{D}$ is CombEcon.*

The result also holds with Hom in place of aHom.

5.2.3 Bucket splitting: sphere packing argument

We apply strong negative correlation (Lemma 5.2.2) to agreement sets, in order to bound the size of a set of homomorphisms, each with high agreement with the received word but also with low pairwise agreement. This will serve as our base tool to split $\mathcal{L}(\mathrm{aHom}(G,H), f, \Lambda+\varepsilon)$ into "buckets."

Lemma 5.2.5 (Sphere packing bound)**.** *Let G be a finite group, H a group, and $\varepsilon > 0$. Let $f\colon G \to H$ be the received word. Let $\Psi \subseteq \mathcal{L}(\mathrm{aHom}(G,H), f, \Lambda+\varepsilon)$ be a subset of the list. Suppose that Ψ is maximal under the condition that its members have low pairwise agreement, i.e., $\mathrm{agr}(\psi_1,\psi_2) \leq \Lambda^2$ for all distinct $\psi_1,\psi_2 \in \Psi$. Then, the size of Ψ can be bound by*

$$|\Psi| \leq \frac{1}{4(\Lambda+\varepsilon)\varepsilon} + 1. \tag{5.3}$$

Notice that the result also holds with Hom in place of aHom, as Hom ⊂ aHom.

Proof. The sets $\mathrm{Eq}(f,\psi)$ for $\psi \in \Psi$ are $(\Lambda+\varepsilon, (\Lambda+\varepsilon)\varepsilon)$-strongly negatively correlated, so the result follows by Lemma 5.2.2. □

Remark 5.2.6. While Lemma 5.2.5 applies to all groups, it is an existential result. We cannot find the homomorphisms chosen in Ψ, so this is the segment of the CombEcon proof that cannot be translated into an algorithmic method. Imposing "shallow random generation" assumptions on the domain (see the next chapter) will allow us to bypass this issue and arrive at amenable buckets.

The set Ψ will label the buckets that we split $\mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon)$ into. Each bucket, denoted $\Phi_{\psi,\lambda}$, will contain the sphere centered at the homomorphism $\psi \in \Psi$ with radius $(1-\lambda)$. We introduce notation then formally state the bucket splitting.

Notation 5.2.7 ($\Phi_{\psi,\lambda}$)**.** Let G be a finite group, H a group, $\psi \in \text{aHom}(G,H)$, $f: G \to H$, and $\varepsilon > 0$. Let $0 \leq \lambda \leq 1$. Denote the subset of $\mathcal{L} = \mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon)$ that has λ-agreement with ψ by

$$\Phi_{\psi,\lambda} := \{\varphi \in \mathcal{L} : \text{agr}(\varphi,\psi) > \lambda\}.$$

Lemma 5.2.8 (Bucket-splitting lemma)**.** *Let G be a finite group, H a group, $f: G \to H$, $\psi \in \text{aHom}(G,H)$, and $\varepsilon > 0$. Then, there exists a subset $\Psi \subset \mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon)$, with size $|\Psi| \leq \frac{1}{4(\Lambda+\varepsilon)\varepsilon} + 1$, that satisfies*

$$\mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon) \subseteq \bigcup_{\psi\in\Psi} \Phi_{\psi,\Lambda^2}.$$

Proof. Let $f: G \to H$. Let $\Psi \subseteq \mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon)$ be a maximal subset satisfying the conditions of Lemma 5.2.5, i.e., maximal under the conditions that distinct $\psi_1, \psi_2 \in \Psi$ have small agreement $\text{agr}(\psi_1,\psi_2) \leq \Lambda^2$. Lemma 5.2.5 concludes the desired bound on $|\Psi|$. By the maximality of Ψ, any $\varphi \in \mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon) > \Lambda^2$ has high agreement $\text{agr}(\varphi,\psi)$ with some homomorphism $\psi \in \Psi$. Lemma follows. □

5.2.4 Further bucket splitting: localizing to subgroups

We present a method to further split the buckets Φ_{ψ,Λ^2} (found in Lemma 5.2.8 of Section 5.2.1) based on the location of agreement with ψ. These "sub-buckets" consist of homomorphisms that all agree on a low-depth subgroup. The size of the sub-buckets can then be bounded using the depth of the label subgroup. This approach depends very little on the codomain H and will be used to prove that alternating groups are universally CombEcon.

The bucket Φ_ψ will be split according to the location of agreement with ψ, with sub-buckets labeled by subgroups K. We introduce notation for the sub-buckets.

Notation 5.2.9 ($\Phi_{\psi,K}$)**.** Let G be a finite group, H a group, $\psi \in \mathrm{aHom}(G,H)$, $f : G \to H$, and $\varepsilon > 0$. Let $K \leq G$ be a subgroup. Denote the subset of $\mathcal{L} = \mathcal{L}(\mathrm{aHom}(G,H), f, \Lambda + \varepsilon)$ that agrees with ψ on K by

$$\Phi_{\psi,K} = \{\varphi \in \mathcal{L} : K \leq \mathrm{Eq}(\varphi,\psi)\}. \tag{5.4}$$

Next we define starting sets, sets of subgroups, which we use to label the sub-buckets we would like to split one bucket $\Phi_{\psi,\lambda}$ into. Intuitively, a set $\mathcal{S}$ of subgroups is a starting set if the upper range of the subgroup lattice of G contains only supergroups of elements in $\mathcal{S}$.

Definition 5.2.10 ((G,λ)-starting-set)**.** Let $\mathcal{S}$ be a set of subgroups of G. Let $\lambda \in (0,1)$. We say that $\mathcal{S}$ is a (G,λ)**-starting-set** if

$$(\forall K \leq G)(\mu_G(K) > \lambda \Rightarrow (\exists S \in \mathcal{S})(S \leq K)).$$

Remark 5.2.11. let G be a finite group, H a group, $\psi \in \mathrm{aHom}(G,H)$, $f : G \to H$, and $\varepsilon > 0$. Let $0 \leq \lambda \leq 1$. If $\mathcal{S}$ is a (G,λ)-starting set, then

$$\Phi_{\psi,\lambda} = \bigcup_{K \in \mathcal{S}} \Phi_{\psi,K}.$$

Combining a (G, λ)-starting set $\mathcal{S}$ with a subset $\Psi \subset \mathcal{L}(\mathrm{aHom}(G, H), f, \Lambda + \varepsilon)$ found by Lemma 5.2.8, we find that

$$\mathcal{L}(\mathrm{aHom}(G, H), f, \Lambda + \varepsilon) \subset \bigcup_{\psi \in \Psi} \bigcup_{K \in \mathcal{S}} \Phi_{\psi, K}. \tag{5.5}$$

We will use Equation (5.5) to bound list size. Lemma 5.2.8 bounds $|\Psi|$. The next section provides tools to bound $|\Psi_{\psi,K}|$ using the depth of K. Then, a bound on the size of a $(G, \Lambda^2_{G,H})$-starting set would suffice to provide list-size bounds for $\mathrm{aHom}(G, H)$.

5.2.5 Bounding list size via subgroup depth

In this section, we will bound $|\Phi_{\psi,K}|$. We do this by choosing a few random elements from G then by considering the probability that they along with K generate a large subgroup of G (Lemma 5.2.12). The number of random elements is bounded by the subgroup depth of K.

First, we show how the probability that random elements generate a large subgroup can be used to bound $|\Phi_{\psi,K}|$.

Lemma 5.2.12. *Let G be a finite group, H a group, $K \leq G$ a subgroup, $f \colon G \to H$, $\psi \in \mathrm{aHom}(G, H)$, $d \in \mathbb{N}$ and $\varepsilon, \eta > 0$. Suppose that for every subset $S \subseteq G$ satisfying $\mu_G(S) > \Lambda + \varepsilon$ we have*

$$\Pr_{g_1, \ldots, g_d \in G}[g_1, \ldots, g_d \in S \text{ and } \mu(\langle K, g_1, \ldots, g_d \rangle) > \Lambda] > \eta. \tag{5.6}$$

Then, $|\Phi_{\psi,K}| \leq 1/\eta$, where $\Phi_{\psi,K}$ is as defined in Notation 5.2.9.

We intuitively argue as follows. The assumption guarantees that the k-wise products $\mathrm{Eq}(f, \varphi)^k$ (for φ in the list) occupies a "large" portion of the the k-tuples $(g_1, \ldots, g_k)$ that generate large subgroups, i.e., $\mu(\langle g_1, \ldots, g_k \rangle) > \Lambda$. Large is measured with respect to density

in G^k. Since Λ is defined using equalizers between distinct homomorphisms, $\mathrm{Eq}(f,\varphi)^k$ must be disjoint within these k-tuples. This bounds the size of the list.

Proof. Pick $g_1, \ldots, g_d$ be independently and uniformly from G. For $\varphi \in \Phi_{\psi,K}$, let E_φ be the event that both $g_1, \ldots, g_d \in \mathrm{Eq}(f,\varphi)$ and $\mu(\langle K, g_1, \ldots, g_d\rangle) > \Lambda$. Since the subset $\mathrm{Eq}(f,\varphi) \subseteq G$ satisfies $\mu(\mathrm{Eq}(f,\varphi)) > \Lambda + \varepsilon$, we have by assumption that $\Pr[E_\varphi] > \eta$ for all $\varphi \in \Phi_{\psi,K}$.

We show that the events E_φ are mutually exclusive. For $\varphi_1, \varphi_2 \in \Phi$, suppose that E_{φ_1} and E_{φ_2} overlap, i.e., there exist $g_1, \ldots, g_d \in E_{\varphi_1} \cap E_{\varphi_2}$. Then, $g_i \subset \mathrm{Eq}(f,\varphi_1) \cap \mathrm{Eq}(f,\varphi_2)$ so $\varphi_1(g_i) = \varphi_2(g_i)$ for all $1 \leq i \leq d$. Recall that $K \subset \mathrm{Eq}(\varphi_1,\varphi_2)$ by definition of $\Phi_{\psi,K}$. So, $\langle K, g_1, \ldots, g_d\rangle \subseteq \mathrm{Eq}(\varphi_1,\varphi_2)$ and $\mu(\mathrm{Eq}(\varphi_1,\varphi_2)) > \Lambda$, so $\varphi_1 = \varphi_2$.

We have found that $\sum_{\varphi\in\Phi_{\psi,K}} \Pr(E_\varphi) \leq 1$ and $\Pr[E_\varphi] > \eta$ for all $\varphi \in \Phi_{\psi,K}$. Conclusion follows. □

Now, we give a bound on the probability assumed in Lemma 5.2.12 in terms of the depth of the subgroup K.

Lemma 5.2.13. *Let $0 \leq \lambda < 1$. Let G be a finite group, $K \leq G$ a subgroup, and $S \subseteq G$ a subset. Suppose that $\mu_G(S) > \lambda$. We denote $\varepsilon = \mu(S) - \lambda$ and $d = \mathrm{depth}_G(K)$. Then,*

$$\Pr_{s_1,\ldots,s_d\in S}[\mu(\langle K, s_1, \ldots, s_d\rangle) > \lambda] \geq \left(\frac{\varepsilon}{\lambda+\varepsilon}\right)^d.$$

It follows that

$$\Pr_{g_1,\ldots,g_d\in G}[g_1, \ldots, g_d \in S \text{ and } \mu(\langle K, g_1, \ldots, g_d\rangle) > \Lambda] \geq \varepsilon^d.$$

This is proved by repeated application of Bayes' rule.

Proof. Pick $s_1, s_2, s_3, \ldots$ independently and uniformly from S. We proceed by induction on $|G : K|$.

Suppose $\mu(K) > \lambda$. Then, $\Pr[\mu(\langle K, s_1, \ldots, s_d\rangle) > \lambda] = 1$.

Suppose $\mu(K) \leq \lambda$. Then, with probability $\dfrac{\mu(S \smallsetminus K)}{\mu(S)} \geq \dfrac{\varepsilon}{\lambda + \varepsilon}$, we have that $s_1 \notin K$, so $\langle K, s_1\rangle > K$, and $\mathrm{depth}_G\langle K, s_1\rangle = d - 1$. By in the induction hypothesis,

$$\Pr[\mu(\langle K, s_1, \ldots, s_d\rangle) > \lambda] \geq \Pr[\mu(\langle K, s_1, \ldots, s_d\rangle) > \lambda \mid s_1 \notin K] \cdot \Pr[s_1 \notin K] \tag{5.7}$$

$$\geq \left(\frac{\varepsilon}{\lambda + \varepsilon}\right)^{d-1} \cdot \left(\frac{\varepsilon}{\lambda + \varepsilon}\right). \tag{5.8}$$

□

As a corollary to Lemmas 5.2.12 and 5.2.13, we bound $|\Phi_{\psi,K}|$ using the depth of K.

Corollary 5.2.14 (Bucket bound)**.** *Let G be a finite group, H a group, $K \leq G$ a subgroup, $f\colon G \to H$, and $\varepsilon > 0$. Then,*

$$|\Phi_{\psi,K}| \leq 1/\varepsilon^{\mathrm{depth}_G(K)}.$$

5.3 Alternating domain

In this section, we will find that homomorphism codes with alternating domain are CombEcon. The exact constant is stated in Theorem 5.3.3. We remark that the constant in the poly$(1/\varepsilon)$-bound on list size can be improved using the SRG methods of Section 6.2. The proof here uses the strategy outlined in Remark 5.2.11 and in particular Equation (5.5). A small starting set is found using Jordan-Liebeck (Theorem 2.4.2), then a bound on the depth of its subgroups follows a previous result on length of subgroup chains in symmetric groups [Bab86].

Section 5.3.1 presents background on the structure of alternating groups. Section 5.3.2 proves the claim that $\mathfrak{Alt} \times \mathfrak{Groups}$ is CombEcon. Section 5.3.3 addresses the list-decoding

radius of homomorphism codes with alternating domain, by exhibiting a "blowup" in list size when the agreement is exactly Λ, or when the radius is $(1-\Lambda)$.

5.3.1 Background on structure of alternating groups

For a set Ω, let $\mathrm{Alt}(\Omega)$ denote the alternating group on Ω. Similarly, let $\mathrm{Sym}(\Omega)$ denote the symmetric group on Ω. We denote $A_n = \mathrm{Alt}([n])$ and $S_n = \mathrm{Sym}([n])$.

Let $G \leq S_n$. For $\pi \in G$, we denote $x^\pi := \varphi(\pi)(x)$. For $x \in [n]$, denote by $G_x = \{\pi \in G \mid x^\pi = x\}$ the point stabilizer of x. Let $\Delta \subseteq [n]$. Denote by $G_{(\Delta)} = \{\pi \in G \mid (\forall x \in \Delta)(x^\pi = x)\}$ the pointwise stabilizer of Δ. Denote by $G_{\{\Delta\}} = \{\pi \in G \mid \Delta^\pi = \Delta\}$ the setwise stabilizer of Δ, where $\Delta^\pi := \{x^\pi : x \in \Delta\}$.

Below is a reult of [Bab86] that describes the length of subgroup chains. This will be used to bound the depth of large subgroups, in order to apply Lemma 5.2.13.

Theorem 5.3.1 (Babai)**.** *The length of any subgroup chain in S_n is at most $2n-3$.*

Corollary 5.3.2. *The length of every subgroup chain between A_{n-k} and S_n is at most $2k-1$.*

5.3.2 Alterating groups are universally CombEcon

We state the universal CombEcon result for alternating groups with a specific constant.

Theorem 5.3.3. *For every group H, $n \geq 10$ and $\varepsilon > 0$, we find that*

$$\ell(\mathrm{Hom}(A_n, H), \Lambda_{A_n,H} + \varepsilon) \leq 1/\varepsilon^{16}.$$

We follow the strategy of Remark 5.2.11. What remains is to find a small $(A_n, \Lambda^2_{A_n,H})$-starting set that contains subgroups of small index, for all groups H. Recall from Lemma 3.2.15 that $\Lambda_{A_n,H} \geq 1/\binom{n}{2}$. Since any (A_n, λ_1)-starting set is also an (A_n, λ_2)-starting set if $\lambda_1 < \lambda_2$, finding an $(A_n, 1/\binom{n}{2}^2)$-starting set suffices.

We will use as our starting set the subgroups A_{n-5} of A_n found by fixing five points.

Notation 5.3.4 ($\mathcal{S}(n)$)**.** For every $n \in \mathbb{N}$, define the set $\mathcal{S}(n)$ of subgroups of A_n by

$$\mathcal{S}(n) := \{(A_n)_{(\Delta)} : \Delta \subset [n], |\Delta| = 5\}.$$

That $\mathcal{S}(n)$ is a sufficient starting set follows immediately from Jordan-Liebeck and the order of alternating groups.

Corollary 5.3.5. *If $n \geq 10$, the set $\mathcal{S}(n)$ of subgroups of A_n is an $(A_n, 1/\binom{n}{2}^2)$-starting set.*

The depth of subgroups in $\mathcal{S}(n)$ is bound by Corollary 5.3.2.

Corollary 5.3.6. *If $K \in \mathcal{S}(n)$, then* $\mathrm{depth}_{A_n}(K) \leq 8$.

Now, we have all the tools to conclude the list-size bound of Theorem 5.3.3. We combine arguments presented above.

Proof of Theorem 5.3.3. Fix $f : G \to H$. Let $\Psi \subseteq \mathrm{aHom}$ be as found by Lemma 5.2.8. Let $\mathcal{S}(n)$ be as defined in Notation 5.3.4. By Remark 5.2.11, we have that

$$|\mathcal{L}(\mathrm{aHom}(G,H), f, \Lambda + \varepsilon)| \leq |\Psi| \cdot |\mathcal{S}(n)| \cdot \max_{\substack{\psi \in \Psi \\ K \in \mathcal{S}(n)}} |\Phi_{\psi,K}|.$$

We bound each term. By Lemma 5.2.8, we have $|\Psi| \leq \left(\frac{1}{4(\Lambda+\varepsilon)\varepsilon} + 1\right)$. Subgroups in $\mathcal{S}(n)$ can be labeled by 5-tuples in $[n]$, so $|\mathcal{S}(n)| = \binom{n}{5}$. We may assume that $\varepsilon^2 < 2\Lambda$ by Lemma 5.2.3, so that $|\mathcal{S}| = \binom{n}{5} < (\binom{n}{2}/2)^3 = (\Lambda/2)^3 < 1/\varepsilon^6$. Corollary 5.3.6 showed that $\mathrm{depth}_{A_n}(K) \leq 8$ for all $K \in \mathcal{S}(n)$, so by Corollary 5.2.14, we find that $|\Phi_{\psi,K}| \leq 1/\varepsilon^8$. Combining, this gives the exponent of 16 as claimed. □

5.3.3 Upper bound on list-decoding radius

We showed in Section 5.3.2 that $\mathfrak{Alt} \times \mathfrak{Groups}$, and all of its subclasses, have list-decoding radius greater than $1 - (\Lambda + \varepsilon)$ for all $\varepsilon > 0$.

In contrast, $\mathfrak{Alt} \times \mathfrak{Groups}$ and many of its subclasses have list-decoding radius at most $1 - \Lambda$. In this section, we demonstrate such a subclass. The number of homomorphisms within a closed ball of radius $1 - \Lambda$ of a received word will be exponential in $\log|G|$ and $\log|H|$. We note that $|H| \geq |G|$ unless $\Lambda = 0$.

Proposition 5.3.7. *For any n, and either $\lambda \in \{1/n, 1/\binom{n}{2}\}$, there exists a group H_n such that $\Lambda_{A_n,H_n} = \lambda$ and*

$$\ell(\mathrm{Hom}(A_n, H_n), \Lambda) = 2^{\Omega(n)} \geq 2^{\Omega\left(\sqrt[3]{\log|H|}\right)}. \tag{5.9}$$

Proof. Suppose $\lambda = 1/n$. Let $H_n = A_{n+1}^k$, the direct product of k copies of A_{n+1}. Then $\Lambda_{A_n,H_n} = 1/n$. Let $f\colon A_n \to H_n$ by $f(g) = (g, \ldots, g)$, the diagonal identity map, where A_n is embedded in A_{n+1}. For nonempty $S \subseteq [n]$ and $j \in [n]$, let $h = h(S, j) = (h_1, \ldots, h_k) \in H_n$, where h_i is the transposition $(j, n+1)$ if $i \in S$ and 1 otherwise. For each such h, let $\varphi_h \in \mathrm{Hom}(A_n, H_n)$ be given by $\varphi_h(g) = h^{-1}f(g)h$. Each φ_h has agreement $\mathrm{agr}(\varphi_h, f) = 1/n = \Lambda$ with f. There are $n(2^k - 1)$ such h, so $\ell(\mathrm{Hom}(A_n, H_n), \Lambda) \geq n(2^k - 1)$.

Suppose $\lambda = 1/\binom{n}{2}$. Let $H_n = A_n^k$. Then, $\Lambda_{A_n,H_n} = 1/\binom{n}{2}$. Let $f\colon A_n \to H_n$ by $f(g) = (g, \ldots, g)$, the diagonal identity map. For nonempty $S \subseteq [n]$ and $\tau \in S_n$ is a transposition, let $h = h_{S,\tau} = (h_1, \ldots, h_k) \in A_n^k$, where $h_i = \tau$ if $i \in S$ and 1 otherwise. For each such h, let $\varphi_h \in \mathrm{Hom}(A_n, H_n)$ be given by $\varphi_h(g) = h^{-1}f(g)h$. Each such φ_h has agreement $\mathrm{agr}(\varphi_h, f) = 1/\binom{n}{2}$. There are $\binom{n}{2}(2^k - 1)$ such h, so $\ell(\mathrm{Hom}(A_n, H_n), \Lambda) \geq \binom{n}{2}(2^k - 1)$. □

We remark in particular that $\ell(\mathrm{Hom}(A_n, H), \Lambda_{A_n,H})$ is not bounded as a function of n for a wide variety of classes of H.

CHAPTER 6

ALGORITHMIC LIST-DECODABILITY

6.1 Introduction

This chapter addresses our algorithmic approaches to list-decoding.

Section 6.2 introduces shallow random generating (SRG) groups, then proves that they are universally CombEcon and CertEcon. This gives a separate proof that alternating groups are universally CombEcon. More notably, however, is that the existence of a $\mathcal{W}$-certificate-list-decoder for all homomorphism codes with SRG domain (see Section 6.2.5), where $\mathcal{W}$ consists of certificates γ with $\mu(\langle \mathrm{dom}(\gamma) \rangle) > \Lambda_{G,H}$ (Condition (2) discussed in Section 3.2.3). See Section 3.1.5 for $\mathcal{W}$-certificate terminology.

Section 6.3 gives an overview of currently existing tools that prove homomorphism codes are AlgEcon: those in prior works, the extension principles in Chapter 4, and the approach that bridges the gap between CertEcon and AlgEcon (as outlined in Section 3.1.5, finding an efficient $\mathcal{W}$-subword extender would show AlgEcon of homomorphism codes with alternating domain).

6.1.1 Structure of chapter

In Section 6.2, we prove results about SRG groups (defined in Section 3.2.2), group where few random elements tend to generate a shallow (low depth) subgroup. We will show that alternating groups are SRG (Section 6.2.1). SRG groups are universally CombEcon and CertEcon, as proved in Sections 6.2.4 and 6.2.5. This is accomplished by defining a technical "subset-generated" condition (Section 6.2.2) then proving that SRG groups satisfy this condition.

Section 6.3 discuss the algorithmic methods available to list decode homomorphism codes. Section 6.3.1 gives an overview of the three main methods at our disposal, including a

discussion of their access assumptions. Sections 6.3.2 and 6.3.3 discuss the gap between certificate-list-decoders and list-decoders; HomExt is exactly subword extension for Hom (Section 6.3.2), whereas HomExt can solve subword extension for aHom (Section 6.3.3). Section 6.3.4 discusses the necessity and perks of knowing or lower bounding Λ. It turns out that a stronger version (HomExt012) of Homomorphism Extension helps improve lower bounds (Section 6.3.5). Section 6.3.6 discusses how to establish the value of Λ in the algorithm of prior works.

Homomorphism Extension is the subject of Chapter 7.

6.2 Shallow random generation

Let $k, d \in \mathbb{N}$. Recall Definition 3.2.4: a finite group G is (k, d)-**shallow generating** if

$$\Pr_{g_1,\ldots,g_k \in G}[\text{depth}(\langle g_1, \ldots, g_k \rangle) > d] < (\Lambda^*_G)^k. \tag{6.1}$$

6.2.1 Alternating groups are SRG

In this subsection, we prove that $\mathfrak{Alt}$ is SRG.

Theorem 6.2.1. *The class of alternating groups is SRG. In fact, for all sufficiently large n, the alternating group A_n is $(2, 6)$-shallow generating.*

Consequences. Before proving Theorem 6.2.1, we first discuss its consequences.

From Theorem 6.2.10, we find that $\ell(\text{aHom}(A_n, H), \Lambda + \varepsilon) < 1/\varepsilon^9$ for all $H \in \mathfrak{Groups}$. We remark that the constant 9 can be improved to 7. This can be accomplished by going through the proof with a "depth$_\Lambda$" notion instead of depth$_G$ as written. (This improves the resulting $(8, \Lambda, 7)$-generated claim from Section 6.2.3 to $(6, \Lambda, 5)$-generated.) This depth$_\Lambda(K)$ refers to maximal length of a subgroup chain from K to a subgroup of density greater than Λ.

By Theorem 6.2.11 and using the certificate-list-decoder CertLD, $\mathfrak{Alt} \times \mathfrak{Groups}$ is strong certificate-list-decodable using $O(\ln(1/\varepsilon)/\varepsilon^9)$ queries.

If $H = S_m$ and $m < 2^n/\sqrt{1.6n}$, then calls of HomExt in CertToAlg can be executed in poly(n, m)-time, as shown in the Appendix. So, CertToAlg has a running time of poly(n, m). This gives a list decoder for this case, but running in poly$(1/\varepsilon, n, m)$-time while using poly$(1/\varepsilon)$-queries.

Proof. Towards proving Theorem 6.2.1, we first present Theorem 6.2.2 from [Bab89, Theorem 1.5]. It says that two elements of the alternating group have extremely high probability of generating a shallow subgroup.

First, we see that two elements have extremely high probability of generating a large subgroup. This event is denoted $E(n, k)$ in the statement.

Theorem 6.2.2 (Babai)**.** *Let π, σ be a pair of independent uniform random elements from S_n. For $0 \leq k \leq n/3$, let $E(n, k)$ denote the following event: The subgroup $H = \langle \pi, \sigma \rangle$ acts as S_r or A_r on r elements of the permutation domain for some $r \geq n - k$. Then,*

$$\Pr(E(n,k)) = 1 - \binom{n}{k+1}^{-1} + O\left(\binom{n}{k+2}^{-1}\right). \tag{6.2}$$

The constant implied by the big-O notation is absolute.

Remark 6.2.3. Suppose that we choose π and σ from A_n (instead of S_n) in Theorem 6.2.2. The same conclusion is still true. However, using only Theorem 6.2.2 as justification, the conclusion is slightly weaker – there will be a coefficient of 4 in front of $\binom{n}{k+1}^{-1}$. In our application, this coefficient makes no difference to our argument.

Theorem 5.3.1 [Bab86] (stated in Section 5.3.1) bounds the depth of large subgroups. We immediately find that two elements generate a shallow subgroup with high probability.

Claim 6.2.4. *Let E, k, π, σ be defined as in Theorem 6.2.2. If $E(n,k)$ occurs, then* $\mathrm{depth}_{A_n}(\langle \pi, \sigma \rangle) \leq 2k - 2$.

Proof. Let $K = \langle \pi, \sigma \rangle$. If $E(n,k)$ occurs, then K acts as S_r or A_r on some a subset of r elements of $[n]$, for $r \geq n - k > n/2$. So, $A_r \leq K$. By Corollary 5.3.2, we find that $\mathrm{depth}_{A_n}(K) \leq \mathrm{depth}_{A_n}(A_r) \leq 2k - 2$.

□

Now we can prove Theorem 6.2.1. By the above discussion which relies on the main results of [Bab86, Bab89], two elements generate a depth-5 subgroup.

Proof of Theorem 6.2.1. By Theorem 6.2.2 and Claim 6.2.4, we find for large n that

$$\Pr_{\pi,\sigma \in A_n}[\mathrm{depth}_{A_n}(\langle \pi, \sigma \rangle) > 6] \leq \Pr_{\pi,\sigma \in A_n}[\neg E(n,4)] \leq \frac{4}{\binom{n}{5}} \leq \frac{1}{\binom{n}{2}^2}.$$

It follows that A_n is $(2,6)$-shallow generating. □

6.2.2 Subset-generation

In this section we define a useful technical property of groups, which SRG groups have (shown in the next section). Our SRG implies CombEcon and SRG implies CertEcon results will be proven by assuming this property.

Definition 6.2.5 ((k,λ,c)-subset-generated)**.** Let G be a finite group, k a nonnegative integer, $0 \leq \lambda < 1$, and $c \geq 0$. We say that G is (k,λ,c)**-subset-generated** if, for all subsets $S \subseteq G$ with $\mu(S) > \lambda$, we have that

$$\Pr_{s_1,\ldots,s_k \in S}[\mu(\langle s_1, \ldots, s_k \rangle) > \lambda] \geq \left(1 - \frac{\lambda}{\mu(S)}\right)^c, \tag{6.3}$$

where $s_1, \ldots, s_k$ are chosen independently and uniformly from S.

Note that, if we define $\varepsilon = \mu(S) - \lambda$, then $1 - \frac{\lambda}{\mu(S)} = \frac{\varepsilon}{\lambda+\varepsilon}$, matching the expression of Lemma 5.2.13.

We say that G is (k, λ, c)-**affine-generated** if it satisfies Definition 6.2.5 but with $\langle s_1, \ldots, s_k \rangle$ replaced by $\langle s_1, \ldots, s_k \rangle_{\mathsf{aff}}$.

We will make a few remarks on these definitions below, but first we define KLC classes of groups.

Definition 6.2.6 (KLC)**.** Let $\mathfrak{G}$ be a class of finite groups. We say that $\mathfrak{G}$ is **KLC** if there exists a positive integer k and a constant $c > 0$ such that, for all $G \in \mathfrak{G}$ and for all groups H, we have that G is $(k, \Lambda_{G,H}, c)$-subset-generated.

The notion of "KLC-coset" can be defined analogously. But, according to Remark 6.2.7 (c) and Lemma 6.2.8 below, the two conditions are equivalent conditions on a class of groups.

If a class of groups is KLC then it is universally CombEcon and CertEcon (Sections 6.2.4 and 6.2.5 These implications follow also with ε^c in Definition 6.2.5 instead of $\left(\frac{\varepsilon}{\lambda+\varepsilon}\right)^c$, but with a worse constant.

We make a few remarks on the definitions of (k, λ, c)-subset-generated groups.

Remark 6.2.7. (a) For every $k \geq 1$ and $c \geq 0$, the class $\mathfrak{Groups}$ of all finite groups is $(k, 0, c)$-subset-generated.

(b) Classes of (k, λ, c)-subset-generated groups are monotone in both k and c. More specifically, for $k' > k$ and $c' > c$, if G is (k, λ, c)-subset-generated, then G is also (k', λ, c)-subset-generated and (k, λ, c')-subset-generated.

(c) If G is (k, λ, c)-affine-generated, then it is (k, λ, c)-subset-generated. The other direction is described in the next lemma.

Lemma 6.2.8. *Let G be a group, k a nonnegative integer, $0 < \lambda < 1$, and $c \geq 0$ such that G is (k, λ, c)-subset-generated. Then, G is $(k+1, \lambda, c)$-affine-generated.*

Proof. This lemma follows from the next inequality, which holds for all $0 < \lambda < 1$, $0 < \tau < 1$, and $k \in \mathbb{N}$:

$$\min_{S \subset G, \mu(S)=\lambda} \Pr_{s_1,\ldots,s_k \in S} [\mu(\langle s_1, \ldots, s_k \rangle) > \tau] \leq \min_{S \subset G, \mu(S)=\lambda} \Pr_{s_1,\ldots,s_{k+1} \in S} [\mu(\langle s_1, \ldots, s_{k+1} \rangle_{\mathsf{aff}}) > \tau].$$

We check this equation below. Denote the quantity of the left hand side by M. Let $S \subset G$ satisfy $\mu(S) = \lambda$. Then,

$$\begin{aligned}
\Pr_{s_1,\ldots,s_{k+1} \in S} [\mu(\langle s_1, \ldots, s_{k+1} \rangle_{\mathsf{aff}}) > \tau] &= \Pr_{s_{k+1} \in S} \left[\Pr_{s_1,\ldots,s_k \in S} [\mu(\langle s_1, \ldots, s_{k+1} \rangle_{\mathsf{aff}}) > \tau] \right] \\
&= \Pr_{s_{k+1} \in S} \left[\Pr_{s_1,\ldots,s_k \in S} \left[\mu\left(\langle s_1 s_{k+1}^{-1}, \ldots, s_k s_{k+1}^{-1} \rangle \right) > \tau \right] \right] \\
&= \Pr_{s_{k+1} \in S} \left[\Pr_{s_1,\ldots,s_k \in S \cdot s_{k+1}^{-1}} [\mu(\langle s_1, \ldots, s_k \rangle) > \tau] \right] \\
&\geq \Pr_{s_{k+1} \in S} [M] = M
\end{aligned}$$

□

6.2.3 SRG implies subset-generation

We prove that SRG implies KLC.

Theorem 6.2.9 (SRG implies KLC)**.** *If a class $\mathfrak{G}$ of groups is SRG, then $\mathfrak{G}$ is KLC.*

In particular, let G be a finite group, $k, d \in \mathbb{N}$, and $\lambda > 0$. If G is (k, d)-shallow generating, then G is $(k + d, \lambda, 1 + d)$-subset generated for all $\lambda \geq \Lambda_G^*$.

Proof. All groups are trivially $(1, 0, 1)$-subset-generated, which covers the case where $\lambda = 0$.

Let $\lambda > 0$. By the assumptions, we know that

$$\Pr_{g_1,\ldots,g_k \in G} [\mathrm{depth}(\langle g_1, \ldots, g_k \rangle) > d] < \lambda^k. \tag{6.4}$$

We check the definition of $(k+d, \lambda, 1+d)$-subset-generated.

Let $S \subset G$ be such that $\mu(S) > \lambda$ and let $\varepsilon = \mu(S) - \lambda$. We will pick a k-tuple $\mathbf{g} = (g_1, \ldots, g_k)$ and a d-tuple $\mathbf{s} = (s_1, \ldots, s_d)$ from S. We write $\langle \mathbf{g} \rangle$ to mean $\langle g_1, \ldots, g_k \rangle$ and $\langle \mathbf{s} \rangle$ similarly.

Observe that

$$\Pr_{\mathbf{g} \in S^k, \mathbf{s} \in S^d} [\mu(\langle \mathbf{g}, \mathbf{s} \rangle) > \lambda] \geq \Pr_{\mathbf{s} \in S^d} [\mu(\langle \mathbf{g}, \mathbf{s} \rangle) > \lambda \mid \operatorname{depth}(\langle \mathbf{g} \rangle) \geq d] \cdot \Pr_{\mathbf{g} \in S^k} [\operatorname{depth}(\langle \mathbf{g} \rangle) \geq d].$$

We bound the two components of the right hand side separately. When we drop the subscript on Pr, that means the elements are chosen uniformly from G. First, we consider the second component.

$$\begin{aligned}
\Pr_{\mathbf{g} \in S^k} [\operatorname{depth}(\langle \mathbf{g} \rangle) \geq d] &= \Pr_{\mathbf{g} \in G^k} [\operatorname{depth}(\langle \mathbf{g} \rangle) \geq d \mid \mathbf{g} \in S^k] \\
&= \frac{\Pr[\mathbf{g} \in S^k \text{ and } \operatorname{depth}(\langle \mathbf{g} \rangle) \geq d]}{\Pr[\mathbf{g} \in S^k]} \\
&\geq \frac{\Pr[\mathbf{g} \in S^k] + \Pr[\operatorname{depth}(\langle \mathbf{g} \rangle) \geq d] - 1}{\Pr[\mathbf{g} \in S^k]} \\
&> \frac{\mu(S)^k - (\lambda)^k}{\mu(S)^k} = 1 - \left(\frac{\lambda}{\mu(S)} \right)^k \\
&\geq 1 - \frac{\lambda}{\lambda + \varepsilon} = \frac{\varepsilon}{\lambda + \varepsilon}.
\end{aligned}$$

Now, it suffices to show that the first component has probability bounded by $\left(\frac{\varepsilon}{\lambda+\varepsilon} \right)^d$. But, if $\operatorname{depth}(\langle \mathbf{g} \rangle) \geq d$, then it follows from Lemma 5.2.13, with $K = \langle \mathbf{g} \rangle$ and $\lambda = \lambda$, that

$$\Pr_{\mathbf{s} \in S^d} [\mu(\langle \mathbf{g}, \mathbf{s} \rangle) > \lambda \mid \operatorname{depth}(\langle \mathbf{g} \rangle) \geq d] > \left(\frac{\varepsilon}{\lambda + \varepsilon} \right)^d.$$

We conclude that G is $(k+d, \lambda, 1+d)$-subset-generated, or,

$$\Pr_{s_1,\ldots,s_{k+d}\in S}\left[\mu\left(\langle s_1,\ldots,s_{k+d}\rangle\right) > \lambda\right] = \Pr_{\mathbf{g}\in S^k,\mathbf{s}\in S^d}\left[\mu\left(\langle \mathbf{g},\mathbf{s}\rangle\right) > \lambda\right] > \left(\frac{\varepsilon}{\lambda+\varepsilon}\right)^{d+1}.$$

□

6.2.4 SRG implies CombEcon

KLC quickly implies universally CombEcon. The proof is presented here for conceptual clarity; of course, it follows from the "KLC implies universally CertEcon" result (Theorem 6.2.11) in the next section.

Theorem 6.2.10. *If a class $\mathfrak{G}$ is SRG, then $\mathfrak{G}$ is universally CombEcon. More precisely, let k be a non-negative integer and $c > 0$. If G is $(k, \Lambda_{G,H}, c)$-subset-generated and H a group, then $\ell(\mathrm{aHom}(G,H), \Lambda_{G,H}+\varepsilon) \le 1/\varepsilon^{\max\{c,k\}+1}$ for all $\varepsilon \in (0, 1-\Lambda)$.*[1]

The proof uses Lemma 5.2.12 but is conceptually very simple. A t-tuple of elements $s_1,\ldots,s_t \in G$ that generate a subgroup of density $> \Lambda$ cannot lie in the equalizer of distinct affine homomorphisms in $\mathrm{aHom}(G,H)$. The KLC condition gives a density lower bound for the number of such t-tuples for each affine homomorphism in the list. This gives an upper bound for the length of the list.

Proof. Apply Lemma 5.2.12, with $K = 1$ to $\Phi = \mathcal{L}(\mathrm{Hom}(G,H), f, \Lambda+\varepsilon)$. The definition of (k,Λ,c)-generating satisfies the assumptions of Equation (5.6) with $\eta = \left(\frac{\varepsilon}{\Lambda+\varepsilon}\right)^c$. This shows that

$$\ell(\mathrm{Hom}(G,H), \Lambda_{G,H}+\varepsilon) \le \frac{(\Lambda+\varepsilon)^c}{\varepsilon^c(\Lambda+\varepsilon)^k} \le 1/\varepsilon^{\max\{c,k\}}.$$

□

1. The constant in Theorem 6.2.10 can be shown to be $\max\{c, k+1\}$ instead of $\max\{c,k\}+1$, by using a coset version of Lemma 5.2.12 or by Theorem 6.2.11 below.

6.2.5 SRG implies CertEcon

A KLC class of groups is also universally CertEcon. The argument is conceptually similar to that of CombEcon, but we formalize algorithmic issues.

Again, let $\mathcal{W}_\Lambda$ denote the set of $G \rightharpoonup H$ partial maps γ satisfying $\mu(\langle \mathrm{dom}\,\gamma\rangle) > \Lambda_{G,H}$.

Theorem 6.2.11. *If $\mathfrak{G}$ be an SRG class of groups, then $\mathfrak{G}$ is universally strong $\mathcal{W}_\Lambda$-CertEcon. We assume that all groups in $\mathfrak{G}$ are encoded groups, that (nearly) uniform elements of G are provided, and that we have oracle access to the entries of the received word.*

If a partial map γ may extend to some homomorphism and satisfies $\mu(\langle \mathrm{dom}(\gamma)\rangle) > \Lambda$, then γ is a $\mathcal{W}$-certificate. However, $\mathrm{dom}(\gamma)$ may fail to generate the entire group, i.e., $\langle \mathrm{dom}(\gamma)\rangle \lneq G$. Section 6.3.2 will follow the strategy of Section 3.1.5 to find a full local list-decoder, given CertLD (a $\mathcal{W}$-certificate-list-decoder) and a Homomorphism Extension oracle (a $\mathcal{W}$-subword extender).

Domain certificates versus certificates.

We develop some terminology for Theorem 3.2.12, based on the natural idea of generating certificates by querying the received word f. "Domain certificates" are subsets of the domain that define a certificate by restricting f to that set.

Let G and H be groups and $f \in H^G$ be a received word in the codespace of $\mathrm{aHom}(G,H)$. Let $S \subseteq G$ be a subset. Denote by f_S the restriction of f to S, i.e., the $H \rightharpoonup G$ partial map defined on domain S with values $f_S(s) = f(s)$ agreeing with f.

Definition 6.2.12 (Domain certificate)**.** When the code $\mathrm{aHom}(G,H)$ and the received word f are understood, we say that a subset $S \subseteq G$ is a **domain certificate for the code** $\mathrm{aHom}(G,H)$ if the $G \rightharpoonup H$ partial map f_S is a certificate for $\mathrm{aHom}(G,H)$.

For a set $\mathcal{W}$ of $G \rightharpoonup H$ partial maps, we say that S is a **domain $\mathcal{W}$-certificate** if f_S is a $\mathcal{W}$-certificate for $\mathrm{aHom}(G,H)$.

Note that a domain certificate S is a domain $\mathcal{W}_\Lambda$-certificate if and only if $\mu(S) > \Lambda_{G,H}$.

Definition 6.2.13 (Domain-certificate-list)**.** We say that a list Υ of subsets of G is a **domain-certificate-list** for a subset $\mathcal{K} \subseteq \mathrm{aHom}(G, H)$ of affine homomorphisms if Υ contains a domain certificate for each codeword in Υ. We define **domain-$\mathcal{W}$-certificate-lists** similarly.

Domain certificate result.

Now, we can restate the unabridged SRG result (Theorem 3.2.12) in terms of domain certificates.

Theorem 6.2.14 (SRG implies CertEcon, via domain certficates)**.** *Let $k \in \mathbb{N}$ and $c > 0$. Let G be a $(k, \Lambda_{G,H}, c)$-subset-generated group and H a group. Let $f : G \to H$, $\varepsilon > 0$ and $\eta > 0$. Let Υ be a list of $\left\lceil \frac{1}{\varepsilon^b} \ln\left(\frac{1}{\eta\varepsilon^b}\right)\right\rceil$ independently chosen subsets of G, each of size $\max\{c, k\}$. Then, with probability at least $(1 - \eta)$, Υ is a domain-$\mathcal{W}_\Lambda$-certificate-list of $\mathcal{L}(\mathrm{aHom}(G, H), f, \Lambda + \varepsilon)$.*

The proof is delayed to first discuss its implications and access model.

Remark 6.2.15 (Access model)**.** To generate the domain-$\mathcal{W}$-certificate-list, we need access only the domain, only in the ability to generate random elements. No knowledge of H is required. The dependence on H appears only in the $\Lambda_{G,H}$ of the assumption that G is $(k, \Lambda_{G,H}, c)$-subset generated, but the KLC assumption means that G is $(k, \Lambda_{G,H}, c)$-subset generated for every H. Knowledge of $\Lambda_{G,H}$ is also not required.

Theorem 6.2.14 produces domain $\mathcal{W}$-certificates. No work is involved other generating these poly$(1/\varepsilon)$ uniform random elements of G. To then produce actual $\mathcal{W}$-certificates, simply query f on the domain $\mathcal{W}$-certificates (subsets of G), an additional poly$(1/\varepsilon)$ queries to f. Theorem 6.2.11 follows.

Remark 6.2.16 (Amount of work)**.** Theorem 6.2.11 implies a stronger result than strong CertEcon, as only a poly$(1/\varepsilon)$ amount of work is required in the unit cost model (no dependency on $|G|$).[2]

Analysis.

To prove Theorem 6.2.11 we check the definition of domain $\mathcal{W}$-certificate-list. In other words, with probability $(1-\eta)$, for every $\varphi \in \mathcal{L}(\mathrm{aHom}(G,H), f, \Lambda+\varepsilon)$ the list Υ contains a domain $\mathcal{W}$-certificate $S_\varphi \subset G$ for φ.

Observation 6.2.17. *If the conditions $\mu(\langle S\rangle_{\mathsf{aff}}) > \Lambda$ and $S \subset \mathrm{Eq}(\varphi, f)$ are satisfied, then S is a domain $\mathcal{W}$-certificate for φ.*

We prove Theorem 6.2.11 by checking the conditions of Observation 6.2.17.

Proof of Theorem 6.2.11. Let $\mathcal{L} = \mathcal{L}(\mathrm{aHom}(G,H), f, \Lambda+\varepsilon)$. Recall that G is $(k, \Lambda_{G,H}, c)$-subset generated and we denote $b = \max\{c, k+1\}$. Let $\Upsilon = \{S_1, \ldots, S_t\}$ be the list of subsets of G as assumed. Then, $t = \left\lceil \frac{1}{\varepsilon^b} \ln\left(\frac{1}{\eta\varepsilon^b}\right)\right\rceil$ and $|S_i| = b$.

Fix $\varphi \in \mathcal{L}$. Fix $S \in \Upsilon$. We calculate the following.

$$\begin{aligned}
\Pr[S \text{ is a domain } \mathcal{W}\text{-certificate for } \varphi] &\geq \Pr\left[S \subseteq \mathrm{Eq}(\psi, f) \cap \mu\left(\langle S\rangle_{\mathsf{aff}}\right) > \Lambda\right] \\
&= \Pr\left[\mu\left(\langle S\rangle_{\mathsf{aff}}\right) > \Lambda \middle| S \subseteq \mathrm{Eq}(\psi, f)\right] \cdot \Pr[S \subseteq \mathrm{Eq}(\psi, f)] \\
&> \left(\frac{\varepsilon}{\Lambda+\varepsilon}\right)^c \cdot (\Lambda+\varepsilon)^{k+1} \\
&> \varepsilon^b.
\end{aligned}$$

The first inequality follows from Observation 6.2.17, and the third inequality follows from the definition of (k, Λ, c)-subset generated.

2. Two incomparable sufficient conditions for the access model to G are black-box access and polycyclic presentations. In a black-box group, ε-uniform elements can be generated in polynomial time [Bab91]. Given a polycyclic presentation, exactly uniform elements can be generated.

The probability that Υ is not a domain $\mathcal{W}$-certificate-list for $\mathcal{L}$, i.e., there is $\varphi \in \mathcal{L}$ such that Υ contains no domain-$\mathcal{W}$-certificate for φ, is bounded by

$$|\mathcal{L}| \cdot \left(1 - \varepsilon^b\right)^t \leq \frac{1}{\varepsilon^b} \exp\left(-\varepsilon^b \cdot t\right) < \eta.$$

□

6.3 Algorithms and access: HomExt and role of Λ

6.3.1 Overview of algorithms and access issues

We give an overview of the algorithmic tools available to list decode homomorphism codes, with the goal of proving that they are AlgEcon. These are: (1) the list-decoders of prior works (Theorem 6.3.1 below), (2) the extension principles of Chapter 4, and (3) the "certificate-list-decoder along with a subword extender gives a list-decoder" principle of Remark 6.3.13.

Discussing method (3) and issues with finding Λ will constitute the meat of this section.

Prior work

Prior literature [DGKS08, GS14] provided an AlgEcon list-decoder that inductively extends homomomorphisms through a subgroup chain in the domain with prime cyclic factors. Homomorphism codes satisfying the assumptions *must have supersolvable domain.*

To list-decode, a set of homomorphisms is maintained at each step of the subgroup chain. An extension of a homomorphism in the current set to the next subgroup in the chain can be determined by its value on one element. The value on that element is strategically sampled, to find a set of candidate extensions. The set of extensions is pruned to a reasonable size by sampling agreement with f on cosets. This extends the set of homomorphisms to the next subgroup in the chain.

The result is stated more formally below. For this algorithm, it is crucial that a list of

affine homomorphisms are maintained, in order to sample from all cosets.

Theorem 6.3.1. *Let $\mathfrak{D}$ be the class of pairs (G, H), where G is a supersolvable group given by polycyclic presentation with the subgroup chain $1 = G_0 \lhd G_1 \lhd \cdots \lhd G_k = G$ of subgroups with prime cyclic factors, with the primes in decreasing order, and H is a group with black-box access. Suppose that for all pairs $(G, H) \in \mathfrak{D}$ and for all subgroups G_i in the chain (G_i, H) is CombEcon. Then, $\mathfrak{D}$ is AlgEcon.*

The access assumptions are quite strict. The domain G must be given in polycyclic presentation, with the primes ordered. Only black-box access is needed to H. The value of Λ must be known (for pruning) in order to bound the run time and list size.

Bipartite covering method.

Chapter 4 uses the Bipartite Covering Lemma to prove two extension principles for economical list-decoding. First, economical list-decoding and economical mean-list-decoding are equivalent. This equivalence allows us to extend economical list-decoding conclusions to repeated codes. Second, economical list-decoding for homomorphism codes is equivalent whether considering homomorphisms $\mathrm{Hom}(G, H)$ or affine homomorphisms $\mathrm{aHom}(G, H)$.

We discuss the algorithmic issues of this method. The principle assumes a (certificate-)list-decoder for the original class is readily available and returns a (certificate-)list-decoder for the extension class. We know that the extension list is contained in few lists in the original code.

The algorithm simply samples few functions (centers of the list), list decodes those functions using the assumed list-decoders, then outputs the union of these lists (perhaps after translation) as the list for the extension code. This corresponds to running the known list-decoder a manageable number of times. The multiplicative cost is described more precisely in Remark 4.2.14 (nearly linear in the deterioration factor of agreement for mean-lists) and Remark 4.3.20 (nearly linear in the agreement $\Lambda + \varepsilon$ for aHom lists).

The add-on algorithm is so simple that, once the original list-decoder is given, access assumptions are relatively minimal.

- aHom versus Hom: We need only the ability to generate uniform random elements of G. The input radius/agreement of the desired list-decoder for $\mathcal{C}^0$ is used as input to the known list-decoders for $\mathcal{C}$. The work is done by translation of homomorphisms and functions, but this is easy given oracle access to the received word f.

- Mean-lists versus lists: We need (i) knowledge of ε, and (ii) the ability to generate uniform random elements of $\mathscr{F}$, the family of received words.

  A deterioration factor must be added to the (known) input radius ($\mathsf{mindist} - \varepsilon$) of the desired mean-list-decoder, before calling the known list-decoder. This requires knowledge of either ε or $\mathsf{mindist}$, so that the deterioration can be set to be smaller then ε. The received word of the known list-decoder is a sampled uniformly from the family $\mathscr{F}$ of received words for the mean-list-decoder.

Via CertEcon.

This method will be discussed extensively in the rest of this section. Theorem 6.2.11 gives a certificate-list-decoder, whose output we hope to turn into a list, to obtain a true list-decoder. The main observation (Remark 6.3.13) is that a certificate-list can be made into a list by extending every subword (partial map) in the certificate-list, if possible, and deleting the subword if not.

6.3.2 Homomorphism Extension and AlgEcon

In this section we define the Homomorphism Extension Problem (the topic of the next chapter), which bridges from certificate-list-decoding to list-decoding for homomorphism codes.

Following the CertEcon result stated above, we would like to find efficient subword extenders for the case of homomorphism codes, in order to find AlgEcon results. By the results of Section 4.3.3, we will without loss of generality consider homomorphisms in $\mathrm{Hom}(G, H)$ instead of affine homomorphisms in $\mathrm{aHom}(G, H)$. For $\mathrm{Hom}(G, H)$, finding subword extenders exactly amounts to solving the Homomorphism Extension Search Problem, defined below. (This is almost true for $\mathrm{aHom}(G, H)$ as well, addressed in the next section.)

The Homomorphism Extension Problem asks whether a partial map extends to a homomorphism on the whole group. Below, let G and H be groups. As before, let $\Lambda = \Lambda_{G,H}$.

Definition 6.3.2. (Homomorphism Extension, HomExt(G, H))

Instance: A partial map $\gamma : G \rightharpoonup H$.

Solution: A homomorphism $\varphi \in \mathrm{Hom}(G, H)$ that extends γ, i.e., $\varphi|_{\mathrm{dom}\,\gamma} = \gamma$.

The **Homomorphism Extension Decision Problem** asks whether a solution exists. The **Homomorphism Extension Search Problem** asks whether a solution exists and, if so, to find one.

Remark 6.3.3 (HomExt is subword extender for Hom)**.** A subword extender for the code $\mathrm{Hom}(G, H)$ is an algorithm that solves HomExt(G, H).

Remark 6.3.4 (Subword Extender for Hom versus aHom)**.** A subword extender for the code $\mathrm{aHom}(G, H)$ can also be found using an algorithm that solves HomExt(G, H). Implementing this is addressed in then next section. However, since economical list-decoding for Hom and aHom are equivalent (Section 4.3.3), a subword extender only for Hom suffices for our list-decoding purposes.

We elaborate on Observation 3.1.15, that a certificate-list-decoder and a subword extender combine to a list-decoder. Suppose that Γ is the output list of a certificate-list-decoder. Then, calling a subword extender on each partial map in Γ will give the output of a list-decoder. (If the subword extender fails to extend, then that partial map can be deleted from the list.)

Observation 6.3.5. *Let G and H be groups. Given a certificate-list Γ for the code $\mathrm{Hom}(G,H)$, $|\Gamma|$ calls to $\textsc{HomExt}(G,H)$ suffices to return a list.*

Since we have found stronger certificate-list-decoders ($\mathcal{W}_\Lambda$-certificate-list-decoders, in Theorem 6.2.11), we define weaker subword extenders ($\mathcal{W}_\Lambda$-subword extenders) for the case of homomorphism codes. Below, let $\lambda > 0$.

Definition 6.3.6. (Homomorphism Extension with Threshold, $\textsc{HomExt}_\lambda(G,H)$)

Instance: A partial map $\gamma : G \rightharpoonup H$ satisfying $\mu(\langle \mathrm{dom}\,\gamma \rangle) > \lambda$.

Solution: A homomorphism $\varphi \in \mathrm{Hom}(G,H)$ that extends γ, i.e., $\varphi|_{\mathrm{dom}\,\gamma} = \gamma$.

The $\textsc{HomExt}_\lambda$ Decision and Search Problems are defined as for $\textsc{HomExt}$. $\textsc{HomExt}_\lambda$ is allowed to answer incorrectly on inputs γ that do not satisfy the promise $\mu(\langle \mathrm{dom}\,\gamma \rangle) > \lambda$.

Remark 6.3.7. A $\mathcal{W}_\Lambda$ subword extender for the code $\mathrm{Hom}(G,H)$ is an algorithm that solves $\textsc{HomExt}_\lambda(G,H)$.

Note that, if $\lambda_1 \leq \lambda_2$, then an oracle for $\textsc{HomExt}_{\lambda_1}(G,H)$ can answer $\textsc{HomExt}_{\lambda_2}(G,H)$ queries as well.

Similarly to Observation 6.3.5, we can combine the output of a $\mathcal{W}_\Lambda$-certificate-list-decoder with calls to $\textsc{HomExt}_\Lambda(G,H)$ to find a list-decoder.

Observation 6.3.8. *Let G and H be groups. Let $\lambda_1 \leq \lambda_2$. Given a $\mathcal{W}_{\lambda_2}$-certificate-list Γ for the code $\mathrm{Hom}(G,H)$, $|\Gamma|$ calls to $\textsc{HomExt}_{\lambda_1}(G,H)$ suffices to return a list.*

Theorem 6.2.11 gives a $\mathcal{W}_\Lambda$-certificate-list-decoder for $\mathrm{Hom}(G,H)$ when G is SRG. In practice we may not be able to determine the value of Λ but instead have a large lower bound $\lambda \leq \Lambda$. In this case, we may apply Observation 6.3.8.

Corollary 6.3.9. *Let G be an SRG group and H be an arbitrary group. If an oracle is given for $\textsc{HomExt}_\lambda$ for some $\lambda \leq \Lambda$, then $\mathrm{Hom}(G,H)$ is AlgEcon.*

We have seen the benefits of a better Λ lower bound (weaker $\textsc{HomExt}_\lambda$ oracle required). Other benefits are discussed in Section 6.3.4.

Homomorphism extension from alternating groups.

Further elaborating on the comment after Remark 3.1.16, we note that this relaxation is critical in our application to the alternating group. While we can solve $\textsc{HomExt}(A_n, S_m)$ for partial maps whose domain generate subgroups of polynomial index, we have no hope of solving it for all partial maps on A_n. We follow the procedure of Observation 6.3.8 for alternating domain.

The following theorem addresses the HomExt Search Problem for the permutation representations of the alternating groups. Proving this theorem (and extensions of it) is the focus of the next chapter.

Theorem 6.3.10. *Let $G = A_n$, $H = S_m$ and $\lambda = 1/\operatorname{poly}(n)$. If $m < 2^{n-1}/\sqrt{n}$, then $\textsc{HomExt}_\lambda(G, H)$ Search can be solved in $\operatorname{poly}(n, m)$ time.*

Remark 6.3.11. In fact, under the assumptions of Theorem 6.3.10, the number of extensions can be counted in $\operatorname{poly}(n, m)$ time.

Corollary 6.3.12 (Theorem 3.2.16 restatement)**.** *If $G = A_n$, $H = S_m$ and $m < 2^{n-1}/\sqrt{n}$, then $\operatorname{Hom}(G, H)$ is AlgEcon.*

6.3.3 Subword extension for $\operatorname{aHom}(G, H)$ using $\textsc{HomExt}(G, H)$

In this section we show how HomExt can be used to find a subword extender for aHom (as opposed to Hom). While this is not necessary to achieve our economical list-decoding goals given the equivalence of economically list-decoding Hom versus aHom (Corollary 4.3.19), this shows that HomExt is the natural question for subword extension in homomorphism codes.

Remark 6.3.13. Let G and H be groups to which we are given black-box access. Then, a subword extender for $\operatorname{aHom}(G, H)$ can be implemented in $\operatorname{poly}(\operatorname{enc}(G))$-time in the unit-cost model for H, assuming we are given an oracle for $\textsc{HomExt}(G, H)$.

Since Theorem 3.2.8 guarantees $\mathcal{W}_\Lambda$-certificate-lists, we need only provide a $\mathcal{W}_\Lambda$-subword extender (see Remark 6.3.14). In this case, the HomExt oracle may be relaxed to account for this restriction on certificates.

The next result is the $\mathcal{W}_\Lambda$-certificate version of Remark 6.3.13.

Remark 6.3.14. Let G and H be groups to which we are given black-box access. Suppose that we are given two oracles, one for $\text{HomExt}_\Lambda(G, H)$ and one which returns the order of a subgroup in G. Then, a $\mathcal{W}_\Lambda$-subword extender for $\text{aHom}(G, H)$ can be implemented in $\text{poly}(\text{enc}(G))$ time in the unit-cost model for H.

In this section, we show that solving HomExt is sufficient for finding an aHom subword extender. This gives a strategy of combining a subword extender and certificate-list-decoder directly for aHom, to achieve a list-decoder. In particular, we prove Remark 6.3.13 (Remark 6.3.14 for $\mathcal{W}$-certificates), which states that subword extenders ($\mathcal{W}$-subword-extenders) can be implemented efficiently given a HomExt (HomExt_Λ) oracle.

Subword extender from HomExt oracle, generic version.

We address Remark 6.3.13.

We would like to implement an efficient subword extender (extending partial maps to affine homomorphisms) using the $\text{HomExt}(G, H)$-oracle (extending partial maps to homomorphisms). For every $G \rightharpoonup H$ partial map γ that we would like to extend, we construct a $G \rightharpoonup H$ partial map τ, feed it into $\text{HomExt}(G, H)$, then take an affine translation of the extension homomorphism (if it exists) as follows.

Construction 6.3.15 (Mapping of partial maps)**.** The algorithm Subword (pseudocode below) returns the output of a subword extender on input γ, a $G \rightharpoonup H$ partial map, for the code $\text{aHom}(G, H)$.

1: **procedure** $\text{Subword}(\gamma : G \rightharpoonup H)$

2: Set $S \leftarrow \text{dom}(\gamma)$ and pick $a \in S$

3: Define $T = a^{-1}S = \{a^{-1}s : s \in S\}$

4: Define $\tau : G \rightharpoonup H$ with domain T and $\tau(a^{-1}s) = \gamma(a)^{-1}\gamma(s)$

5: **if** HomExt$(G, H)(\tau) =$ 'no solution' **then**

6: **return** 'no solution'

7: **else**

8: $\psi \leftarrow$ HomExt$(G, H)(\tau)$ ▶ $\psi \in \mathrm{Hom}(G, H)$ `is the extension of` τ `found`

9: $\varphi \leftarrow (\gamma(a)\psi(a)^{-1})\psi$ ▶ φ `is an affine translate of` ψ `by` $\gamma(a)\psi(a)^{-1} \in H$

10: **return** φ

11: **end if**

12: **end procedure**

The next result shows that the construction above is a subword extender for the code aHom(G, H).

Lemma 6.3.16. *Let $\gamma, \tau : G \rightharpoonup H$ be the partial maps defined as in Construction 6.3.15 above.*

(a) Suppose τ extends to $\psi \in \mathrm{Hom}(G, H)$. Define $\varphi \in \mathrm{aHom}(G, H)$ (as in Construction 6.3.15) by $\varphi = (\gamma(a)\psi(a)^{-1})\psi$, for any $a \in S$. Then, γ extends to φ.

(b) The partial map τ extends to a homomorphism in Hom(G, H) *if and only if the partial map γ extends to an affine homomorphism in* aHom(G, H).

It is clear that if φ is defined, it is an affine homomorphism. Item (a) of Lemma 6.3.16 follows from the next statement.

Claim 6.3.17. *Using the notation of Construction 6.3.15, if φ is defined, then φ extends γ, i.e., $\varphi|_S = \gamma$.*

Proof. Let $s \in S$. Fix a as above. Then, $\varphi(s) = (\gamma(a)^{-1}\psi(a)^{-1})\psi(s)$ by definition. But, since ψ is a homomorphism and by the definition of ψ, we find that $\psi(a)^{-1}\psi(s) = \psi(a^{-1}s) = \tau(a^{-1}s) = \gamma(a)^{-1}\gamma(s)$. So, $\varphi(s) = \gamma(s)$. □

Item (b) of Lemma 6.3.16 follows from showing that if an extension φ of γ exists then HomExt(G, H) will return a homomorphism solution.

Lemma 6.3.18. *Let $\gamma, \tau : G \rightharpoonup H$ be partial maps as in Lemma 6.3.16. Suppose that $\varphi \in$ aHom(G, H) extends γ. Write $\varphi = h\psi$ for $h \in H$ and $\psi \in \mathrm{Hom}(G, H)$. Then, ψ extends τ.*

Proof. Let $a \in \mathrm{dom}(\gamma)$ and $T = a^{-1} \cdot \mathrm{dom}(\gamma)$ be as defined above. It suffices to show that $\psi|_T = \tau$. Let $s \in \mathrm{dom}(\gamma)$. Then, by definition, $\psi(a^{-1}s) = \psi(a)^{-1}\psi(s) = (h\varphi(a))^{-1}(h\varphi(s)) = \varphi(a)^{-1}\varphi(s) = \tau(a^{-1}s)$. □

Subword extender from HomExt oracle, $\mathcal{W}$-certificate version.

Remark 6.3.14 is shown similarly. A $\mathcal{W}_\Lambda$-subword extender can be implemented by replacing, in the algorithm Subword above, the oracle for HomExt(G, H) by an oracle for HomExt$_\lambda(G, H)$, for any $\lambda \leq \Lambda$. The order oracle is called before the HomExt$_\lambda$ oracle to ensure a valid input.

Recall that the oracle for HomExt$_\lambda(G, H)$ is only guaranteed to answer correctly on an input $\tau : G \rightharpoonup H$ when the domain of γ generates a density λ subgroup, i.e., $\mu(\langle \mathrm{dom}\, \tau \rangle) > \lambda$. However the $\mathcal{W}$-subword extender need only extend a partial map $\gamma : G \rightharpoonup H$ when the affine closure of its domain has density $\Lambda_{G,H}$, i.e., $\mu(\langle \mathrm{dom}\, \gamma \rangle_{\mathsf{aff}}) > \Lambda_{G,H}$. Remark 6.3.14 follows from the next statements.

Observation 6.3.19. *Let $S \subset G$. Pick $a \in S$ and define $T = a^{-1}S$. Then, the subcoset $\langle S \rangle_{\mathsf{aff}}$ is a coset of the subgroup $\langle T \rangle$.*

Corollary 6.3.20. *Let $\gamma, \tau : G \rightharpoonup H$ and $S, T \subset G$ be defined as in Lemma 6.3.16. Let $\lambda \leq \Lambda_{G,H}$. If $\mu(\mathrm{dom}\, \gamma) > \Lambda_{G,H}$, then $\mu(\mathrm{dom}\, \tau) > \Lambda_{G,H} \geq \lambda$, so* HomExt$_\lambda(G, H)$ *will give a correct answer on input τ.*

6.3.4 The role of pruning and Λ

We discuss the importance of knowing Λ and in particular the benefits of Λ lower bounds. Our definition of AlgEcon does not require that Λ be part of the input.

Pruning using exact knowledge of Λ.

Knowing Λ lets us prune the output list of a list-decoder. In particular, for a CombEcon code, this guarantees a $\text{poly}(1/\varepsilon)$ bound on the size of the output list.

Remark 6.3.21 (Pruning the output list)**.** The definition of list-decoder requires only that the output list $\widetilde{\mathcal{L}}$ be a superlist of the desired list, i.e., $\widetilde{\mathcal{L}} \supseteq \mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon)$. If Λ is known then it is possible to "prune" $\widetilde{\mathcal{L}}$. In other words, we can guarantee that with high probability $\widetilde{\mathcal{L}}$ contains only homomorphisms in $\mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon/2)$, i.e., $\widetilde{\mathcal{L}} \subseteq \mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon/2)$. This can be accomplished by estimating $\text{agr}(f,\varphi)$ through sampling[3] for every $\varphi \in \widetilde{\mathcal{L}}$.

The definition of AlgEcon requires that the output list for the code $\text{aHom}(G,H)$ be of length $\text{poly}(1/\varepsilon)$, but allows $\text{poly}(1/\varepsilon, \log|G|)$ queries and $\text{poly}(1/\varepsilon, \log|G|, \log|H|)$ computation time.[4] A CombEcon code $\text{aHom}(G,H)$ has a list-size bound of $\ell(\text{aHom}(G,H), \Lambda+\varepsilon) = \text{poly}(1/\varepsilon)$. The output of a list-decoder satisfying the AlgEcon query and time complexities may not necessary satisfy the list-size requirement. But, through pruning of the output list, knowing Λ allows us to guarantee the desired $\text{poly}(1/\varepsilon)$ bound on the output list size, since $|\widetilde{\mathcal{L}}| \le |\mathcal{L}(\text{aHom}(G,H), f, \Lambda+\varepsilon/2)| = \text{poly}(2/\varepsilon)$.

In fact, in the algorithm of [DGKS08, GS14], knowing Λ is necessary. The time complexity analysis of the AlgEcon claim depends on the list-size bound at every iteration after pruning.

3. In $\text{poly}(\delta, \log|G|)$ time, we can estimate $\text{agr}(f,\varphi)$ for φ in the output list to within δ with high confidence. With this, we can prune (remove) all homomorphisms with small agreement $\text{agr}(f,\varphi) \le \Lambda + \epsilon/2$.

4. This is assuming an efficient encoding of G and H, otherwise this is a function of the encoding length.

Pruning using lower bounds on Λ.

Even without exact knowledge of Λ, lower bounds still have pruning benefits. We discuss the benefits in the approach of the previous section.

- Final output list: Again, the output list $\widetilde{\mathcal{L}}$ can be pruned using any lower bound $\lambda \leq \Lambda$. This guarantees that the output list contains only affine homomorphisms with agreement greater than $(\lambda+\varepsilon/2)$, i.e., $\widetilde{\mathcal{L}} \subseteq \mathcal{L}(\mathrm{aHom}(G,H), f, \lambda+\varepsilon/2)$. However, there is no polynomial output-list-size guarantee. AlgEcon list-size bounds must be found through other means.

- Faster processing of certificate-lists (output of certificate-list-decoder) into output lists (output of list-decoder): Regardless of the value of Λ and whether it is known, the certificate-list-decoder of Theorem 6.2.11 is guaranteed to return certificates in $\mathcal{W}_{\Lambda_{G,H}}$. A known lower bound λ for $\Lambda_{G,H}$ allows pruning of partial maps γ that do not satisfy $\mu(\langle \mathrm{dom}\,\gamma \rangle_{\mathsf{aff}}) > \lambda$. The subword extender need not be called on these maps.

Additionally, a better lower bound $\lambda \leq \Lambda$ allows us to call a weaker $\textsc{HomExt}_\lambda$ oracle, as discussed in the previous section.

We remark on a phenomenon in our new approach. The certificate-list-decoder presented in Section 6.2.5 is valid regardless of the value of Λ. Even without any bounds on Λ, if we are handed a HomExt oracle then we immediately have a list-decoder. However, this ability relies on an extremely strong $\textsc{HomExt}_\lambda$ oracle that corresponds to $\lambda = 0$, which seems an unreasonable assumption.

6.3.5 Homomorphism Extension 012 and Λ lower bounds

We have seen that stronger Λ lower bounds improve the efficiency of list-decoders through pruning and allow use of weaker HomExt oracles. In this section, we will see that a stronger HomExt oracle may, conversely, help improve Λ lower bounds.

We define the HomExt012 Problem. It asks to distinguish between the cases of no extension, unique extension, and multiple extensions. In the case of unique extension, it asks for the extension.

Definition 6.3.22. (HomExt012(G, H))

Instance: A partial map $\gamma : G \rightharpoonup H$.

Solutions: The set defined by

$$\mathrm{HExt}(\gamma) := \{\varphi \in \mathrm{Hom}(G, H) : \varphi|_{\mathrm{dom}\,\gamma} = \gamma\}.$$

Output:
$$\begin{cases} \text{'none'} & \text{if } |\mathrm{HExt}(\gamma)| = 0 \\ \varphi \in \mathrm{HExt}(\gamma) & \text{if } |\mathrm{HExt}(\gamma)| = 1 \cdot \\ \text{'multiple'} & \text{if } |\mathrm{HExt}(\gamma)| \geq 2 \end{cases}$$

The HomExt012$_\lambda(G, H)$ problem is defined similarly, but requiring only correct answers on the $G \rightharpoonup H$ partial maps γ that satisfy $\mu(\mathrm{dom}\,\gamma) > \lambda$.

Of course, the HomExt Search Problem is weaker than HomExt012.

Proposition 6.3.23. *Let G and H be groups to which we are given black-box access. Suppose that we are given two oracles, one for* HomExt012(G, H) *and one which returns the order of a subgroup of G. Then, for any $G \rightharpoonup H$ partial map τ on which* HomExt012(G, H) *returns 'multiple,' the value of $\mu(\langle\tau\rangle)$ is a lower bound for Λ.*

Proof. If the $G \rightharpoonup H$ partial map τ extends to two homomorphisms $\varphi, \psi \in \mathrm{Hom}(G, H)$, then $\mathrm{agr}(\varphi, \psi) \geq \mu(\langle \mathrm{dom}\,\tau\rangle_{\mathsf{aff}})$. So, $\mu(\langle \mathrm{dom}\,\tau\rangle_{\mathsf{aff}})$ is a lower bound for $\Lambda_{G,H}$, and the order oracle can be used to calculate its value. □

The next chapter solves a case of HomExt012$_\lambda$, a stronger version of HomExt012. The case corresponds to the conditions of Corollary 6.3.12: HomExt012$_\lambda(A_n, S_m)$ with $\lambda = 1/\binom{n}{2}$ and $m < 2^{n-1}/\sqrt{n}$.

6.3.6 Finding Λ for prior algorithms

In this section we present an algorithm to determine the value of $\Lambda_{G,H}$ under certain access assumptions.

Theorem 6.3.24. *Let G be a solvable group represented by a polycyclic presentation. Let H be a nilpotent black-box group along with one of the following types of information,*

(a) an oracle that returns the order of any element of H, or

(b) a multiple of the order $|H|$, or

(c) a superset of the prime divisors of the order $|H|$, or

(d) the exact set of prime divisors of the order $|H|$.

Then, there exists a polynomial-time deterministic algorithm that finds the value of $\Lambda_{G,H}$.

We remark that the additional assumptions on access to $|H|$ are ordered in increasing strength.

That the black-box group H is nilpotent does not need to be taken as a promise. Determining nilpotence and solvability of a black-box group is known to be decidable in randomized polynomial time. For more details, see [BS84].

Description of algorithm.

The procedure FindLambda (Algorithm 2 presented below) satisfies Theorem 6.3.24.

For convenience, we restate Proposition 2.2.8, which characterizes $\Lambda_{G,H}$ for solvable G or nilpotent H.

Theorem 6.3.25. *Let G and H be groups. If G is solvable or if H is nilpotent, then $\Lambda_{G,H} = 1/p$, where p is the smallest prime p dividing $|G|$ that satisfies the following.*

(1) G contains a normal subgroup of index p, and

(2) p divides the order of H.

If no p dividing $|G|$ satisfies both conditions above, then $\mathrm{Hom}(G, H)$ *contains only the identity, and* $\Lambda_{G,H} = 0$.

Intuitively, FindLambda checks Theorem 6.3.25, by searching through the primes p_i from the polycyclic presentation of G in increasing order, to find the smallest p_i satisfying both Condition (1) and (2).

First, FindLambda undergoes a preprocessing stage. It considers the abelian presentation $\langle X|\mathcal{R}^{\mathsf{ab}}\rangle_{\mathsf{ab}}$ of G/G' found by abelianizing the relations of the polycyclic presentation for G. The algorithm $\mathcal{A}_{\mathrm{KB}}$ from Kannan and Bachem in Theorem 2.1.13 is used to find the Smith normal form of $A(\mathcal{R}^{\mathsf{ab}})$. From this, M is assigned to be the largest elementary divisor of $A(\mathcal{R}^{\mathsf{ab}})$, or, the order of the largest cyclic subgroup of G/G'.

Then, FindLambda searches the primes p_i in increasing order. The prime p_i satisfies Condition (1) exactly when p_i divides M (see "Testing G" below). The prime p_i satisfies Condition (2) exactly when p_i divides $|h|$ for some generator h of H (see "Testing H" below). Recall that a set of generators is available from the black-box representation of H.

Denote by $\mathcal{A}_{\mathrm{KB}}$ the deterministic algorithm provided by Kannan and Bachem in Theorem 2.1.13 that takes as input an integer matrix B and outputs the elementary divisors $a_1, \ldots, a_\ell$ of B.

Algorithm 2 FindLambda

1: **procedure** FINDLAMBDA(G, H)
2: Initialize $X = \{g_1, \ldots, g_k\}$, $\mathcal{R}$, and $p_1 \geq \cdots \geq p_k$ from G
3: $B \leftarrow B(\mathcal{R}^{\mathsf{ab}})$, the $|\mathcal{R}| \times k$ matrix from Theorem 2.1.12
4: $a_1, \ldots, a_\ell \leftarrow \mathcal{A}_{\mathrm{KB}}(B)$ ▶ Find elementary divisors of B
5: $M \leftarrow a_\ell$ ▶ M is the largest elementary divisor
6: Sort $\{p_i\}$ so that $p_1 \leq \cdots \leq p_k$
7: **for** $i = 1 \ldots k$ **do**
8: **if** p_i divides M and TESTH(p_i) = TRUE **then**
9: **return** $\Lambda_{G,H} = 1/p_i$ ▶ Found smallest p_i satisfying Cond. (1) and (2).
10: **end if**
11: **end for**
12: **return** $\Lambda_{G,H} = 0$ ▶ No such p_i exists
13: **end procedure**

Testing G.

Lemma 6.3.26. *The prime p_i divides M in Line 8 of* FINDLAMBDA *if and only if G contains a normal subgroup of index p_i.*

Proof. Consider the preprocessing steps for G, carried out in Lines 2–5.

The number M computed in Line 5 is the largest elementary divisor of B. By Corollary 2.1.12 and Fact 2.1.4, we find that G/G' is the direct product of cyclic groups with orders given by the elementary divisors of B. Since every elementary divisor of B divides M, every prime divisor of $|G/G'|$ divides M.

We found that a prime p divides $|G/G'|$ exactly when p divides M. Lemma 6.3.26 follows from Lemma 6.3.27 below. □

Lemma 6.3.27. *We claim that a group G contains contains a normal subgroup of prime index p if and only if p divides the order of G/G', the abelianization of G.*

Proof. Let $N \lhd G$ have prime index $p = |G/N|$. Then, $G/N \cong \mathbb{Z}/p\mathbb{Z}$ is abelian, so $G' \leq N$.

If p divides $|G/G'|$, then there exists $K \lhd G/G'$ such that $|G/G' : K| = p$. Let $\pi : G \to G/G'$ be the natural projection map. Then, $\pi^{-1}(K)$ is a normal subgroup of G of index p. □

Testing H.

Lemma 6.3.28. *Let p be a prime. Let H be a nilpotent black-box group of one of the types described in Theorem 6.3.24. The algorithm* TestH *(Algorithm 3 defined below) decides whether p divides $|H|$ in polynomial time.*

If H is provided access type (d), then TestH(p, H) simply checks whether p is in the provided list of primes.

If H is provided access type (a), (b) or (c), then TestH satisfies Lemma 6.3.28. We will prove the correctness of TestH below.

Algorithm 3 TestH

```
1:  procedure TestH(p, H)
2:      Initialize S, the set of generators from black-box representation of H
3:      for h ∈ S do
4:          if p divides |h| then
5:              return TRUE
6:              Exit TestH
7:          end if
8:      end for
9:      return false
10: end procedure
```

Proof of Lemma 6.3.28, TESTH *works.* We show that TESTH returns TRUE exactly when p divides $|H|$. If $\text{TESTH}(p, H) = \text{TRUE}$, then p divides the $|h|$ for some $h \in S \subset H$, so p divides $|H|$. The other direction follows from Lemma 6.3.29 below.

Line 4 is possible in polynomial time. This is obvious for access type (a). For access type (b), denote by m the given multiple of the order of $|H|$. Divide m by p until it is no longer divisible by p, which may be accomplished in poly$(\log m)$-time. Call this resulting number $\tilde{m}$. Calculate $h^{\tilde{m}}$, which may be accomplished in $O(\log \tilde{m})$ queries to the mult oracle of H. If $h^{\tilde{m}} = 1$ then p does not divide the order of h. Otherwise, p does divide the order of h. Thus, Line 4 can be carried out in polynomial time for access type (a). For access type (c), we show that a multiple of the order $|H|$ (access type (b)) can be calculated in polynomial time, provided a superset P of the prime divisors of $|H|$. Let $k = \lceil \log_2(|H|) \rceil$. Then, $\prod_{p \in P} p^k$ is a multiple of $|H|$.

The number of iterations of the **for** loop is bounded by $|S|$, the size of the generating set of the black-box representation. The algorithm TESTH runs in time bounded by $|S|\,\text{poly}(m)$. □

Lemma 6.3.29. *Let p be a prime. Let H be a nilpotent group. If p divides $|H|$, then any set of generators of H will contain an element with order divisible by p.*

Proof. Let $S \subset H$ be a set of generators for H.

Let $p_1, \ldots, p_\ell$ be the prime divisors of $|H|$. Since H is nilpotent, we can write $H = K_1 \times \cdots \times K_\ell$ so that K_i is a nontrivial p_i-group for $i = 1 \ldots \ell$. Each $h \in H$ can be written as $(k_1, \ldots, k_\ell)$, for $k_i \in K_i$ and $|h| = |k_1| \cdot \ldots \cdot |k_\ell|$.

Since p divides $|H|$, $p = p_i$ for some i. There must exist an element $s = (k_1, \ldots, k_\ell) \in S$ such that $k_i \neq 1$. (Otherwise S cannot generate all of H.) It follows that p divides $|s|$. □

Comparison of access models with [DGKS08].

In [DGKS08], list-decoding bounds are proved for pairs of abelian groups, assuming the

prime-power decomposition of both groups. Converting from a general presentation to a prime-power decomposition requires prime factorization. However, their local list-decoder for abelian groups G and H requires only knowledge of $\Lambda_{G,H}$ and black-box access to H.

As we have shown in Theorem 6.3.24, black-box access to H along with additional information, such as a multiple of $|H|$, is sufficient for finding $\Lambda_{G,H}$. We don't need prime factorization of $|H|$. So, the result of [DGKS08] can be converted to the less restricted model, by composing with our algorithm for finding $\Lambda_{G,H}$.

CHAPTER 7

HOMOMORPHISM EXTENSION

7.1 Introduction

HOMOMORPHISM EXTENSION asks whether a group homomorphism from a subgroup can be extended to a homomorphism from the entire group. We consider the case that the groups are represented as permutation groups. The complexity of this natural problem within NP is unresolved.

7.1.1 Structure of chapter

This chapter defines and proves our results for Homomorphism Extension. The remainder of this section presents definitions, results, and methods. Our results pertain to HOMEXTSYM, the version of Homomorphism Extension that considers group actions (symmetric codomain).

Section 7.2 sets notation local to this chapter.

Section 7.3 defines Multi-Dimensional Subset Sum with Repetition (MULTISSR) and other versions, then states relevant results for these problems.

Section 7.4 reduces HOMEXTSYM to an oracle version of MULTISSR efficiently. This section contains the meat of this chapter.

Section 7.5 reduces the case of HOMEXTSYM considered in our main theorem to a triangular version of MULTISSR, which can be solved efficiently.

Section 7.6 addresses how to generate extensions within one equivalence class (for our enumeration results).

Section 7.7 calls upon results by Lenstra and Kannan [LJ83, Kan87] for Integer Linear Programming, which solves HOMEXTSYM for large permutation domain.

7.1.2 Definition and results

We define the HOMOMORPHISM EXTENSION problem. Denote by $\mathrm{Hom}(G, H)$ the set of homomorphisms from group G to group H.

Definition 7.1.1. HOMOMORPHISM EXTENSION

Instance: Groups G and H and a partial map $\gamma : G \rightharpoonup H$.

Solution: A homomorphism $\varphi \in \mathrm{Hom}(G, H)$ that extends γ, i.e., $\varphi|_M = \gamma$.

The HOMOMORPHISM EXTENSION Decision Problem (HOMEXT) asks whether a solution exists.

Remark 7.1.2. Our algorithmic results for HOMEXT solve the HOMOMORPHISM EXTENSION Search Problem as well, which asks whether a solution exists and, if so, to find one.

The problems as stated above are not fully specified. Representation choices of the groups G and H affect the complexity of the problem. For example, G may be given as a permutation group, a black-box group, or a group given by a generator-relator presentation.

For the rest of this chapter we restrict the problem to permutation groups.

Definition 7.1.3. HOMEXTPERM is the version of HOMEXT where the groups are permutation groups *given by a list of generators.* HOMEXTSYM is the subcase of HOMEXTPERM where the codomain H is a symmetric group.

Membership in permutation groups is polynomial-time testable. Our standard reference for permutation group algorithms is [Ser03]. Section 2.4 summarizes the results we need, including material not easily found in the literature. Our standard reference for permutation group theory is [DM96].

Partial maps are represented by listing their domain and values on the domain. Homomorphisms in $\mathrm{Hom}(G, H)$ are represented by their values on a set of generators of G.

For a partial map $\gamma : G \rightharpoonup H$, we denote by $M_\gamma := \langle \mathrm{dom}\, \gamma \rangle$ the subgroup of G generated by the domaim $\mathrm{dom}\, \gamma$ of γ.

Remark 7.1.4. Whether the input map $\gamma : G \rightharpoonup H$ extends to a homomorphism from the subgroup generated by its domain ($\exists \psi \in \mathrm{Hom}(M_\gamma, H)$ s.t. $\psi|_{\mathrm{dom}\,\gamma} = \gamma$) is a polynomial-time testable condition in permutation groups.

Since extending to $M_\gamma \leq G$ is easy, we are primarily concerned with extending a homomorphism from a subgroup to a homomorphism from the whole group.

Assumption 7.1.5 (Given partial map defines a homomorphism on subgroup)**.** Unless otherwise stated, in our analysis we assume without loss of generality that the input partial map $\gamma : G \rightharpoonup H$ extends to a homomorphism in $\mathrm{Hom}(M_\gamma, H)$. This is possible due to Remark 7.1.4. In this case, the homomorphism ψ is represented by γ, as a partial map on generators of M_γ. We will think of ψ as the input to HomExt. We often drop the subscript on M_γ.

Since a minimal set of generators of a permutation group of degree n has no more than $2n$ elements [Bab86] and any set of generators can be reduced to a minimal set in polynomial time, we shall assume our permutation groups are always given by at most $2n$ generators.

We note that the decision problem HomExtPerm is in NP.

Open Problem 7.1.6. *Is* HomExtPerm *NP-complete?*

We consider the important subcase of the problem when $H = S_m$, the symmetric group of degree m. A homomorphism $G \to S_m$ is called a **group action** (more specifically, a G**-action**) on the set $[m] = \{1, \ldots, m\}$.

The HomExtSym problem seems nontrivial even for bounded G (and variable m).

Theorem 7.1.7 (Bounded domain)**.** *If G has bounded order, then* HomExtSym *can be solved in polynomial time.*

The degree of the polynomial in the polynomial running time is exponential in $\log^2|G|$.

Open Problem 7.1.8. *Can* HomExtSym *be replaced by* HomExtPerm *in Theorem 7.1.7, i.e., can $H = S_m$ be replaced by $H \leq S_m$?*

Our main result, the one used in our work on homomorphism codes, concerns variable n and is stated next.

In the results below, "polynomial time" refers to poly(n, m) time.

Theorem 7.1.9 (Main)**.** *If $G = A_n$ (alternating group of degree n),* HomExtSym *can be solved in polynomial time under the following assumptions.*

(i) The index of M in A_n is bounded by poly(n)*, and*

(ii) $m < 2^{n-1}/\sqrt{n}$, where $H = S_m$.

Under the assumptions above, counting the number of extensions is also polynomial-time.

Theorem 7.1.10 (Main, counting)**.** *Under the assumption of Theorem 7.1.9, the number of solutions to the instance of* HomExtSym *can be found in polynomial time.*

Note the rather generous upper bound on m in item (ii). Whether an instance of HomExtSym satisfies the conditions of Theorem 7.1.9 can be verified in poly(n) time (see Section 2.4.3).

We state a polynomial-time result for very large m (Theorem 7.1.11, of which Theorem 7.1.7 is a special case).

Theorem 7.1.11 (Large range)**.** *If $G \leq S_n$ and $m > 2^{1.7^{n^2}}$, then* HomExtSym *can be solved in polynomial time.*

7.1.3 Methods

We prove the results stated above by reducing HomExtSym to a polynomial-time solvable case of a multi-dimensional oracle version of Subset Sum with Repetition (SSR). SSR asks

to represent a target number as a non-negative *integral linear combination* of given numbers, whereas the classical Subset Sum problem asks for a *0-1 combination*. SSR is NP-complete by easy reduction from Subset Sum.

We call the multi-dimensional version of the SSR problem MULTISSR. The reduction from homomorphism extension to MULTISSR is the main technical contribution of this chapter (Theorem 7.1.12 below).

The reduction is polynomial time; the complexity of our solutions to HOMEXTSYM will depend on the complexity of special cases of MULTISSR that arise. The principal case of MULTISSR is one we call "triangular" ; this case can be solved in polynomial time. The difficulty is aggravated by exponentially large input to MULTISSR, to which we assume oracle access (ORMULTISSR Problem). Implementing oracles calls will amount to solving certain problems in computational group theory, addressed in Section 8 of the Appendix.

The MULTISSR problem takes as input a multiset K in universe $\mathcal{U}$ (viewed as a non-negative integral function $\mathsf{K} : \mathcal{U} \to \mathbb{Z}^{\geq 0}$) and a set $\mathfrak{F}$ of multisets in $\mathcal{U}$. MULTISSR asks if K is a nonnegative integral linear combination of multisets in $\mathfrak{F}$ (see Section 7.4.2). The set $\mathfrak{F}$ will be too large to be explicitly given (it will contain one member per conjugacy class of subgroups of G). Instead, we contend with oracle access to the set $\mathfrak{F}$. For a more formal presentation of MULTISSR and ORMULTISSR, see Section 7.3.

From every instance ψ of HOMEXTSYM describing a group action, we will construct an ORMULTISSR instance OMS_ψ (see Section 7.4.2). In the next result, we describe the merits of this translation.

Two permutation actions $\varphi_1, \varphi_2 : G \to S_m$ are **permutation equivalent** if there exists $h \in S_m$ such that $\varphi_1(g) = h^{-1}\varphi_2(g)h$ for all $g \in G$.

Theorem 7.1.12 (Translation)**.** *For every instance $\psi \in \mathrm{Hom}(M, S_m)$, the instance OMS_ψ of* ORMULTISSR *satisfies the following.*

(a) OMS_ψ can be efficiently computed from ψ. For what this means, see Section 7.4.2.

(b) There exists a bijection between the set of non-empty classes of equivalent (under permutation equivalence) extensions $\widetilde{\varphi} : G \to S_m$ and the set of solutions to OMS_ψ.

(c) Given a solution to OMS_ψ*, a representative $\widetilde{\varphi}$ of the equivalence class of extensions can be computed efficiently.*

Here, "efficiently" means in $\text{poly}(n, m)$-time. The universe $\mathcal{U}$ of OMS_ψ will be the conjugacy classes of subgroups of M. The set $\mathfrak{F}$ will be indexed by the conjugacy classes of subgroups of G. These sets can be exponentially large. For $G = S_n$, $|\mathfrak{F}| = \exp(\widetilde{\Theta}(n^2))$ by [Pyb93].

Now, it suffices to efficiently find solutions to instances OMS_ψ of OrMultiSSR arising under this reduction.

Theorem 7.1.11 (large m) follows from Theorem 7.1.12 and a result of Lenstra [LJ83] (cf. Kannan [Kan87]), that shows Integer Linear Programming is fixed-parameter tractable. As MultiSSR can naturally be formulated as an $|\mathcal{U}| \times |\mathfrak{F}|$ integer linear program, we conclude polynomial-time solvability due to the assumed magnitude of m (see Appendix, Section 7).

To prove Theorem 7.1.9, we show that OMS_ψ instances satisfy the conditions of the problem TriOrMultiSSR, a "triangular" version of OrMultiSSR (see Section 7.5).

Theorem 7.1.13 (Reduction to TriOrMultiSSR)**.** *If an instance ψ of* HomExtSym *satisfies the conditions of Theorem 7.1.9, the instance* OMS_ψ *of* OrMultiSSR *is also an instance of* TriOrMultiSSR*. The oracle queries can be answered in polynomial time.*

Despite only being given oracle access, TriOrMultiSSR turns out to be polynomial-time solvable (see Section 7.3.2, or the Appendix, Section 5).

Proposition 7.1.14. TriOrMultiSSR *can be solved in polynomial time.*

Proposition 7.1.15. *If a solution to* TriOrMultiSSR *exists, then it is unique.*

Polynomial time for an OrMultiSSR problem means polynomial in the length of K and the length of the representation of elements of $\mathfrak{F}$. For details on representating multisets, see Section 7.2.1.

7.1.4 Efficient enumeration

The methods discussed give a more general result than claimed. Instead of solving the Search Problem, we can in fact efficiently solve the Threshold-k Enumeration Problem for HomExtSym. This problem asks to find the set of extensions, unless there are more than k, in which case output k of them.

This question is also motivated by the list-decoding problem; specifically, Threshold-2 Enumeration can be used to prune the output list. See Sections 6.3.4 and 6.3.5 for details. We remark that solving Threshold-2 Enumeration already requires all relevant ideas in solving Threshold-k Enumeration.

Definition 7.1.16 (Threshold-k)**.** For a set $\mathcal{S}$ and an integer $k \geq 0$, the **Threshold-k Enumeration Problem** asks to return the following pair $(\texttt{val}, \texttt{out})$ of outputs.

If $|\mathcal{S}| \leq k$, return $\texttt{val} = |\mathcal{S}|$ and $\texttt{out} = \mathcal{S}$

Else, return $\texttt{val}$ = "more" and $\texttt{out}$ = a list of k distinct elements of $\mathcal{S}$.

Note that the threshold-0 enumeration problem is simply the **decision problem** "is $\mathcal{S}$ non-empty?" while the threshold-1 enumeration problem includes the **search problem** (if not empty, find an element of $\mathcal{S}$).

We say that an algorithm **efficiently** solves the threshold-k Enumeration Problem if the cost divided by k is considered "modest" (in our case, polynomial in the input length).

Our work on list-decoding homomorphism codes uses solutions to the *threshold-2 enumeration problem* for the set of extensions of a given homomorphism. With potential future applications in mind, we discuss the threshold-k enumeration problem for variable k.

Definition 7.1.17. Homomorphism Extension Threshold-k Enumeration (HomExtThreshold) is the Threshold-k Enumeration Problem for the set of solutions to Homomorphism Extension (HExt_G defined below).

Notation 7.1.18 ($\mathrm{HExt}_G(\varphi)$)**.** We will denote by $\mathrm{HExt}_G(\varphi)$ the set of solutions to an instance φ of HomExt.

$$\mathrm{HExt}_G(\varphi) := \{\widetilde{\varphi} \in \mathrm{Hom}(G, H) : \widetilde{\varphi}|_M = \varphi\}.$$

The following condition strengthens the notion of efficient solutions to threshold enumeration.

Definition 7.1.19 (Efficient enumeration)**.** We say that a set $\mathcal{S}$ can be **efficiently enumerated** if an algorithm lists the elements of $\mathcal{S}$ at modest marginal cost.

The marginal cost of the i-th element is the time spent between producing the $(i-1)$-st and the i-th elements. In this chapter, "modest marginal cost" will mean $\mathrm{poly}(n, m)$ marginal cost, where n and m denote the degrees of the permutation groups G and H, respectively.

Observation 7.1.20. *If a set $\mathcal{S}$ can be efficiently enumerated then the threshold enumeration problem can be solved efficiently.*

In particular, the decision and search problems can be solved efficiently. The following theorems are the strengthened versions of the ones stated above.

Theorem 7.1.21 (Bounded domain, enumeration)**.** *If G has bounded order, then the set $\mathrm{HExt}_G(\varphi)$ can be efficiently enumerated.*

Theorem 7.1.22 (Main, enumeration)**.** *If $G = A_n$ (alternating group of degree n), then the set $\mathrm{HExt}_G(\varphi)$ can be efficiently enumerated under the following assumptions:*

(i) the index of M in A_n is bounded by $\mathrm{poly}(n)$, and

(ii) $m < 2^{n-1}/\sqrt{n}$, *where* $H = S_m$.

Theorem 7.1.23 (Large range, enumeration)**.** *If* $G \leq S_n$ *and* $m > 2^{1.7^{n^2}}$, *then the* HomExtSym *Threshold-*k *Enumeration Problem can be solved in* $\text{poly}(n, m, k)$*) time.*

7.1.5 Enumeration methods

Theorem 7.1.12 gives a bijection between classes of equivalent extensions and solutions to the OrMultiSSR instance. It remains to solve the Threshold-k Enumeration Problem for OrMultiSSR, then to efficiently enumerate extensions within one equivalence class, given a representative of that class.

Solutions of Threshold-k for OrMultiSSR

For Theorem 7.1.9, we stated that the OMS_φ are instances of TriOrMultiSSR. Since solutions are unique if they exist (Proposition 7.1.15), solving the Search Problem also solves the Threshold-k Enumeration Problem for TriOrMultiSSR. But, the Search Problem can be solved in polynomial time by Proposition 7.1.14.

In the case of Theorem 7.1.7, OMS_φ is an integer linear program with a bounded number of variables and constraints (corresponding to classes of subgroups of G) and the solutions can therefore be efficiently enumerated.

For Theorem 7.1.23 (thus also implying Theorem 7.1.21), the Threshold-k Enumeration Problem for the Integer Linear Program version of OMS_φ can be answered in polynomial time by viewing it as an integer linear program. See Section 7.7.

Efficient enumeration within one equivalence class

We now wish to efficiently enumerate extensions within each class of equivalent extensions, given a representative.

Two permutation actions $\varphi_1, \varphi_2 : G \to S_m$ are **equivalent (permutation) actions** if there exists $\lambda \in S_m$ such that $\varphi_1(g) = \lambda^{-1}\varphi_2(g)\lambda$ for all $g \in G$. We say that two

homomorphisms $\widetilde{\varphi}_1, \widetilde{\varphi}_2 : G \to S_m$ are **equivalent extensions** of the homomorphism $\varphi : M \to S_m$ if they (1) both extend φ and (2) are equivalent permutation actions.

Enumerating extensions within one equivalence class reduces to the following: Given subgroups $K \leq L \leq S_m$, efficiently enumerate coset representatives for K in L.

This problem was solved by Blaha and Luks in the 1980s (unpublished, cf. [BL94]). For completeness we include the solution based on communication by Gene Luks [Luk] (see Section 2.4.5).

We explain the connection between finding coset representatives and the classes of equivalent extensions of φ. Consider an extension $\widetilde{\varphi}_0 \in \mathrm{Hom}(G, S_m)$ of $\varphi \in \mathrm{Hom}(M, S_m)$. For any $\lambda \in S_m$, the homomorphism $\widetilde{\varphi}_\lambda$, defined as $\widetilde{\varphi}_\lambda(g) = \lambda^{-1}\widetilde{\varphi}(g)_0\lambda$ for all $g \in G$, is an equivalent permutation action. First, $\widetilde{\varphi}_\lambda = \widetilde{\varphi}$ if and only if $\lambda \in C_{S_m}(\varphi(G))$ (the centralizer in S_m of the φ-image of G, i.e., the set of elemenets of S_m that commute with all elements in $\varphi(G)$). The centralizer of a group in the symmetric group can be found in polynomial time (see Section 2.4.4). Also, $\widetilde{\varphi}|_\lambda$ extends φ (thus is an equivalent extension to $\widetilde{\varphi}$) if and only if $\lambda \in C_{S_m}(\varphi(M))$.

So, finding coset representatives of $K = C_{S_m}(\varphi(G))$ in $L = C_{S_m}(\varphi(M))$ suffices for finding all equivalent extensions. Applying the Blaha–Luks result yields the following corollary (see Section 7.6).

Corollary 7.1.24. *Let $M \leq G \leq S_n$ and $\varphi : M \to S_m$. Suppose that $\widetilde{\varphi}_0 : G \to S_m$ extends φ. Then, the class of extensions equivalent to $\widetilde{\varphi}_0$ can be efficiently enumerated.*

7.2 Notation

We fix notation local to this chapter.

7.2.1 Multisets

In this Homomorphism Extension chapter, we will consider both sets and multisets. All sets and multisets are finite.

We typographically distinguish multisets using "mathsf" font, e.g., F, K and L denote multisets. A multiset within a universe $\mathcal{U}$ is formally a function $\mathsf{L} : \mathcal{U} \to \mathbb{N}$. For a member $u \in \mathcal{U}$ of the universe, the **multiplicity** of u in L is $\mathsf{L}(u)$. We say that u is an **element** of L ($u \in \mathsf{L}$) if $\mathsf{L}(u) > 0$, i.e., if u has non-zero multiplicity in L. The set of elements of L is called the **support of L,** $\mathrm{supp}(\mathsf{L}) \subseteq \mathcal{U}$. We algorithmically represent a multiset $\mathsf{L} : \mathcal{U} \to \mathbb{N}$ by listing its support $\mathrm{supp}(\mathsf{L}) \subseteq \mathcal{U}$ and the values on the support, so the description is of length $|\mathrm{supp}(\mathsf{L})| \cdot \log(\|\mathsf{L}\|_\infty) \cdot \ell$, where ℓ is the description length for elements of L. The **size** of L is $\|\mathsf{L}\|_1$, the 1-norm of the function $\mathsf{L} : \mathcal{U} \to \mathbb{N}$.

Let $\mathsf{L}_1, \mathsf{L}_2 : \mathcal{U} \to \mathbb{N}$ be two multisets in the same universe. Their **sum** $\mathsf{L}_1 + \mathsf{L}_2$ is the multiset obtained by adding the multiplicities. We say that L_1 is a **submultiset** of L_2 if $\mathsf{L}_1(u) \leq \mathsf{L}_2(u)$ for all u.

Sets will continue to be denoted by standard font and defined via one set of braces $\{\}$. Often it is convenient to list the elements of a multiset L as $\{\{L_1, \ldots, L_r\}\} = \{\{L_i : i = 1 \ldots r\}\}$ using double braces, where $L_i \in \mathcal{U}$ and each $u \in \mathcal{U}$ occurs $\mathsf{L}(u)$ times in this list. The length r of this list is the size of L. In our notation, $\{A, A\} = \{A\}$ but $\{\{A, A\}\} \neq \{\{A\}\}$.

A disjoint union of two sets is denoted by $\Omega = \Omega_1 \dot\cup \Omega_2$.

7.2.2 Group theory

Let G be a group. We write $M \leq G$ to express that M is a subgroup; we write $N \trianglelefteq G$ to denote that N is a normal subgroup.

For $M \leq G$ and $a \in G$, we call the coset Ma of M a **subcoset** of G. We define the **index** of a subcoset Ma in G by $|G : Ma| := |G : M|$. For a subset S of a group G, we denote by $\langle S \rangle$ the subgroup generated by S.

Notation 7.2.1 (Sub(G))**.** We denote the set of subgroups of G by $\mathrm{Sub}(G) := \{L : L \leq G\}$.

Notation 7.2.2 (Cosets $L\backslash G$)**.** For $L \leq G$, denote by $L\backslash G := \{Lg : g \in G\}$ the (right) **coset space** (set of right cosets). For $L, M \leq G$, denote by $L\backslash G/M := \{LgM : g \in G\}$ the set of **double cosets**. Double cosets form an uneven partition of G. They are important in defining the MultiSSR instance from an instance of HomExtSym (see Section 7.4).

Two subgroups $L_1, L_2 \leq G$ are **conjugate in** G if there exists $g \in G$ such that $L_1 = g^{-1}L_2g$. The equivalence relation of conjugacy in G is denoted by $L_1 \sim_G L_2$, or $L_1 \sim L_2$ if G is understood.

Notation 7.2.3. For a subgroup $L \leq G$, the **conjugacy class of** L **in** G is denoted by $[L]_G$ (or $[L]$ if G is understood), so $[L]_G := \{L_1 \leq G : L_1 \sim_G L\}$.

Notation 7.2.4 (Conj(G))**.** We denote the set of conjugacy classes of G by $\mathrm{Conj}(G) := \{[L] : L \leq G\}$.

Using the introduced notation, if $L \leq G$, then $L \in \mathrm{Sub}(G)$, $L \in [L] \in \mathrm{Conj}(G)$ and $[L] \subset \mathrm{Sub}(G)$.

7.3 Multi-dimensional subset sum with repetition

We consider the Subset Sum Problem with Repetitions (SSR). An instance is given by a set of positive integers and a "target" positive integer s. The question is "can s be represented as a non-negative linear combination[1] of the other integers?" This problem is NP-complete by an easy reduction from the standard Subset Sum problem, which asks instead for a 0-1 linear combination.

We define a multidimensional version (MultiSSR) below. It is associated to its own threshold-k enumeration problem.

1. Notice that a non-negative linear combination of a set of integers is exactly the sum of a multiset in that set of integers. This question is asking for the existence of a multiset.

Definition 7.3.1. Multi-dimensional Subset Sum with Repetition (MultiSSR)

Instance: Multiset $\mathsf{K} : \mathcal{U} \to \mathbb{N}$ and set $\mathfrak{F}$ of multisets in $\mathcal{U}$.[2]

Solution: A multiset of $\mathfrak{F}$ summing to K, i.e., a multiset $\mathsf{L} : \mathfrak{F} \to \mathbb{N}$ satisfying $\sum_{\mathsf{F} \in \mathfrak{F}} \mathsf{L}(\mathsf{F}) \cdot \mathsf{F} = \mathsf{K}$.

Notation 7.3.2 (SubSum($\mathsf{K}, \mathfrak{F}$))**.** We write SubSum for the set of solutions to an instance of MultiSSR, i.e.,

$$\mathrm{SubSum}(\mathsf{K}, \mathfrak{F}) := \left\{ \mathsf{L} : \mathfrak{F} \to \mathbb{N} \;\middle|\; \sum_{\mathsf{F} \in \mathfrak{F}} \mathsf{L}(\mathsf{F}) \cdot \mathsf{F} = \mathsf{K} \right\}.$$

The MultiSSR Decision Problem asks whether a solution exists (SubSum is nonempty).

The MultiSSR Search Problem asks whether a solution exists and, if so, find one.

The MultiSSR Threshold-k Enumeration Problem asks for the solution to the Threshold-k Enumeration Problem for the set SubSum.

Remark 7.3.3 (MultiSSR as Integer Program)**.** Every instance of MultiSSR can naturally be viewed as an instance of Integer Linear Programming, with $|\mathcal{U}|$ constraints and $|\mathfrak{F}|$ variables. The variables $\mathsf{L}(\mathsf{F})$ are the number of copies of each $\mathsf{F} \in \mathfrak{F}$ in the subset sum. The constraints correspond to checking that every element in $\mathcal{U}$ has the same multiplicities in K and $\sum \mathsf{L}(\mathsf{F}) \cdot \mathsf{F}$.

7.3.1 Oracle MultiSSR

In our application, the set $\mathfrak{F}$ and universe $\mathcal{U}$ will be prohibitively large to input explicitly. To address this, we define an oracle version of the MultiSSR problem called Oracle Multi-dimensional Subset Sum with Repetitions (OrMultiSSR). We will reduce

2. $\mathcal{U}$ is the underlying universe. Its entirety is not required in the input, but its size is the dimensionality of this problem. An element $\mathsf{F} \in \mathfrak{F}$ is a multiset $\mathsf{F} : \mathcal{U} \to \mathbb{N}$ in $\mathcal{U}$.

a HomExtSym instance φ to an OrMultiSSR instance OMS_φ, then show that the oracles can be answered efficiently.

We will find it convenient to introduce a bijection between $\mathfrak{F}$ and another set $\mathcal{V}$ of simpler objects, used to index $\mathfrak{F}$.[3] Access to $\mathfrak{F}$ is given by the oracle "$\mathfrak{F}$-oracle," which on input $v \in \mathcal{V}$ returns the element F_v of $\mathfrak{F}$ indexed by v. Elements of the universes $\mathcal{U}$ and $\mathcal{V}$ are encoded by strings in $\Sigma_1^{n_2}$ and $\Sigma_2^{n_2}$, respectively, and the alphabets Σ_i and encoding lengths n_i constitute the input.

We allow non-unique[4] encodings of $\mathcal{U}$ and $\mathcal{V}$, but provide "equality" oracles.[5] To handle non-unique encodings of $\mathcal{V}$ in $\Sigma_2^{n_2}$, we assume that $\mathfrak{F}$-oracle returns the same multiset on $\mathcal{U}$ (though possibly via different encodings) when handed different encodings of the same $v \in \mathcal{V}$. Writing $\mathsf{K} : \mathcal{U} \to \mathbb{N}$ implies that K is represented as a multiset on $\Sigma_1^{n_1}$ but with the promise that all strings in its support are encodings of elements of $\mathcal{U}$.

Definition 7.3.4. Oracle Multi-dimensional Subset Sum with Repetition (OrMultiSSR)

Instance:

Explicit input

Alphabets Σ_1 and Σ_2;

Numbers $n_1, n_2 \in \mathbb{N}$, in unary; and

Multiset $\mathsf{K} : \mathcal{U} \to \mathbb{N}$, by listing the elements in its support and their multiplicities.

Oracles

$\equiv$ oracle for equality in $\mathcal{U}$ or $\mathcal{V}$, and

$\mathfrak{F}$-oracle oracle for the set $\mathfrak{F} = \{\mathsf{F}_v : \mathcal{U} \to \mathbb{N}\}_{v \in \mathcal{V}}$, indexed by $\mathcal{V}$.

3. The index set $\mathcal{V}$ will be the conjugacy classes of subgroups of G, whereas $\mathfrak{F}$ will be a set of multisets of conjugacy classes of subgroups of M.

4. In our application, $\Sigma_1 = S_n$ and $\Sigma_2 = S_m$. The universes $\mathcal{U}$ and $\mathcal{V}$ will be conjugacy classes of large subgroups of S_n and S_m, respectively. Each conjugacy class is non-uniquely encoded by generators of a subgroup in the class.

5. We will not need to test membership of a string from Σ^n in the universe.

Solution: A sub-multiset of $\mathcal{V}$ that defines a sub-multiset of $\mathfrak{F}$ summing to K, i.e., a multiset $\mathsf{L} : \mathcal{V} \to \mathbb{N}$ satisfying $\sum_{v \in \mathcal{V}} \mathsf{L}(v) \cdot \mathsf{F}_v = \mathsf{K}$.

Notation 7.3.5 (SubSum($\mathsf{K}, \mathfrak{F}$))**.** Again, we write SubSum for the set of solutions to an instance of OrMultiSSR, though the indexing is slightly different.

$$\text{SubSum}(\mathsf{K}, \mathfrak{F}) := \left\{ \mathsf{L} : \mathcal{U} \to \mathbb{N} \;\middle|\; \sum_{v \in \mathcal{V}} \mathsf{L}(v) \cdot \mathsf{F}_v = \mathsf{K} \right\}.$$

The length of the input is $\log|\Sigma_1| + \log|\Sigma_2| + n_1 + n_2 + \|\mathsf{K}\|_0 \cdot \log\|\mathsf{K}_\infty\| \cdot n_1 \log|\Sigma_1|$.

Due to non-unique encodings, checking whether a multiset L satisfies $\sum_{v \in \mathcal{V}} \mathsf{L}(v) \cdot \mathsf{F}_v = \mathsf{K}$ will actually require calling the $\equiv$ oracle, as the multisets on the left and right sides of the equation may be encoded differently.

7.3.2 Triangular MultiSSR

The Search Problem for OrMultiSSR with an additional "Triangular Condition" (and oracles corresponding to this condition) can be solved in polynomial time. We call this problem TriOrMultiSSR. This section defines TriOrMultiSSR. The next section will provide an algorithm that solves the TriOrMultiSSR Search Problem in polynomial time, proving Proposition 7.1.14.

Under the conditions of Theorem 7.1.9 (when $G = A_n$, $M \leq G$ has polynomial index, and the codomain S_m has exponentially bounded permutation domain size $m < 2^{n-1}/\sqrt{n}$), a HomExtSym instance φ reduces to an instance OMS_φ that satisfies the additional assumptions of TriOrMultiSSR. The additional oracles of TriOrMultiSSR can be efficiently answered (see Section 7.5).

Definition of TriOrMultiSSR

The triangular condition roughly says that the matrix for the corresponding (prohibitively

large) integer linear program is upper triangular.

Below we say that $\preccurlyeq$ is a **total preorder** if it is a reflexive and transitive relation with no incomparable elements.[6]

Definition 7.3.6. TRIANGULAR ORACLE MULTI-DIMENSIONAL SUBSET SUM WITH REPETITION (TRIORMULTISSR)

Input, Set, Oracles, Output: Same as ORMULTISSR.

Triangular Condition: $\mathcal{U}$ has a total preordering $\preccurlyeq$.

For every $v \in \mathcal{V}$, the multiset F_v contains a unique $\preccurlyeq$-minimal element $\tau(v) \in \mathcal{U}$.

The map $\tau : \mathcal{V} \to \mathcal{U}$ is injective.

Additional Oracles:

$\preccurlyeq$: compares two elements of $\mathcal{U}$, and

$\triangle : \mathcal{U} \to \mathcal{V} \cup \{\texttt{Error}\}$ inverts τ, i.e., on input $u \in \mathcal{U}$ it returns

$$\triangle(u) = \begin{cases} \text{the unique } v \in \mathcal{V} \text{ such that } \tau(v) = u & \text{if } v \text{ exists} \\ \texttt{Error} & \text{if no such } v \text{ exists.} \end{cases} \tag{7.1}$$

Integer program and uniqueness of solutions

Uniqueness of solutions for TRIORMULTISSR can be seen by looking at the integer linear program formulation, where variables correspond to $\mathcal{V}$ and constraints correspond to $\mathcal{U}$. The Triangular Condition implies that, for every variable ($v \in \mathcal{V}$), there exists a unique minimal constraint ($\tau(v) \in \mathcal{U}$) containing this variable. The ordering $\preccurlyeq$ on $\mathcal{U}$ gives an ordering $\preccurlyeq_{\mathcal{V}}$ on $\mathcal{V}$ by setting $v_1 \preccurlyeq_{\mathcal{V}} v_2$ when $\tau(v_1) \preccurlyeq \tau(v_2)$. Order the variables and constraints by $\preccurlyeq_{\mathcal{V}}$ and $\preccurlyeq$, respectively (break ties in $\preccurlyeq$ arbitrarily and have $\preccurlyeq_{\mathcal{V}}$ respect the tie-breaking of $\preccurlyeq$). The matrix for the corresponding linear program is upper triangular.

Hence, if the integer program has a solution, it is unique. It trivially follows that solving

6. A total order also imposes antisymmetry, i.e., if $x \preccurlyeq y$ and $y \preccurlyeq x$ then $x = y$. That is the assumption we omit.

the TriOrMultiSSR Search Problem also solves the corresponding Threshold-k Enumeration Problem.

7.3.3 TriOrMultiSSR *Search Problem*

Algorithm 4 (TriOrMultiSSR) below solves the TriOrMultiSSR Search Problem in polynomial time (Proposition 7.1.14). If viewing the problem as a linear program, the algorithm essentially solves the upper triangular system of equations by row reduction, except that the dimensions are too big and only oracle access is provided.

In each iteration, TriOrMultiSSR finds one minimal element u in $\mathrm{supp}(K)$. It removes the correct number m of copies of $\mathsf{F}_{\triangle(u)}$ from K, in order to remove all copies of u from K. If this operation fails, the algorithm returns 'no solution.' Meanwhile, $\mathsf{L}(\triangle(u))$ is updated in each iteration to record the number of copies of $\mathsf{F}_{\triangle(u)}$ removed.

There are three reasons the operation may fail. (1) Removing all copies of u from K may not be possible through removal of $\mathsf{F}_{\triangle(u)}$ (the number $m = \mathsf{K}(u)/\mathsf{F}_{\triangle(u)}$ of copies is not an integer). (2) K may not contain m copies of $\mathsf{F}_{\triangle(u)}$ (the operation $\mathsf{K} - m \cdot \mathsf{F}_{\triangle(u)}$ results in negative values). (3) $\triangle(u)$ returns `Error` (u is not in the range of τ).

Subroutines

min(S): min takes as input a subset $S \subset \Sigma_1^{n_1}$ and outputs one minimal element under $\preccurlyeq$. Using the $\preccurlyeq$ oracle, a min call can be executed in poly$(|S|)$-time.

Remove$(\mathsf{K}, \mathsf{F}, m)$: Remove takes as input multisets $\mathsf{F}, \mathsf{K} : \Sigma_1^{n_1} \to \mathbb{N}$ and a nonnegative integer m. It returns K after removing m copies of the multiset if possible, while accounting for non-unique encodings. Otherwise, it returns 'no solution.' Pseudocode for Remove is provided below.

Consolidate$(\mathsf{K}_1, \ldots, \mathsf{K}_n)$: Consolidate adjusts for non-unique encodings of $\mathcal{U} \to \mathbb{N}$ multisets as $\Sigma_1^{n_1} \to \mathbb{N}$ multisets. On the input encoded multisets $\mathsf{K}_1, \ldots, \mathsf{K}_n : \Sigma_1^{n_1} \to \mathbb{N}$,

CONSOLIDATE outputs multisets $\widetilde{\mathsf{K}}_1, \ldots, \widetilde{\mathsf{K}}_n : \Sigma_1^{n_1} \to \mathbb{N}$ that encode the same multisets of $\mathcal{U}$, but uniquely. In other words, $\widetilde{\mathsf{K}}_i$ satisfy $\widetilde{\mathsf{K}}_i = \mathsf{K}_i$, with their combined support $\dot{\bigcup}_i \operatorname{supp}(\widetilde{\mathsf{K}}_i) \subset \Sigma_1^{n_1}$ containing at most one encoding per element of $\mathcal{U}$.

Algorithm

Recall that we denote the empty multiset by $\{\{\}\}$. We give pseudocode for the REMOVE subroutine, followed by the main algorithm.

procedure REMOVE($\mathsf{K}, \mathsf{F}, m$)
  CONSOLIDATE(K, F) ▶ `Remove duplicate encodings within` $\operatorname{supp}(\mathsf{K}) \cup \operatorname{supp}(\mathsf{F})$.
  $\mathsf{K} \leftarrow \mathsf{K} - m \cdot \mathsf{F}$ ▶ `Execute as` $\mathsf{K}, \mathsf{F} : \Sigma_1^{n_1} \to \mathbb{Z}$`, assuming integer range`
  if K has negative values **then**
    return 'no solution'
  else return K
  end if
end procedure

Algorithm 4 Triangular Oracle MultiSS

1: **procedure** TRIORMULTISS(Σ_1, n_1, Σ_2, n_2, K, $\equiv$, $\preccurlyeq$, $\mathfrak{F}$-oracle, $\triangle$)
2: Initialize $\mathsf{L} = \{\{\}\}$ ▶ `L is the empty multiset of` $\Sigma_2^{n_2}$
3: CONSOLIDATE(K). ▶ `Remove duplicate encodings within` supp(K)
4: **while** $\mathsf{K} \neq \{\{\}\}$ **do**
5: $u \leftarrow$ MIN(supp(K)) ▶ u `is a minimal element of K`
6: **if** $\triangle(u) =$ `Error` **then**
7: **return** 'no solution'
8: **else**
9: $\mathsf{F} \leftarrow \mathfrak{F}\text{-oracle}_{\triangle(u)}$ ▶ `F is` F_v`, where` $\tau(v) = u$ `by Triangular Condition`
10: $m \leftarrow \frac{\mathsf{K}(u)}{\mathsf{F}(u)}$ ▶ m `is number of copies of F to remove from K.`
11: **if** $(m \notin \mathbb{N})$ or (REMOVE($\mathsf{K}, \mathsf{F}, m$) = 'no solution') **then**
12: **return** 'no solution'
13: **else**
14: $\mathsf{L}(\triangle(u)) \leftarrow \mathsf{L}(\triangle(u)) + m$
15: $\mathsf{K} \leftarrow$ REMOVE($\mathsf{K}, \mathsf{F}, m$)
16: **end if**
17: **end if**
18: **end while**
19: **return** L
20: **end procedure**

Analysis

The pre-processing step of Line 3 can be computed in time $|\text{supp}(\mathsf{K})|^2$, by pairwise comparisons. The **while** loop of Line 4 is executed exactly $|\text{supp}(\mathsf{K})|$ number of times, for each $u \in \text{supp}(\mathsf{K})$.

The CONSOLIDATE call in TRIORMULTISSR returns $\widetilde{\mathsf{K}} : \Sigma_1^{n_1} \to \mathbb{N}$, a different encoding

of the multiset K of $\mathcal{U}$, such that all elements of $\mathrm{supp}(\widetilde{\mathsf{K}})$ are uniquely encoded. This requires $\binom{|\mathrm{supp}(\mathsf{K})|}{2}$ pairwise comparisons, or, $< |\mathrm{supp}(\mathsf{K})|^2$ calls to the $\equiv$ oracle. Similarly, the CONSOLIDATE call in REMOVE can be achieved in $< |\mathrm{supp}(\mathsf{K}) \cup \mathrm{supp}(\mathsf{F})|^2$ calls to the $\equiv$ oracle.

7.4 Reduction of HOMEXTSYM to ORMULTISSR

We define the reduction from HOMEXTSYM to ORMULTISSR then prove the three parts of Theorem 7.1.12. Theorem 7.1.12 states the polynomial-time efficiency of the reduction, the bijection between classes of equivalent extensions in $\mathrm{HExt}(\varphi)$ and the set $\mathrm{SubSum}(\mathrm{OMS}_\varphi)$ of solutions to OMS_φ, and issues of defining an extension homomorphism $\widetilde{\varphi} \in \mathrm{HExt}(\varphi)$ from a solution $\mathsf{L} \in \mathrm{SubSum}(\mathrm{OMS}_\varphi)$.

Section 7.4.1 defines for notational convenience "(G, L)-actions," which describe permutation actions up to equivalence.

Section 7.4.2 presents the reduction from a HOMEXTSYM instance φ to the ORMULTISSR instance OMS_φ. The input to ORMULTISSR can be found and the oracles for ORMULTISSR can be answered in $\mathrm{poly}(n, m)$ time, proving Theorem 7.1.12 (a).

Section 7.4.3 proves the bijection described in Theorem 7.1.12 (b), assuming the transitive case. The transitive case is proved in Sections 7.4.4 and 7.4.5.

Section 7.4.6 Theorem 7.1.12 (c), regarding the algorithmic details of defining $\widetilde{\varphi} \in \mathrm{HExt}(\varphi)$ given a solution in $\mathrm{SubSum}(\mathrm{OMS}_\varphi)$.

7.4.1 Equivalent extensions and definition of (G, L)-actions

In this section we characterize equivalence of two group actions and, in particular, fix notation to describe equivalence.

Definition 7.4.1 (Equivalent permutation actions)**.** Two permutation actions $G \curvearrowright \Omega$ and $G \curvearrowright \Gamma$ are **equivalent** if there exists a bijection $\zeta : \Omega \to \Gamma$ such that $\zeta(\omega^g) = (\zeta(\omega))^g$ for all $g \in G$ and $\omega \in \Omega$.

Note that two permutation actions $\varphi_1, \varphi_2 : G \to S_m$ of G on the same domain are equivalent if there exists $\zeta \in S_m$ such that $\varphi_1(g) = \zeta^{-1}\varphi_2(g)\zeta$ for all $g \in G$.

The Introduction defined two homomorphisms $\widetilde{\varphi}_1, \widetilde{\varphi}_2 : G \to S_m$ as "equivalent extensions" of $\varphi : M \to S_m$ if they both extend φ and if they are equivalent as actions. The following definition is equivalent to that definition provided in the Introduction.

For groups $M \leq G$, the **centralizer** of M in G is given by $C_G(M) = \{g \in G : (\forall x \in M)(gx = xg)\}$.

Definition 7.4.2 (Equivalent extensions)**.** Let $M \leq G$ and $\varphi : M \to S_m$. We say that $\widetilde{\varphi}_1$ and $\widetilde{\varphi}_2$ are **equivalent extensions of** φ if there exists $\zeta \in C_{S_m}(\varphi(M))$ such that $\zeta^{-1}\widetilde{\varphi}_2(g)\zeta = \widetilde{\varphi}_1(g)$ for all $g \in G$.

Next we consider the equivalence of transitive group actions, through their point stabilizers. A G-action on Ω is **transitive** if $\omega^G = \Omega$ for all $\omega \in \Omega$, i.e., for every pair $\omega_1, \omega_2 \in \Omega$, there is a group element $g \in G$ satisfying $\omega_1^g = \omega_2$. Lemma 7.4.3 is Lemma 1.6A in [DM96].

Lemma 7.4.3. *Suppose G acts transitively on the sets Ω and Γ. Let L be the stabilizer of a point in the first action. Then, the actions are equivalent if and only if L is the stabilizer of some point in the second action.*

Recall that we denote the conjugacy class of a subgroup $L \leq G$ by $[L]$, so L is conjugate to L_1 if and only if $[L] = [L_1]$. We find all point stabilizers are conjugate, and all conjugate subgroups are point stabilizers.

Fact 7.4.4. Let L be a point stabilizer of a transitive G-action on Ω. A subgroup L_1 is conjugate to L ($[L_1] = [L]$) if and only if L_1 is also the stabilizer of a point in Ω.

All transitive G-actions are equivalent to one of its natural actions on cosets, ρ_L defined below.

Example 7.4.5 (Natural actions on cosets)**.** *For $L \leq G$, we denote by ρ_L the natural action of G on $L\backslash G$. More specifically, an element $g \in G$ acts on a coset $Lh \in L\backslash G$ as $(Lh)^g := L(hg)$.*

We see that the equivalence class of a transitive action is determined by the conjugacy class of its point stabilizers.

Corollary 7.4.6. *Consider a transitive G-action $\varphi : G \to \mathrm{Sym}(\Omega)$. Let $L \leq G$. The following are equivalent.*

(1) φ is equivalent to ρ_L.

(2) L is a point stabilizer of the G-action.

(3) Some $L_1 \leq G$ satisfying $L_1 \sim L$ is a point stabilizer of the G-action.

(4) φ is equivalent to ρ_{L_1} for $L_1 \sim L$.

Motivated by Corollary 7.4.6, we will define the notion of "(G, L)-actions," which describe transitive G-actions up to equivalence. This definition will be generalized to intransitive actions as "(G, L)-actions." The $\mathsf{L} : \mathrm{Sub}(G) \to \mathbb{N}$ is a multiset of subgroups of G, describing point stabilizers of the action. We make this more precise.

Recall that we write $[L]_G$ to denote the conjugacy class of the subgroup L in G.

Definition 7.4.7 ((G, L)-action)**.** Let $\varphi : G \to \mathrm{Sym}(\Omega)$ be a transitive action. Let $L \leq G$. We say that φ is a (G, L)**-action** if φ is equivalent to ρ_L. We say that φ is a $(G, [L])$**-action** if φ is a (G, L)-action.

By Corollary 7.4.6, we see that G-action is a (G, L)-action if and only if L is a point stabilizer of the action. Moreoever, a (G, L)-action is a (G, L_1)-action if and only if $[L] = [L_1]$

So, we can speak of $(G,[L])$-actions and make no distinction between $(G,[L])$-action and (G,L)-actions.

In what follows we introduce notation to describe equivalence between intransitive actions.

Definition 7.4.8 ((G,L)-action)**.** Let $\varphi : G \to \mathrm{Sym}(\Omega)$ be a group action. Denote by Ω_i the orbits of G, so $\Omega = \Omega_1 \dot\cup \cdots \dot\cup \Omega_d$. Let $\mathsf{L} : \mathrm{Sub}(G) \to \mathbb{N}$ be a multiset listed as $\mathsf{L} = \{\{L_i \leq G\}\}_{i=1}^{d}$. We say the action of G on Ω is a (G,L)**-action** if G acts on Ω_i as a (G,L_i)-action for all $1 \leq i \leq d$.[7]

Again, the multiset of conjugacy classes of the elements of L determines the G-action up to equivalence.

Notation 7.4.9. Let $\mathsf{L} = \{\{L_1,\ldots,L_k\}\}$ be a multiset of subgroups of G. We denote by $[\mathsf{L}]_G = \{\{[L_1]_G,\ldots,[L_k]_G\}\}$ the multiset of conjugacy classes of the subgroups of L.

In other words, for a multiset $\mathsf{L} : \mathrm{Sub}(G) \to \mathbb{N}$, denote by $[\mathsf{L}]_G : \mathrm{Conj}(G) \to \mathbb{N}$ the multiset found by replacing every element $L \in \mathsf{L}$ by $[L]_G$. Multiplicities of subgroup conjugacy classes $[L]$ in the multiset $[\mathsf{L}]$ satisfy $[\mathsf{L}]([L]) = \sum_{L\in[L]} \mathsf{L}(L)$. We may write $[L]$ for $[L]_G$ if G is understood.

Definition 7.4.10 (Conjugate multisets)**.** We say that two multisets $\mathsf{L},\mathsf{L} : \mathrm{Sub}(G) \to \mathbb{N}$ are **conjugate** if $[\mathsf{L}] = [\mathsf{L}_1]$. In other words, there exists a bijection $\pi : \mathsf{L} \to \mathsf{L}_1$ such that $\pi(L) \sim_G L$ for all $L \in \mathsf{L}$.[8]

Conjugate multisets describes group actions up to equivalence, as we see in the following corollary to our definitions and Corollary 7.4.6.

---

7. The multiset $\mathsf{L} : \mathrm{Sub}(G) \to \mathbb{N}$ contains one point stabilizer per orbit of the G-action. Viewing L as a multiset is essential. For example, $\mathsf{L} = \{\{G\}\}$ describes the trivial action of G on one point, whereas $\mathsf{L} = \{\{G,G\}\}$ describes the trivial action of G on two points.

8. This definition does not require conjugacy of all pairs simultaneously via the one element of G.

Corollary 7.4.11. *Let* $\mathsf{L}_1, \mathsf{L}_2 : \mathrm{Sub}(G) \to \mathbb{N}$. *The following are equivalent.*

- L_1 *and* L_2 *are conjugate, or* $[\mathsf{L}_1] = [\mathsf{L}_2]$.
- *A* (G, L_1)*-action is equivalent to a* (G, L_2)*-action.*
- *A* (G, L_1)*-action is also a* (G, L_2)*-action.*

So, we can speak of $(G, [\mathsf{L}])$-actions and make no distinction between $(G, [\mathsf{L}])$-action and (G, L)-actions.

7.4.2 Reduction

This section addresses the $\mathrm{poly}(n, m)$-time reduction from HomExtPerm to OrMultiSSR.

Remark 7.4.12 (Meaning of "reduction")**.** As usual, we will find the explicit inputs to OrMultiSSR from a HomExtSym instance in $\mathrm{poly}(n, m)$ time. To account for the oracles in OrMultiSSR, we provide answers to its oracles in $\mathrm{poly}(n, m)$-time as well.

Recall that $\mathrm{Sub}(G)$ denotes the set of subgroups of G and $\mathrm{Conj}(G)$ denotes the set of conjugacy classes of subgroups of G. Denote by $\mathrm{Sub}^{\leq \mathrm{m}}(G)$ the set of subgroups of G with index bounded by m. Denote by $\mathrm{Conj}^{\leq \mathrm{m}}(G)$ the set of conjugacy classes of subgroups of G with index bounded by m.

Construction of OMS_φ We define $\mathcal{U}, \mathcal{V}, [\mathsf{K}]$ and encodings $\Sigma_1^{n_1}, \Sigma_2^{n_1}$ of the OrMultiSSR instance OMS_φ.

$\mathcal{U}$: $\mathrm{Conj}^{\leq \mathrm{m}}(M)$.

$\mathcal{V}$: $\mathrm{Conj}^{\leq \mathrm{m}}(G)$.

Encoding of $\mathcal{U}$: $\Sigma_1 = M$ and $n_1 = 2n$. Conjugacy classes in $\mathcal{U}$ of subgroups are encoded by subgroups in $\mathrm{Sub}^{\leq \mathrm{m}}(M)$, which are themselves encoded by a list of at most $2n$ generators

Encoding of $\mathcal{V}$: Likewise, with $\Sigma_2 = G$ and $n_2 = 2n$.

[K]: Let $\mathsf{K} : \mathrm{Sub}^{\leq \mathrm{m}}(M) \to \mathbb{N}$ be a multiset containing one point stabilizer per orbit of the action $\varphi : M \to S_m$. So, $[\mathsf{K}] : \mathrm{Conj}^{\leq \mathrm{m}}(M) \to \mathbb{N}$ is the multiset of conjugacy classes, as in Notation 7.4.9.

Notational issues. Using $[\mathsf{K}]$ versus K reflects the non-unique encoding of $\mathcal{U} = \mathrm{Conj}^{\leq \mathrm{m}}(M)$ by $\mathrm{Sub}^{\leq \mathrm{m}}(G)$ (and $\mathcal{V} = \mathrm{Conj}^{\leq \mathrm{m}}(G)$ by $\mathrm{Sub}^{\leq \mathrm{m}}(G)$), adhering to Notation 7.2.3 and 7.4.9. A conjugacy class $[K] \in \mathcal{U}$ will be encoded by $K \in \mathrm{Sub}^{\leq \mathrm{m}}(M)$. A multiset $[\mathsf{K}] : \mathcal{U} \to \mathbb{N}$ will be encoded by $\mathsf{K} : \mathrm{Sub}^{\leq \mathrm{m}}(M) \to \mathbb{N}$.

Calculating $[\mathsf{K}]$

Calculating $[\mathsf{K}] : \mathcal{U} \to \mathbb{N}$ from $\varphi : M \to S_m$: Consider the decomposition $[m] = \Sigma_1 \dot{\cup} \ldots \dot{\cup} \Sigma_s$ of $[m]$ into its M-orbits under the action described by φ. Choose one element $x_i \in \Sigma_i$ per orbit.[9] Then, calculate the multiset K by finding the point stabilizers $\mathsf{K} := \{\{M_{x_i} : i = 1 \ldots s\}\}$. So, calculating K can be accomplished in poly(n)-time by Proposition 2.4.5.

Answering $\equiv$ oracle

The $\equiv$ oracle is given by checking conjugacy of two subgroups in $\mathrm{Sub}^{\leq \mathrm{m}}(M)$, which can be accomplished in poly(n, m)-time by Proposition 2.4.7.

Answering $\mathfrak{F}$-oracle oracle.

The set $\mathfrak{F}$ is indexed by $\mathcal{V} = \mathrm{Conj}^{\leq \mathrm{m}}(G)$. $\mathfrak{F}$-oracle takes as input $[L] \in \mathrm{Conj}^{\leq \mathrm{m}}(G)$ (represented by a $L \in \mathrm{Sub}^{\leq \mathrm{m}}(G)$) and returns $[\mathsf{F}_L] : \mathrm{Conj}^{\leq \mathrm{m}}(M) \to \mathbb{N}$ (represented by $\mathsf{F}_L : \mathrm{Sub}^{\leq \mathrm{m}}(M) \to \mathbb{N}$), defined below. The multiset $\mathsf{F}_L : \mathrm{Sub}^{\leq \mathrm{m}}(M) \to \mathbb{N}$ labels the M-actions that extend to (G, L)-actions.

9. The choice of x_i will not affect the correctness of the reduction.

Definition 7.4.13 ($\mathsf{F}_L(\boldsymbol{\sigma})$)**.** Let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_d)$ be a list of double coset representatives for $L\backslash G/M$. We define the multiset $\mathsf{F}_L^M(\boldsymbol{\sigma}) : \mathrm{Sub}(M) \to \mathbb{N}$ by

$$\mathsf{F}_L^M(\boldsymbol{\sigma}) = \mathsf{F}_L := \{\{\sigma_i^{-1} L \sigma_i \cap M : i = 1 \ldots d\}\}.$$

In the context of extending an M-action $\varphi : M \to S_m$ to a G-action, M is understood, so we drop the superscript and write F_L.

The $\mathfrak{F}$-oracle oracle is well-defined. The choice $\boldsymbol{\sigma}$ of double coset representatives will not matter (see Remark 7.4.23). Moreover, if $[L]_G = [L_1]_G$ then $[\mathsf{F}_L]_M = [\mathsf{F}_{L_1}]_M$, so the $\mathfrak{F}$-oracle respects non-unique encodings (see Lemma 7.4.24). Section 7.4.4 gives further discussion on the properties of F_L.

The $\mathfrak{F}$ oracle can be answered in $\mathrm{poly}(n, m)$-time by Proposition 2.4.8.

7.4.3 Combinatorial condition for extensions

We are now equipped to state the central technical result. It relates M-actions to extension G-actions by describing how M-orbits may be grouped to form G-orbits.

First, we address the case of transitive extensions.

As in Definition 7.4.13, $\mathsf{F}_L : \mathrm{Sub}(M) \to \mathbb{N}$ denotes the multiset returned by the oracle $\mathfrak{F}$-oracle on input $L \in \mathrm{Sub}(G)$. The multiset F_L describes exactly how M-orbits must be collected to form one (G, L)-orbit.

Lemma 7.4.14 (Characterization of transitive extensions)**.** *Let $M \leq G$ and $m \in \mathbb{N}$. Let $\varphi : M \to S_m$ be an M-action. Under these circumstances, φ extends to a (G, L)-action if and only if φ is a (M, F_L)-action.*

This follows from Corollary 7.4.22 and Proposition 7.4.27 below.

Remark 7.4.15. To rephrase Lemma 7.4.14, an (M, K)-action extends to a transitive (G, L)-action if and only if $[\mathsf{K}] = [\mathsf{F}_L]$ (see Corollary 7.4.11).

The following result on intransitive actions is a corollary to Lemma 7.4.14.

Theorem 7.4.16 (Key technical lemma: characterization of HomExtSym with codomain S_m)**.** *Let $M \leq G$ and $m \in \mathbb{N}$. Let $\varphi : M \to S_m$ be an $(M, [\mathsf{K}])$-action, where $[\mathsf{K}] : \mathrm{Conj}(M) \to \mathbb{N}$. Under these circumstances, φ extends to a $(G, [\mathsf{L}])$-action, where $\mathsf{L} : \mathrm{Conj}(G) \to \mathbb{N}$, if and only if $[\mathsf{K}]$ is an $[\mathsf{L}]$-linear combination of elements in $\mathfrak{F}$, i.e.,*

$$[\mathsf{K}] = \sum_{L \in \mathsf{L}} [\mathsf{F}_L] = \sum_{[L] \in \mathrm{Conj}^{\leq m}(G)} \mathsf{L}([L])[\mathsf{F}_L]. \tag{7.2}$$

We have found that an (M, K)-action extends exactly if K is a Subset Sum with Repetition of $\{\mathsf{K}_L\}$. Compare Equation (7.2) to the definition of $\mathrm{SubSum}(\mathrm{OMS}_\varphi)$ (see Notation 7.3.2 and the reduction of Section 7.4.2). We have found the following.

Corollary 7.4.17. *Let $M \leq G$ and $m \in \mathbb{N}$. Let $\varphi : M \to S_m$ be an $(M, [\mathsf{K}])$-action, where $[\mathsf{K}] : \mathrm{Conj}(M) \to \mathbb{N}$. Under these circumstances, φ extends to a G-action if and only if $\mathrm{SubSum}(\mathrm{OMS}_\varphi)$ is nonempty.*

So, $\mathrm{HExt}(\varphi)$ is nonempty if and only if $\mathrm{SubSum}(\mathrm{OMS}_\varphi)$ is nonempty.

Remark 7.4.18. In fact, we have found something even stronger. The multisets $[\mathsf{L}]$ satisfying Equation (7.2) are exactly the elements in $\mathrm{SubSum}(\mathrm{OMS}_\varphi)$. A multiset $[\mathsf{L}] : \mathrm{Conj}(G) \to \mathbb{N}$ satisfies Equation (7.2) if and only if $\mathrm{HExt}(\varphi)$ contains a (G, L)-action extending φ. This notation identifies all equivalent extensions, so we have found a bijection between the solutions in $\mathrm{SubSum}(\mathrm{OMS}_\varphi)$ and classes of equivalent extensions in $\mathrm{HExt}(\varphi)$, as promised by Theorem 7.1.12 (b).

7.4.4 (G, L)-actions induce (M, F_L)-actions

Let $M \leq G$. This section describes the M-action found by restricting a (transitive) G-action. If $\varphi : G \to \mathrm{Sym}(\Omega)$ describes a G-action on Ω, we will call the M-action on Ω described by $\varphi|_M : M \to \mathrm{Sym}(\Omega)$ the **M-action induced by φ**.

We identify the permutation domain of a (G, L)-action by the right cosets $L\backslash G$. This is possible due to equivalence with ρ_L, the action on cosets of L. We now describe the behavior of the induced M-action on $L\backslash G$.

Remark 7.4.19. Let $M, L \leq G$. Consider the natural M-action on $L\backslash G$ (the M-action induced by ρ_L). We have that (Lg_1) and (Lg_2) belong to the same M-orbit if and only if $Lg_1M = Lg_2M$, i.e., if g_1 and g_2 belong to the same double coset of $L\backslash G/M$.

Lemma 7.4.20. *Let $g_0 \in G$. Let $M, L \leq G$. The action of M on the orbit $(Lg_0)^M$ of Lg_0 in $L\backslash G$ is equivalent to the action of M on $K\backslash M$, where $K := g_0^{-1}Lg_0 \cap M$. The bijection is given by $La \leftrightarrow Kg_0^{-1}a$.*

Proof. Both actions are transitive. Let $\zeta : (Lg_0)^M \to K\backslash M$ be defined by $\zeta(Lg) = Kg_0^{-1}g$ for all $g \in Lg_0M$. For all $a \in M$,

$$\zeta((Lg)^a) = \zeta(L(ga)) = Kg_0^{-1}(ga) = (Kg_0^{-1}g)^a = \zeta(Lg)^a.$$

□

From Remark 7.4.19 and Lemma 7.4.20, we have found the (possibly non-transitive) natural action of M on $L\backslash G$ satisfies the following.

(1) The number of orbits is $|L\backslash G/M|$.

(2) The point stabilizer of $Lg \in L\backslash G$ under the M-action is $M_{Lg} = g^{-1}Lg \cap M$.

We restate the definition of F_L, which we now see describes the M-action on $L\backslash G$.

Definition 7.4.21 ($\mathsf{F}_L(\boldsymbol{\sigma})$)**.** Let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_d)$ be a list of double coset representatives for $L\backslash G/M$. We define the multiset $\mathsf{F}_L^M(\boldsymbol{\sigma}) : \mathrm{Sub}(M) \to \mathbb{N}$ by

$$\mathsf{F}_L^M(\boldsymbol{\sigma}) = \mathsf{F}_L := \{\{\sigma_i^{-1}L\sigma_i \cap M : i = 1 \ldots d\}\}.$$

If the subgroup M is understood, we drop the superscript M.

From Remark 7.4.19 and Lemma 7.4.20, we find that (G, L)-actions restrict to (M, F_L)-actions.

Corollary 7.4.22. *Let $M, L \leq G$. Let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_d)$ be a set of double coset representatives of $L\backslash G/M$. If G acts on Ω as a (G, L)-action, then the induced action of M on Ω is an $(M, \mathsf{F}_L(\boldsymbol{\sigma}))$-action. In fact, the M-action induced by a $(G, [L])$-action is an $(M, [\mathsf{F}_L])$-action.*

The last sentence of Corollary 7.4.22 follow from Corollary 7.4.11 and Lemma 7.4.24 below, which say that the choice σ of double coset representatives and the choice L of conjugacy class representative make no difference to the conjugacy class $[\mathsf{F}_L(\boldsymbol{\sigma})]$.

We now see the $\mathfrak{F}$-oracle is well-defined.

Remark 7.4.23. For any two choices $\boldsymbol{\sigma}$ or $\boldsymbol{\sigma}'$ of double cosets representatives of $L\backslash G/M$, we have that $[\mathsf{F}_L(\boldsymbol{\sigma})]_M = [\mathsf{F}_L(\boldsymbol{\sigma}')]_M$. So, we may reference (M, F_L)-actions without specifying $\boldsymbol{\sigma}$.

In fact, only the conjugacy class of L matters in determining the conjugacy class of F_L. In particular, the $\mathfrak{F}$-oracle oracle is well-defined.

Lemma 7.4.24. *Let $M, L, L_1 \leq G$. If $[L]_G = [L_1]_G$, then $[\mathsf{F}_L^M]_M = [\mathsf{F}_{L_1}^M]_M$. In other words, if L and L_1 are conjugate in G, then F_L^M and $\mathsf{F}_{L_1}^M$ are conjugate in M.*

Proof. The natural G-actions on $L\backslash G$ and $L_1\backslash G$ are equivalent by Corollary 7.4.6. Thus, the induced M-action on $L\backslash G$ and the induced M-action on $L_1\backslash G$ are equivalent, using the same bijection on the domain. But, the M-action on $L\backslash G$ is an (M, F_L)-action and the M-action on $L_1\backslash G$ is an (M, F_{L_1})-action. By Corollary 7.4.11, we find $[\mathsf{F}_L]_M = [\mathsf{F}_{L_1}]_M$. □

7.4.5 Gluing M-orbits to find extensions to G-actions

In this section we will see that any (M, F_L)-action will extend to a (G, L)-action. It is intuitively straightforward that the (G, L)-action restricts to a (M, F_L)-action, which is equiva-

lent to the original (M, F_L)-action by Corollary 7.4.11. We make the equivalence explicit by specifying the bijection between permutation domains.

Let $M, L \leq G$ and $\sigma \in G$. Lemma 7.4.20 gave an equivalence between the M-action on the orbit $(L\sigma)^M$ of $(L\sigma)$ in $L\backslash G$ and the natural M-action on $F_i\backslash M$, where $F_i = \sigma^{-1}L\sigma \cap M$. We extend this equivalence here.

Construction 7.4.25 (Equivalence ζ)**.** Fix a choice $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_d)$ of double coset representatives for $L\backslash G/M$. Recall the definition $\mathsf{F}_L(\boldsymbol{\sigma}) = \{\{F_i : i = 1 \ldots d\}\}$, where $F_i = \sigma_i^{-1}L\sigma_i \cap M$. Define the map ζ by

$$\zeta : \left(\dot{\bigcup}_i F_i\backslash M\right) \to L\backslash G, \quad \zeta : F_i\tau \mapsto L\sigma_i\tau.$$

That ζ is a permutation equivalence of the M-actions on the two sets follows immediately from Lemma 7.4.20.

Corollary 7.4.26. *The map ζ given in Construction 7.4.25 is a permutation equivalence of the M-action.*

The next result is almost immediate from our discussion above.

Proposition 7.4.27 (Gluing)**.** *Let $L, M \leq G$. Suppose that $\varphi : M \to \mathrm{Sym}(\Omega)$ describes an (M, F_L)-action. Then, there exists an extension $\widetilde{\varphi} : G \to \mathrm{Sym}(\Omega)$ of φ that is a (G, L)-action.*

Proof. We label the M-orbits of Ω by the cosets $F_i\backslash M$, use ζ to label Ω by $L\backslash G$, then let G act on Ω in its natural action on $L\backslash G$. The output is the evaluation of $\widetilde{\varphi}$ on the generators of G as given by $\widetilde{\varphi}(g_j) : La \mapsto Lag_j$. □

7.4.6 Defining one extension from SubSum *solution*

We address Theorem 7.1.12 (c), defining an extension $\widetilde{\varphi} \in \mathrm{HExt}(\varphi)$ given a solution $[L] \in \mathrm{SubSum}(\mathrm{OMS}_\varphi)$.

First, Construction 7.4.25 gives an explicit bijection ζ that, given an (M, F_L)-action, defines an extension (G, L)-action. This bijection ζ can be computed in $\text{poly}(n, m)$ time.

The issue remains of finding the F_L "grouping" of the M-orbits that respect the orbits of the (G, L)-action.

Fix a HomExtSym instance φ. Fix $\mathsf{L} : \text{Sub}(G) \to \mathbb{N}$ in $\text{SubSum}(\text{OMS}_\varphi)$, so L satisfies Equation (7.2). Recall that L is represented by listing the subgroups in its support and their multiplicities. Since $|\text{supp}(\mathsf{L})| \leq \|\mathsf{L}\|_1$, the number of orbits of the G-action, we find that $|\text{supp}(\mathsf{L})| \leq m$.

It takes $\text{poly}(n, m)$ time to compute the multiset K of point stabilizers (one point stabilizer per orbit), and label $[m]$ by $\dot{\bigcup}_{K \in \mathsf{K}} K\backslash M$, the right cosets in M of the subgroups in K. Compute the multiset $\sum_{L \in \mathsf{L}} [\mathsf{F}_L]$ in $\text{poly}(n, m, \|\mathsf{K}\|_1)$-time, by calling the $\mathfrak{F}$ oracle.

By Theorem 7.4.16, $[\mathsf{K}] = \sum_{L \in \mathsf{L}} [\mathsf{F}_L]$. Via at most m^2 $\text{poly}(n, m)$-time conjugacy checks between subgroups in M, compute the map $\pi : \mathsf{K} \leftrightarrow \sum_{L \in \mathsf{L}} \mathsf{F}_L$ that identifies conjugate subgroups. Compute the conjugating element for each pair.

For each $L \in \mathsf{L}$, use the map ζ of Construction 7.4.25 to label Ω by right cosets of elements in L. Define $\widetilde{\varphi}$ by its natural action on cosets.

7.5 Reducing to TriOrMultiSSR

In this section we address the additional assumptions and oracles for TriOrMultiSSR, in the case that φ is an instance of HomExtSym satisfying the conditions of Theorem 7.1.9 ($M \leq G = A_n$, $|G : M| = \text{poly}(n)$, and codomain S_m with $m < 2^{n-1}/\sqrt{n}$).

Ordering, the $\preccurlyeq$ oracle

The ordering $\preccurlyeq$ on conjugacy classes in $\mathcal{U} = \text{Conj}^{\leq \text{m}}(M)$ is the ordering given by the index of subgroups contained in each class, i.e., $[K_1] \preccurlyeq [K_2]$ whenever $|M : K_1| \leq |M : K_2|$.

This relation is clearly a total preorder.

$\preccurlyeq$ oracle: The index of a subgroup $K \leq M$ can be computed in poly(n)-time by Proposition 2.4.5. The $\preccurlyeq$ compares two conjugacy classes in $\mathrm{Conj}^{\leq \mathrm{m}}(M)$ by comparing the indices of two representatives.

Triangular condition, the $\triangle$ oracle

Here we define the $\triangle$ oracle on $\mathcal{U} = \mathrm{Conj}^{\leq \mathrm{m}}(M)$ (Construction 7.5.1), analyze its efficiency (Remark 7.5.2), then prove its correctness (Lemma 7.5.4). The assumptions of Theorem 7.1.9 are essential.

First we set up some notation. By the assumptions of Theorem 7.1.9, $G = A_n$ and $M \leq G$ satisfies $|G : M| = \mathrm{poly}(n)$. Assume more specifically that $|G : M| < \binom{n}{r}$, for constant r. By Jordan-Liebeck (Theorem 2.4.2) we find that $(A_n)_{(\Sigma)} \leq M \leq (A_n)_{\Sigma}$ for some $\Sigma \subseteq [n]$ with $|\Sigma| < r$. Fix this subset $\Sigma \subset [n]$.

Recall that, for a subset $\Sigma \subseteq [n]$ that is invariant under action by the permutation group $M \leq S_n$, we denote by $M^{\Sigma} \leq \mathrm{Sym}(\Sigma)$ the induced permutation group of the M-action on Σ.

Construction 7.5.1 ($\triangle$ oracle)**.** We define a map $\triangle : \mathrm{Sub}^{\leq \mathrm{m}}(M) \to \mathrm{Sub}^{\leq \mathrm{m}}(G)$.[10] Let $K \in \mathrm{Sub}^{\leq \mathrm{m}}(M)$. By Jordan-Liebeck, we find that $(A_n)_{(\Gamma)} \leq K \leq (A_n)_{\Gamma}$ for $\Gamma \subseteq [n]$ with $|\Gamma| < n/2$. There are two cases. If there is a subset $\Sigma_0 \subseteq \Gamma$ such that $K^{\Sigma_0} = M^{\Sigma}$, then let $\bar{\Gamma} = \Gamma \setminus \Sigma_0$ and

$$\triangle(K) = \begin{cases} \mathrm{Alt}([n] \setminus \bar{\Gamma}) \times K^{\bar{\Gamma}} & \text{if } K^{\bar{\Gamma}} \text{ is even} \\ \text{the subgroup of index 2 in } \mathrm{Sym}([n] \setminus \bar{\Gamma}) \times K^{\bar{\Gamma}} & \text{otherwise} \end{cases}. \tag{7.3}$$

If such a Σ_0 does not exist, then let $\triangle(K) = \mathsf{Error}$.

10. Though the $\triangle$ oracle returns an element of $\mathrm{Conj}^{\leq \mathrm{m}}(G)$ on an input from $\mathrm{Conj}^{\leq \mathrm{m}}(M)$, these conjugacy classes are represented by subgroups. So, the $\triangle$ oracle should return an element of $\mathrm{Sub}^{\leq \mathrm{m}}(G)$ on an input from $\mathrm{Sub}^{\leq \mathrm{m}}(M)$, while respecting conjugacy.

Remark 7.5.2 (Efficiency of $\triangle$ oracle)**.** Answering the $\triangle$ oracle of Construction 7.5.1 requires finding orbits, finding the induced action on orbits, and checking permutation equivalence. These can be accomplished in $\text{poly}(n, m)$ time.

Remark 7.5.3. The $\triangle$ oracle is well-defined as a $\text{Conj}^{\leq \text{m}}(M) \to \text{Conj}^{\leq \text{m}}(G)$ map.

Now, we prove that $\triangle$ as defined in Definition 7.5.1 satisfies the conditions of TriOrMultiSSR. In other words, the equivalence class of the M-action on its longest orbit uniquely determines the equivalence class of the transitive G-action and this correspondence is injective. Lemma 7.5.4 makes this more precise.

Lemma 7.5.4. *Let $M \leq G = A_n$ have index $|G : M| \leq \binom{n}{u}$. Let G act on Ω transitively, with degree $|\Omega| < \binom{n}{v}$. Assume $u + v < n/2$. If K_0 is a point stabilizer of the induced M action on its longest orbit, then $\triangle(K_0)$ is a point stabilizer of the G-action on Ω.*[11]

In other words, the conclusion is that, if M acts on its longest orbit as an (M, K_0)-action, then G acts as a $(G, \triangle(K_0))$-action.

We defer the proof of Lemma 7.5.4 to present a few useful claims.

Claim 7.5.5. *If $(A_n)_{(\Sigma)} \leq L \leq (A_n)_\Sigma$, then the pair (Σ, L^Σ) determines L.*

Proof. We have two cases. Either $L = (A_n)_{(\Sigma)} \times L^\Sigma = A_{n-|\Sigma|} \times L^\Sigma$, or L is an index 2 subgroup of $(S_n)_{(\Sigma)} \times L^\Sigma = S_{n-|\Sigma|} \times L^\Sigma$. In the first case, all permutations in L^Σ must be even. In the second case, L^Σ must contain an odd permutation. □

Claim 7.5.6. *Suppose that $(A_n)_{(\Sigma)} \leq L \leq (A_n)_\Sigma$ and $(A_n)_{(\Gamma)} \leq M \leq (A_n)_\Gamma$ for $\Gamma \cap \Sigma = \emptyset$. Then, $L^\Sigma = (L \cap M)^\Sigma$. (Equivalently, $M^\Gamma = (L \cap M)^\Gamma$.)*

Proof. The inclusion $\supseteq$ is obvious. We show $\subseteq$.

Let $\sigma \in L^\Sigma$. View σ as a permutation in S_n. Let $\Sigma \subseteq [n]$ be such that $[n] = \Gamma \dot\cup \Sigma \dot\cup \Sigma$. Consider the set $T = \{\tau \in S_n : \text{supp}(\tau) \subseteq \Sigma \text{ and } \operatorname{sgn}\tau = \operatorname{sgn}\sigma\}$.

We see that for all $\tau \in T$, $\sigma\tau \in M \cap L$. Thus, $\sigma \in (M \cap L)^\Gamma$. □

11. If M and K_0 are known, then $\triangle(K_0)$ is uniquely determined.

Proof of Lemma 7.5.4. Let L be a point stabilizer of G acting on Ω. Since $|\Omega| < \binom{n}{v}$, by Jordan-Liebeck Theorem 2.4.2, there exists a subset $\bar{\Gamma} \subset [n]$ such that $(A_n)_{(\bar{\Gamma})} \leq L \leq (A_n)_{\bar{\Gamma}}$ and $|\bar{\Gamma}| < v$. Similarly, there exists $\Sigma \subset [n]$ such that $(A_n)_{(\Sigma)} \leq M \leq (A_n)_{\Sigma}$ and $|\Sigma| < u$. Fix $\bar{\Gamma}$ and Σ.

By Theorem 7.4.16, we find that the point stabilizers of the M-action on Ω are described by F_L. By Definition 7.4.13 and Corollary 7.4.11, we find that

$$K_0 = \text{argmax}\{|M : K| : K \in \mathsf{F}_L\} \sim_M \text{argmin}\{|K| : K = g^{-1}Lg \cap M \text{ for } g \in G\}.$$

But, $|g^{-1}Lg \cap M|$ is minimized when $g \in G = A_n$ satisfies $\Gamma^g \cap \Sigma = \emptyset$. Fix this g. By Claims 7.5.5 and 7.5.6 applied to $g^{-1}Lg$ and M, we find that

$$g^{-1}Lg = \begin{cases} \text{Alt}([n] \setminus \bar{\Gamma}) \times K^{\bar{\Gamma}} & \text{if } K^{\bar{\Gamma}} \text{ is even} \\ \text{the subgroup of index 2 in } \text{Sym}([n] \setminus \bar{\Gamma}) \times K^{\bar{\Gamma}} & \text{if } K^{\bar{\Gamma}} \text{ contains an odd permutation} \end{cases}. \tag{7.4}$$

In other words, we have found that $g^{-1}Lg = \triangle(K_0)$, i.e., $L \sim_G \triangle(K_0)$. It follows that the G-action on Ω is a $(G, \triangle(K_0))$-action. □

7.6 Generating extensions within one equivalence class

We now consider how to, given one extension $\widetilde{\varphi} \in \text{Hom}(G, S_m)$ of $\varphi \in \text{Hom}(M, S_m)$, generate all extensions of φ equivalent to $\widetilde{\varphi}$.

Theorem 7.6.1. *Let $M \leq G$ and $\varphi : M \to S_m$. Suppose that $\widetilde{\varphi} : G \to S_m$ extends φ. Then the class of extensions equivalent to $\widetilde{\varphi}$ can be efficiently enumerated.*

We will see that proving this result reduces to finding coset representatives for subgroups of permutation groups. First, some notation for describing group actions equivalent to $\widetilde{\varphi}$.

Notation 7.6.2. Let $\lambda \in S_m$. Let $\widetilde{\varphi} : G \to S_m$. Define $\widetilde{\varphi}_\lambda : G \to S_m$ by $\widetilde{\varphi}_\lambda(g) = \lambda^{-1}\widetilde{\varphi}(g)\lambda$ for all $g \in G$.

While $\widetilde{\varphi}_\lambda$ will be equivalent to $\widetilde{\varphi}$, regardless of the choice of $\lambda \in S_m$, we remark on the distinction between $\widetilde{\varphi}_\lambda$ being the same group action, an equivalent extension of φ, and an equivalent action.

Remark 7.6.3. Let $\lambda \in S_m$. Let $\psi \in \mathrm{Hom}(G, H)$.

- ψ and $\widetilde{\varphi}$ are equivalent (as a permutation actions) $\iff$ $\psi = \widetilde{\varphi}_\lambda$ for some $\lambda \in S_m$.
- ψ and $\widetilde{\varphi}$ are equivalent extensions of φ $\iff$ $\psi = \widetilde{\varphi}_\lambda$ and $\psi|_M = \widetilde{\varphi}$ $\iff$ $\psi = \widetilde{\varphi}_\lambda$ for some $\lambda \in C_{S_m}(\widetilde{\varphi}(M)) = C_{S_m}(\varphi(M))$.
- ψ and $\widetilde{\varphi}$ are equal $\iff$ $\psi = \widetilde{\varphi}_\lambda$ for some $\lambda \in C_{S_m}(\widetilde{\varphi}(G))$.

We conclude that the sets of coset representatives of $C_{S_m}(\widetilde{\varphi}(G))$ in $C_{S_m}(\varphi(M))$ generate the non-equal equivalent extensions of $\widetilde{\varphi}$.

Remark 7.6.4. Let R be a set of coset representatives of $C_{S_m}(\widetilde{\varphi}(G))$ in $C_{S_m}(\varphi(M))$. The set of equivalent extensions to $\widetilde{\varphi}$ can be described (completely and without repetitions) by

$$\{\widetilde{\varphi}_\lambda : \lambda \in R\}.$$

These centralizers can be found in $\mathrm{poly}(n, m)$-time. The centralizer of a set of T permutations in S_m can be found in $\mathrm{poly}(|T|, m)$ time (see Section 2.4.4), and we use this with the set of generators of M and G. We can now apply the cited unpublished result by Blaha and Luks, stated below and proved in Section 2.4.5.

Theorem 7.6.5 (Blaha–Luks)**.** *Given subgroups $K \leq L \leq S_m$, one can efficiently enumerate a representative of each coset of K in L.*

Since coset representatives of $K = C_{S_m}(\varphi(M))$ in $L = C_{S_m}(\widetilde{\varphi}(G))$ can be efficiently enumerated, so can all equivalent extensions to $\widetilde{\varphi}$, by Remark 7.6.4.

As a corollary, we find that the number of equivalent extensions can be computed in $\mathrm{poly}(n, m)$ time.

Corollary 7.6.6. *Suppose $\widetilde{\varphi} \in \mathrm{Hom}(G, S_m)$ extends $\varphi \in \mathrm{Hom}(M, S_m)$. The number of equivalent extensions to $\widetilde{\varphi}$ is $|C_{S_m}(\widetilde{\varphi}(G)) : C_{S_m}(\varphi(M)|$. This can be computed in $\mathrm{poly}(n, m)$-time.*

7.7 Integer linear programming for large m

There is an interesting phenomenon for very large m, when $m > 2^{1.7^{n^2}}$. The instances OMS_φ of OrMultiSSR can be solved in polynomial time.

MultiSSR can naturally be formulated as an Integer Linear Program, with dimensions $|\mathcal{U}| \times |\mathcal{V}|$, the size of the universe $\mathcal{U}$ and length of the list $\mathfrak{F}$ (indexed by $\mathcal{V}$). The variables correspond to multiplicities of the elements of $\mathfrak{F}$. The constraints correspond to elements of $\mathcal{U}$, by checking whether their multiplicities in the multiset and subset sum are equal.

In OMS_φ, these are $\mathrm{Conj}(M)$ and $\mathrm{Conj}(G)$. A result of Pyber [Pyb93] says that for $G \leq S_n$, the number of of subgroups is bounded by $|\mathrm{Sub}(S_n)| \leq 1.69^{n^2}$. This bound is tight, so we cannot hope for the number of variables ($\mathrm{Conj}(M)$) to be smaller than exponential in n^2.

The "low-dimensional" algorithms of Lenstra and Kannan solve Integer Linear Programming in "polynomial" time [LJ83, Kan87], which are sufficient for this purpose. We state their results more precisely below.

Theorem 7.7.1. *The* Integer Linear Programming*–Search and Decision Problems can be solved in time $N^{O(N)} \cdot s$, where N refers to the number of variables and s refers to*

the length of the input.[12]

Lemma 7.7.2. *Suppose that the* Integer Linear Programming *Search Problem can be solved in time $f(N, M, a)$. Then, the* Integer Linear Programming *Threshold-k Enumeration Problem can be solved in time $f(N, M, a) \cdot O(k^2)$.*

We have found that, for instances φ of HomExtSym with $m > 2^{1.7^{n^2}}$, the Threshold-k Enumeration Problem for OMS_φ can be solved in $\text{poly}(n, m, k)$-time. For these instances of φ, the Threshold-k Enumeration Problem can be solved in $\text{poly}(n, m, k)$-time.

12. This result shows that ILP is fixed-parameter tractable, but we will not use that terminology here.

REFERENCES

[Bab79] László Babai. Monte-Carlo algorithms in graph isomorphism testing. Technical Report DMS 79–10, Université de Montréal, 1979.

[Bab86] László Babai. On the length of subgroup chains in the symmetric group. *Communications in Algebra*, 14(9):1729–1736, 1986.

[Bab89] László Babai. The probability of generating the symmetric group. *Journal of Combinatorial Theory, Series A*, 52(1):148 – 153, 1989.

[Bab91] László Babai. Local expansion of vertex-transitive graphs and random generation in finite groups. In *23rd STOC*, pages 164–174, 1991.

[BBS09] László Babai, Robert Beals, and Ákos Seress. Polynomial-time theory of matrix groups. In *41st STOC*, pages 55–64, 2009.

[BBW18] László Babai, Timothy Black, and Angela Wuu. List-decoding homomorphism codes. 2018. Submitted.

[BGSW18] Timothy Black, Alan Guo, Madhu Sudan, and Angela Wuu. List decoding nilpotent groups. Draft, 2018.

[BL94] Ken Blaha and Eugene M. Luks. P-complete permutation group problems. In *Proc. 25th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, volume 100 of *Congressus Numerantium*, pages 119–124, 1994.

[BL15] Abhishek Bhowmick and Shachar Lovett. The list decoding radius of Reed-Muller codes over small fields. In *47th STOC*, pages 277–285, 2015.

[BLGN+05] Robert Beals, Charles R. Leedham-Green, Alice C. Niemeyer, Cheryl E. Praeger, and Ákos Seress. Constructive recognition of finite alternating and symmetric groups acting as matrix groups on their natural permutation modules. *Journal of Algebra*, 292(1):4 – 46, 2005.

[BS84] László Babai and Endre Szemerédi. On the complexity of matrix group problems I. In *25th FOCS*, pages 229–240, 1984.

[DGKS08] Irit Dinur, Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. Decodability of group homomorphisms beyond the Johnson bound. In *40th STOC*, pages 275–284, 2008.

[DM96] John D. Dixon and Brian Mortimer. *Permutation Groups*. Graduate Texts in Mathematics. Springer New York, 1996.

[FHL80] Merrick Furst, John Hopcroft, and Eugene Luks. Polynomial-time algorithms for permutation groups. In *21st FOCS*, pages 36–41, 1980.

[GKS06] Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. Local decoding and testing for homomorphisms. In *APPROX/RANDOM*, volume 4110 of *LNCS*, pages 375–385. Springer, 2006.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st STOC*, pages 25–32, 1989.

[GS14] Alan Guo and Madhu Sudan. List decoding group homomorphisms between supersolvable groups. In *APPROX/RANDOM*, LIPIcs, pages 737–747. Dagstuhl Publ., 2014.

[Guo15] Alan Guo. Group homomorphisms as error correcting codes. *Electronic Journal of Combinatorics*, 22(1):P1.4, 2015.

[HEO05] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien. *Handbook of computational group theory*. Discrete mathematics and its applications. Chapman & Hall/CRC, Boca Raton, 2005.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.

[KB79] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8(4):499–507, 1979.

[LJ83] Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[Luk] Eugene M. Luks. Private communication.

[Nor12] Christopher Norman. *Matrices with Integer Entries: The Smith Normal Form*, pages 9–46. Springer London, London, 2012.

[Pyb93] László Pyber. Enumerating finite groups of given order. *Annals of Mathematics*, 137(1):203–220, 1993.

[Rob95] Derek J. S. Robinson. *A Course in the Theory of Groups*. Springer, 2nd edition, 1995.

[Ser03] Ákos Seress. *Permutation Group Algorithms*. Cambridge Tracts in Mathematics. Cambridge University Press, 2003.

[Wuu18] Angela Wuu. Homomorphism Extension. 2018. arXiv: 1802.08656.