Detecting Properties of Algorithmically Presented Algebraic and Relational Structures

by Iva Iris Bilanovic

B.S. in Mathematics, May 2013, Bemidji State University
M.A. in Mathematics, May 2017, The George Washington University

A Dissertation submitted to

The Faculty of
Columbian College of Arts and Sciences
of The George Washington University in partial satisfaction
of the requirements for the degree of Doctor of Philosophy

August 31, 2020

Dissertation directed by

Valentina Harizanov
Professor of Mathematics

The Columbian College of Arts and Sciences of The George Washington University certifies that Iva Iris Bilanovic has passed the Final Examination for the degree of Doctor of Philosophy as of April 24, 2020. This is the final and approved form of the dissertation.

Detecting Properties of Algorithmically Presented Algebraic and Relational Structures

Iva Iris Bilanovic

Dissertation Research Committee:

Valentina Harizanov, Professor of Mathematics, Dissertation Director

Jozef Przytycki, Professor of Mathematics, Reader

Alexander Shumakovitch, Assistant Professor of Mathematics, Reader

Jennifer Chubb, Associate Professor of Mathematics, University of San Francisco, Reader

Max Alekseyev, Associate Professor of Mathematics, Examiner

Poorvi Vora, Professor of Computer Science, Outside Examiner

Xiang Wan, Visiting Assistant Professor of Mathematics, Committee Chair

© Copyright 2020 by Iva Iris Bilanovic
All rights reserved

Acknowledgments

To my teachers: Lowell Abrams, who taught me too many things to list. Michael Moses, for all the book loans. Jennifer Chubb, who guided me out of every corner I wrote myself into and taught me to use all available tools. My committee: Jozef Przytycki, Alexander Shumakovitch, Max Alexyeev, Poorvi Vora, and Xiang Wan whose edits made my research stronger and whose support got me through the last days.

To my people: My grad school friends: Hyunjung, Pratima, Debdeep, Jiayuan, Jessa, and Tslil, who made me work harder to keep up with them. The friends who supported me when they had better things to do than grad school: Alex, Jane, and Nienke. All the Hulls, especially Rebecca, for helping me to take needed breaks and taught me about corn mazes. Hannah, who always has my back, even when she gets too many texts from me. Leah, without whom I'd be lost.

To Valentina: For teaching me the skills to stand for myself, and pushing me to when I needed it.

To my parents: For everything.

Thank you.

Abstract of Dissertation

Detecting Properties of Algorithmically Presented Algebraic and Relational Structures

A *Markov property*, P, for a class of groups, $\mathcal{C}$, is any property such that there is a *positive witness*, $G_+ \in \mathcal{C}$, that exhibits the property, and there is a *negative witness*, $G_- \in \mathcal{C}$, such that any group in $\mathcal{C}$ which contains a copy of G_- fails to exhibit property P. We show that detecting a Markov property is Π^0_2-hard in the class of *recursively presented groups*, which are those groups that have a presentation with computable set of generators and recursively enumerable set of relators. Furthermore, detecting a Markov property is Σ^0_1-hard in the class of *computable groups*, those with a computable domain and a computable atomic diagram. These results are an extension of the classic result by Adian and Rabin in the class of finitely presented groups.

We apply these results to determine the exact computability-theoretic complexity of detecting Markov properties of groups, including being abelian and torsion-free. We then find the exact complexity of detecting properties at higher levels of the arithmetical hierarchy, notably the properties of being torsion, nilpotent, and cyclic. Finally, we redefine the notion of a Markov property for classes of *computable relational structures* and follow a similar analysis and application of results as in the class of computable groups.

Contents

List of Figures

Chapter 1

Introduction

1.1 Computability Theory

1.1.1 Decision Problems

A collection of classical objects of study in computability theory, which motivated the early development of the field, are the *decision problems*. A simple definition of a decision problem is any problem that can be phrased as a question with a yes or no answer. A decision problem is *decidable* if there is an algorithm which determines whether the answer is yes or no. For example, the set of primes can be thought of as the problem:

Given a natural number x, is x a prime?

Clearly this is a decision problem, and moreover, a simple algorithm can be described which determines the answer regardless of the particular x; that is, you can always determine (given enough time) if your number is prime.

One can trace the history of computability theory through more complex decision problems. In 1928, David Hilbert and Wilhelm Ackermann posed the Entschiedungsproblem, which asked for an algorithm to determine which statements

of the first-order logic are universally valid and which are not. In 1936, a negative solution was independently found by Alan Turing and Alonzo Chuch; the Entschiedungsproblem is undecidable [47]. Amongst his Millennium Problems, Hilbert also posed what would come to be known as Hilbert's Tenth Problem, which asks for an algorithm that can decide if a Diophantine equation with integer coefficients has solutions in the integers, or not. A negative solution for this question took the better part of a century, and the question remained open until 1970 when Yuri Matiyasevich [38] finished the proof begun by Julia Robinson, Martin Davis, and Hilary Putnam.

1.1.2 Basic Computability

We all have an intuitive notion of being computable or algorithmic. However, to fully understand what is meant by a negative solution to decision problems, that is, what is meant by being *undecidable*, we must define the exact notion of computability. A function (of natural numbers) is *computable* if it can be evaluated on its inputs by a program in some formal system of computability.

When a formalization is needed in the following text, we will always refer to the standard Turing Machines as developed by Alan Turing in the 1930s. However, all formal systems of computability are made up of finite collections of instructions which must capture our intuitive understanding of a computation, and each system has a well-defined notion of an *algorithm*, which is a finite list of these instructions.

All formal models of computability are equivalent to one another, and so it does not matter which is used. Furthermore, we seldom need to call on any formalization, as the intuitive notion of *computability* is also taken to be equivalent. This accepted equivalence is captured by the Church Turing Thesis.

The Church-Turing Thesis.

1. *A function f is computable if and only if f is a total function and it is intuitively computable.*

2. A function f is partial computable if and only if f is intuitively computable.

Considering a particular formalization does allow us to capture some useful properties of algorithms. Firstly, it is clear that there are only countably infinitely many algorithms in any given formalization, and, if we establish some computable procedure for listing all the functions these algorithms yield, we get an *effective enumeration* of all partial computable functions. We fix an algorithmic enumeration procedure for this collections of functions and index them.

Definition 1.1. *There is an effective enumeration of all unary partial computable functions:*

$$\varphi_0(x), \varphi_1(x), \ldots, \varphi_e(x), \ldots$$

Note that this list contains functions which do not necessarily halt on every or, in fact, any inputs and thus some are properly *partial* computable functions. That is, φ_e *halts* (denoted $\varphi_e(x)\downarrow$) on any input x from its domain but runs forever on inputs, z, that are not in its domain ($\varphi_e(z)\uparrow$). As is usual, the domain of the e-th partial computable function is denoted by W_e.

Definition 1.2. *There is an effective enumeration of the domains of all unary partial computable functions:*

$$W_0, W_1, \ldots, W_e, \ldots$$

This enumeration will be referred to repeatedly in the following text, and it will serve as the primary objects that we "work against", or *diagonalize* against, when building non-computable examples. (See [15] and [45].)

Another useful tool through this work, is Kleene's *s-m-n Theorem*.

Theorem 1.1. *If $f(x, y)$ is a partial computable function, then there is a computable function g for which*

$$f(x, y) = \varphi_{g(x)}(y).$$

Rather than using the full technical expression of this theorem, usually we use the intuitive notion Kleene captured: that we can computably get the index of any intuitive algorithm and that if we are given an effective list of algorithms we can compute the corresponding list of indices.

As one might expect, sets and relations are defined to be *computable* when their characteristic functions are computable. A set is said to be *computably enumerable* (or c.e.) if there is an algorithm which enumerates its elements; notice such a set is not necessarily computable. Thus, being computably enumerable is a more inclusive characteristic than being computable, and so it is higher in the hierarchy of non-computable sets.

1.2 Some Model Theory

A predicate *language*, L, is a collection of relation symbols, function symbols, and constant symbols. We constrain ourselves to languages with only finitely many such symbols.

A *L-structure*, is given by $\mathcal{A} = (A, I)$, where A is the *universe*, or *domain*, of L and I is the *interpretation* of L in A. The interpretation I is a function taking the symbols in L to their corresponding relations, functions, and constants in $\mathcal{A}$. For example, if we consider the language $L = \{+, \mathbf{0}\}$, then an example of a L-structure is the additive group of rational numbers, $(\mathbb{Q}, +, 0)$.

Formulas in a language L are strings of the usual logical symbols (parentheses, variables, propositional connectives, quantifiers, equality) and the symbols of L, constrained by the formation rules for terms and formulas. When limited to the first-order logic, as we by and large will be, only finite connectives are admissible, and only finitely-nested first-order quantifiers. The formulas obtained by these rules are referred to as $L_{\omega\omega}$ formulas [13].

A formula in first-order logic with no free variables is called a *sentence*. The set of all sentences true of a structure $\mathcal{A}$ is the *theory*, $Th(\mathcal{A})$, of the structure.

Given $\mathcal{A}$, a structure for $\mathcal{L}$, we expand the language $\mathcal{L}$ to a new language

$$\mathcal{L}_A = \mathcal{L} \cup \{\mathbf{a} : a \in A\}$$

by adding a new constant symbol for each element of A. We then expand $\mathcal{A}$ to a model for $\mathcal{L}_A$, $\mathcal{A}_A$, by interpreting each new constant $\mathbf{a}$ by the element a. The *atomic diagram* of a $\mathcal{A}$, denoted $D(\mathcal{A})$, is the set of all atomic sentences and negations of atomic sentences of $\mathcal{L}_A$ which hold in this expanded structure $\mathcal{A}_A$. A structure $\mathcal{A}$ with a computable domain is said to be *computable* when its atomic diagram $D(\mathcal{A})$ is a computable set.

The *elementary* (or *full*) *diagram* of $\mathcal{A}$, denoted $D^c(\mathcal{A})$, is the set of all first-order sentences of $\mathcal{A}$ true in $\mathcal{A}_A$. A structure with a computable domain is said to be *decidable* if its full diagram is a computable set.

The standard model of arithmetic $\mathcal{N} = (\mathbb{N}, +, \cdot, 0, 1)$ is a computable structure, but it is not decidable. There are non-standard models of arithmetic, i.e. models which have the same theory as $\mathcal{N}$ but are not isomorphic to it. Tennenbaum showed that non-standard models of arithmetic are not computable. A computable theory always has a decidable model. This follows from Henkin's proof of the Gödel completeness theorem [3].

Theorem 1.2 (Gödel)**.** *A relation is arithmetical if and only if it is definable in the standard model of arithmetic.*

So the relations definable in $\mathcal{N}$ are precisely the arithmetical relations, hence the name *arithmetical hierarchy*.

1.2.1 The Arithmetical Hierarchy

Relations built from computable relations using quantifiers can be classified using Kleene's *arithmetical hierarchy*. Every such relation is equivalent to a relation in *prenex normal form*, which is a string of quantifiers followed by a computable relation. A relation in prenex normal form can be classified according to the number of alternating quantifiers and the initial quantifier using the *arithmetical hierarchy*.

Definition 1.3. *[The Arithmetical Hierarchy]*

1. $\Sigma^0_0 = \Pi^0_0 = \Delta^0_1 =$ *all the computable relations*
2. $\Sigma^0_{n+1} =$ *all relations of the form* $(\exists \overline{y})R(\overline{x}, \overline{y})$ *where* $R \in \Pi^0_n$
3. $\Pi^0_{n+1} =$ *all relations of the form* $(\forall \overline{y})R(\overline{x}, \overline{y})$ *where* $R \in \Sigma^0_n$
4. $\Delta^0_{n+1} = \Sigma^0_{n+1} \cap \Pi^0_{n+1}$

We get Σ^0_n sets and Π^0_n sets from this definition, as sets are just one place relations of the form $x \in A$. The base level of the arithmetical hierarchy is composed of the computable sets. The Σ^0_1 sets are exactly the computably enumerable sets.

The arithmetical hierarchy is a *proper* hierarchy of computability in the sense that Σ^0_{n+1} and Π^0_{n+1} strictly include Σ^0_n and Π^0_n sets. There is a corresponding *Turing hierarchy*.

An *oracle Turing machine* is a Turing machine which can ask questions of the form "is $n \in S$?" of a set $S \subseteq \mathbb{N}$ and use the yes or no answer in its computation. The set S is called an *oracle*. A function is *S-computable* if it is computable by a Turing machine with oracle S. For a set B whose characteristic function, χ_B, is S-computable we say B is *Turing reducible* to S, denoted $B \leq_T S$. There is an enumeration procedure for S-computable functions, denoted $\varphi^S_0, \varphi^S_1, \ldots, \varphi^S_y, \ldots$, and the corresponding domains are denoted $W^S_0, W^S_1, \ldots, W^S_y, \ldots$

Definition 1.4. *For a set $S \subseteq \mathbb{N}$, the* jump of S, *denoted S', is $S' = \{x : x \in W_x^S\}$.*

Definition 1.5. *The set $\emptyset^{(n)}$, the n-th* jump *of the empty set, is defined recursively:*

$$\emptyset' = \{x : x \in W_x^{\emptyset}\}$$

$$\emptyset^{(n+1)} = (\emptyset^{(n)})' \text{ for } n \geq 1.$$

The jump of the empty set is the halting set, $K = \{e : \varphi_e(e)\downarrow\}$. Note that the jump of a set S, S', is like the halting set with oracle S, rather than the empty set. The correspondence between the arithmetical sets and the Turing hierarchy was captured by Emil Post.

Theorem 1.3 (Post)**.** *For $A \subseteq \mathbb{N}$ and $n \in \mathbb{N}$, $A \in \Delta^0_{n+1}$ if and only if $A \leq_T \emptyset^{(n)}$.*

Corollary 1.3.1. *$S \subseteq \mathbb{N}$ is Δ^0_2 if and only if $S \leq_T \emptyset'$. If $S \subseteq \mathbb{N}$ is c.e., then $S \leq_T \emptyset'$.*

Furthermore, $\emptyset^{(n)} \leq_T \emptyset^{(n+1)}$ and $\emptyset^{(n+1)} \not\leq_T \emptyset^{(n)}$. So the jumps of the empty set form a strictly increasing (in a Turing reducibility sense) chain of sets.

The Turing degree hierarchy will not be explored here, but an introduction to the topic can be found in [45], [46], and [15].

1.3 Index Sets

1.3.1 The Halting Problem

Despite defining computable and non-computable objects, we have yet to see a concrete example of a non-computable object, other than those in the collection of decision problems. It is by no means trivial that a non-computable object should exist at all, since there are uncountably many sets of natural numbers. The first example of an undecidable problem, and, in a computability theoretic sense, perhaps the most important is the *halting problem*:

Given a natural number e, we ask: does the e-th partial computable function halt on input e?

This problem can be expressed as the relation $\exists s(\varphi_{e,s}(e)\downarrow)$, and a straightforward diagonalization argument shows that it is undecidable.

1.3.2 Many-one reducibility

To precisely determine the complexity of a given property or set in the arithmetical hierarchy we need a means for giving lower-bounds on complexity. Simply finding a Γ level statement of a relation only confirms that the relation is not more complicated than Γ, a simpler statement may exist. m-reducibility is the tool which allows us to find such lower bounds. In the following we conflate defining relations with the sets they define, when convenient and unambiguous.

Definition 1.6 (Post, 1944)**.** *We say a set B is **many-one reducible** (or m-**reducible**) to a set A (denoted $B \leq_m A$) if and only if there is a computable function f such that for all $x \in \mathbb{N}$,*

$$x \in B \iff f(x) \in A.$$

Definition 1.7. *A set A is Σ^0_n-**complete** if $A \in \Sigma^0_n$ and $X \leq_m A$ for every set $X \in \Sigma^0_n$. The notion of Π^0_n-**complete** is similarly defined.*

1.3.3 Index Sets

We will frequently build m-reductions in the following text when we seek to establish the exact complexity of a property or set. To aid us in this, there is a collection of sets with well-known complexity. The most famous of these is the *halting set*, the set counterpart to the halting problem in Section 1.3.1. Since the halting problem is undecidable, K is of a higher complexity than $\Sigma^0_0 = \Pi^0_0 = \Delta^0_1$.

Definition 1.8. *(Index set)*

1. *A* computable index *for a structure $\mathcal{A}$ is a number e such that φ_e is the characteristic function of the atomic diagram of $\mathcal{A}$.*

2. *For a class, $\mathcal{C}$, of structures, the* index set*, denoted $I(\mathcal{C})$, is the set of computable indices for elements of $\mathcal{C}$.*

Definition 1.9. *(Common index sets)*

$$FIN = \{x \mid W_x \text{ is finite}\}$$

$$INF = \{x \mid W_x \text{ is infinite}\}$$

$$TOT = \{x \mid \varphi_x \text{ is total}\} = \{x \mid W_x = \omega\}$$

$$COF = \{x \mid W_x \text{ is cofinite}\}$$

$$REC = \{x \mid W_x \text{ is recursive/computable}\}$$

Theorem 1.4 (Rice's Theorem)**.**

If A is an index set, other than $\emptyset$ or $\mathbb{N}$, then $K \leq_m A$ or $K \leq_m \overline{A}$. Hence, every nontrivial index set is incomputable.

Rice's theorem gives us a lower bound for the complexity of the common index sets: they are all at least Σ^0_1 sets. The following result states the precise complexity of each set, which are obtained through m-reductions.

Theorem 1.5. *(Complexities of common sets)*

1. *The set K is Σ^0_1-complete.*

2. *The set FIN is Σ^0_2-complete.*

3. *The set INF is Π^0_2-complete.*

4. *The set* TOT *is* Π^0_2*-complete.*

5. *The set* COF *is* Σ^0_3*-complete.*

6. *The set* $\overline{COF}$ *is* Π^0_3*-complete.*

7. *The set* REC *is* Σ^0_3*-complete.*

A diagram of the arithmetical hierarchy with the placement of these sets can be found in Figure 1.1.

We also make use of *Ershov's hierarchy*, or the *difference hierarchy*. A set S is *d-c.e.* if it is the difference of two c.e. sets, i.e., if $S = S_1 - S_2$ where S_1 and S_2 are Σ^0_1 sets. We can extend this definition to the *n-c.e. sets* where $S = S_1 - S_2$ is n-c.e. if it is the difference of a Σ^0_1 set S_1 and a $(n-1)$-c.e. set S_2. We can also relativize this definition to get the $d - \Sigma^0_n$ sets where the component sets are Σ^0_n sets rather than Σ^0_1 sets.

1.4 Computable Structure Theory

At times, the first-order logic is insufficient for our purposes. Then we turn to the *computable infinitary formulas*. We write $\bigvee_S$ to denote disjunction and $\bigwedge_S$ to denote conjunction over the (possibly infinite) set S. Computable infinitary formulas allow disjunctions and conjunctions over computable sets S but only finite strings of quantifiers. Unlike for first order logic, there is no prenex normal form, but negation can be brought inside the connectives so we have the following classification.

We provide an abridged, intuitive definition of Σ_α/Π_α formulas. A computable ordinal α is one with the same order type as some computable well-ordering. All such ordinals have a notation in Kleene's $\mathcal{O}$, the set of ordinal notations. The exact definition is given by recursion on computable ordinals and can be found in [3].

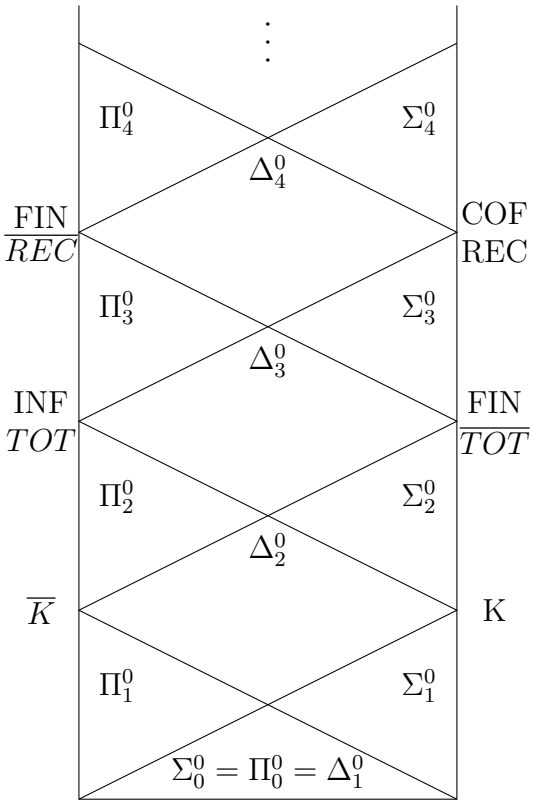

Figure 1.1: The arithmetical hierarchy and complexities of some common sets

Definition 1.10.

1. *The* computable Σ_0 *(which are equivalent to the* computable Π_0 formulas*) are the finitary quantifier-free formulas.*

2. *For a computable ordinal* $\alpha > 0$,

   (a) a computable Σ_α formula $\varphi(\bar{x})$ *has the form*

   $$\bigvee_{i \in I} (\exists \bar{u}_i)\ \psi_i(\bar{x}, \bar{u}_i)$$

   where each ψ_i *is a computable* Π_β *formulas for some* $\beta < \alpha$ *and the disjunction is c.e.,*

   (b) a computable Π_α formula $\varphi(\bar{x})$ *has the form*

   $$\bigwedge_{i \in I} (\forall \bar{u}_i)\ \psi_i(\bar{x}, \bar{u}_i)$$

   where each ψ_i *is a computable* Σ_β *formulas for some* $\beta < \alpha$ *and the conjunction is c.e.*

Theorem 1.6 (Ash, [3])**.** *The relations defined in a countable structure* $\mathcal{A}$ *by a computable infinitary* Σ_α *(or* Π_α*) formula is* Σ^0_α *(or* Π^0_α*, respectively) relative to the atomic diagram of* $\mathcal{A}$*.*

1.5 Decision Problems on Groups

One of the first undecidable problems in mathematics was the *word problem*, which asks: *for a finitely presented group, can we decide whether or not two words in the generators represent the same element.* This problem was posed before the development of computability theory, in 1911 by Max Dehn, along with the *conjugacy*

problem and the *group isomorphism problem*. The word problem for groups was shown to be undecidable independently by Pyotr Novikov in 1955 and William Boone in 1958.

In Chapter 2, we will study decisions problems of *Markov properties* for the class of recursively presented groups, an example of which is the word problem. *Recursively presented groups* are those which are described by a computable set of generators and for which there is an algorithm for enumerating the (possibly infinitely many) relators. We establish that in this class detecting a Markov property is Π^0_1-hard. Then we turn to studying a number of Markov properties at higher levels.

In Chapter 3 we perform a similar analysis on the class of *computable groups*, those with decidable word problem. We conclude that detecting a Markov property is Π^0_1-hard, and then consider a variety of specific group properties at higher levels.

Finally, in Chapter 4, we abstract the main results of the previous chapters, to determine the lower bound for detecting a Markov property in the class of *computable relational structures*. A *relational structure* is a structure with only relation symbols in its signature, that is, no function and constant symbols. We spend the rest of the chapter applying this result to various classes of relational structures.

Chapter 2

The Class of Recursively Presented Groups

2.1 Preliminaries

We begin by considering detection of Markov properties in the class of recursively presented groups, a class which contains both finitely presented and computable groups. While detection problems on finitely presented groups have been well studied (see [40], [14], [33], [31]), recursively presented groups have been less so, despite common groups such as the additive group of rational numbers properly falling into this class. Here we give a deeper treatment to this question.

Definition 2.1. *A group G is said to be* recursively presented *if it is described by a computable set of generators and there is an algorithm for enumerating the (possibly infinitely many) relators. We will denote such a presentation:*

$$\langle x_0, x_1, \ldots \mid R_0, R_1, \ldots \rangle.$$

A remark: literature in computable model theory refers to what we will later call

computable groups as *recursive groups*. This is not what is meant here by recursively presented groups, which are generally not recursive, and not even necessarily isomorphic to computable groups. Only the relator enumerating algorithm of recursively presented groups is computable.

In fact, the set of generators can be made *computable*, not just computably enumerable, but the latter is usually sufficient for our purposes. If a group has presentation

$$\langle x_0, x_1, \ldots \mid R_0, R_1, \ldots \rangle,$$

where the set of generators is computable and we can computably enumerate the relators $R_0, R_1, \ldots$, perform a variant of *Craig's trick*, as described in [13], to find a presentation with a computable set of relators. First, add a generator, t, to the presentation and make it equivalent to the identity by enumerating it into the relators:

$$\langle t, x_0, x_1, \ldots \mid t, R_0, R_1, \ldots \rangle.$$

Then we pad each each R_i with s_i many t's by taking $R_i t^{s_i}$ as a relator, where s_i is the stage at which R_i is enumerated by the computable algorithm for the relators. The presentation

$$\langle t, x_0, x_1, \ldots \mid t, R_0 t^{s_0}, R_1 t^{s_1}, \ldots \rangle$$

gives the same group, since the copies of t will cancel out, leaving the content of S_i, as the relation R_i. Furthermore, it is a presentation with a computable set of relators, not just computably enumerable. Given a relation in the group, it will have form St^r. To decide if it is in the relators, run the original enumeration algorithm for r steps. If S has been enumerated, it is among the relators, if it has not, it will never be among the relators.

Definition 2.2. *(Markov property) A property, P, of groups is* Markov for the class of recursively presented groups *if there is a recursively presented group G_+ so that $G_+ \models P$, and there is a recursively presented group, G_-, so that for any recursively presented group H, if $G_- \hookrightarrow H$ then $H \not\models P$. Note that this also means $G_- \not\models P$.*

We call G_+ a positive witness *for the Markov property, and G_- a* negative witness *for the Markov property.*

Most interesting group properties are either Markov or are the negation of a Markov property. An example of a property which is neither Markov, nor co-Markov is the property of being Hopfian.

Definition 2.3. *Let Γ be a complexity class (for example, in the arithmetical hierarchy), $\mathcal{C}$ the class of recursively presented groups, and A an index set for the collection of recursively presented groups with Markov property P. That is $A \subseteq I(\mathcal{C})$. We say detecting P is:*

1. Γ within the class of recursively presented groups*, if there is a set C in the complexity class Γ such that $A = C \cap I(\mathcal{C})$.*

2. Γ-hard in the class of recursively presented groups*, if for any set S in Γ, there is a computable function $f : \omega \to I(\mathcal{C})$ such that $f(n) \in A$ if and only if $n \in S$.*

3. Γ-complete in the class of recursively presented groups *if P is Γ and Γ-hard in the class of recursively presented groups.*

For example, let A be the set of all indices of computable groups, and P the property "abelian." This property is described by the Π^0_1 formula $\forall x \forall y(xy = yx)$, so the corresponding detection problem is Π^0_1 *in the class of computable groups.* Later (Corollary 3.2.1) we will see that it is Π^0_1-complete.

We use standard computability-theoretic notation throughout. Recall that the eth partial computable function on the natural numbers in some fixed, acceptable

enumeration of Turing machines by φ_e, and its domain by W_e. The set $W_{e,s}$ is the sth finite approximation of W_e, and we may assume throughout that for every s the cardinality $|W_{e,s+1} - W_{e,s}|$ is at most one.

2.1.1 Group Theory

We use standard notation from group theory. The free group on the generators of a group G will be denoted F_G. The symmetric group on a set of n elements will be denoted $Sym(n)$, rather than S_n, which is more typical.

The *free product* of groups A and B is the group whose elements are of the form $a_1b_1a_2b_2 \dots a_rb_r$ where all $a_i \in A$, all $b_i \in B$, and the group operation is concatenation followed by reduction. The free product will be denoted $A * B$, as usual. The *direct product* of groups A and B is the group on the set of the cartesian product of A and B, for which the binary operation is defined component-wise. The direct product will be denoted $A \times B$, as is standard.

When a product of countably infinitely many groups is constructed in the following text it will always the *direct sum* of the groups. That is, all but finitely many entries in each infinite tuple are the component appropriate identity.

A more unusual group product which will be particularly useful to us is the *wreath product*.

Definition 2.4. *(Wreath product) For groups G and H and the left group action ρ of H on a set Ω, the* regular wreath product *of G by H is the semidirect product $G^\Omega \rtimes H$ where G^Ω is the direct product of $|\Omega|$-many copies of G. The* regular wreath product *is denoted $G \wr H$.*

For an introduction to the wreath product, refer to [44].

Here we provide pictures of two wreath products as some background and justification for the the results we use.

Example. *Consider the wreath product $Q \wr D$ where $Q = Sym(2)$ and $D = Sym(3)$.*

Let $\Delta = \{1, 2, 3\}$ viewed as a D-set. The base *of the wreath product is*

$$K = \Pi_{\delta \in \Delta} Sym(2)_\delta \cong (Sym(2))^3 = Sym(2) \times Sym(2) \times Sym(2)$$

with the standard component-wise operations.

Next we must define the action of D on the set K, that is, how elements of $Sym(3)$ permute components of $\Pi_{\delta \in \Delta} Sym(2)_\delta$. We define the action φ by

$$\varphi : dq_\delta = q_{d\delta} \text{ for all } d \in D, q \in Q, \text{ and } \delta \in \Delta,$$

where $d\delta$ is the left action *of d on δ.*

We can also express the action of Q on K by tuples where q_δ is (q, δ) and

$$d : (q, \delta) \mapsto (q, d\delta).$$

For example, if $q_\delta = ((12), 2)$ and $d = (123)$, then $dq_\delta = q_{d\delta} = ((12), 3)$, that is, the element (12) from $Sym(2)$, is moved from the second component of a triple in $(Sym(2))^3$ to the third component.

Finally, the wreath product *is the group*

$$\begin{aligned} Sym(2) \wr Sym(3) &= \Pi_{\delta \in \Delta} Sym(2) \rtimes_\varphi Sym(3) \\ &= \{(u, x) : u \in \Pi_{\delta \in \Delta} Sym(2), x \in Sym(3)\} \end{aligned}$$

with operation

$$(u, x)(v, y) = (yv^{x^{-1}}, xy) = (u(x^{-1}v), xy)$$

in your preferred left action notation.

For example,

$$((1,(12),1),(123))((1,1,(12)),(12)) = ((1,(12),1)(1,(12),1),(123)(12)) = ((1,1,1),(13)).$$

Example. *We can also view the wreath product $Q \wr D$ as a subgroup of $Sym(\Lambda \times \Delta)$ where Λ and Δ are the set being acted on by Q and the set being acted on by D, respectively.*

Again consider $Q = Sym(2)$ and $D = Sym(3)$. Let $\Lambda = \{a, b\}$ and $\Delta = \{1, 2, 3\}$.

Then $Q \wr D \cong W$, where

$$W = \langle D^*, Q^*_\delta : \delta \in \Delta \rangle \leq Sym(\Lambda \times \Delta)$$

the subgroup of $Sym(\Lambda \times \Delta)$ generated by sets of permutations D^* and Q^*_δ.

Define the sets of permutations

$$Q^*_\delta = \{q^*_\delta : q \in Q\}$$

for each $\delta \in \Delta$, where $q^*_\delta : \Lambda \times \Delta \mapsto \Lambda \times \Delta$ is defined by

$$q^*_\delta(\lambda, \delta') = \begin{cases} (q\lambda, \delta') & \text{if } \delta' = \delta \\ (\lambda, \delta') & \text{if } \delta' \neq \delta. \end{cases}$$

That is, a permutation that only works on "matching" second coordinates. In the diagrams below, this can be visualized as an inner permutation.

For example, consider q^*_δ in Q^*_δ for the only non-trivial element of $Sym(2)$, that is $q = (ab)$ and $\delta = 2$. Then

$$q^*_\delta(a, 2) = (qa, 2) = ((ab)a, 2) = (b, 2),$$

but $q^*_\delta(a,3) = (a,3)$ *because* $\delta \neq 3$.

We define $D^* = \{d^* : d \in D\}$, *as a set of permutations* $d^* : \Lambda \times \Delta \mapsto \Lambda \times \Delta$, *where* $d^*(\lambda, \delta) = (\lambda, d\delta)$.

For example, let's consider d^* *in* D^* *for the permutation* $d = (123)$ *and the element* $(\lambda, \delta) = (a, 2)$ *of* $\Lambda \times \Delta$. *Then* $d^*(a,2) = (a,(123)2) = (a,3)$.

The following diagrams give some visualization of this interpretation of the wreath product.

Example. *Symmetric groups.*

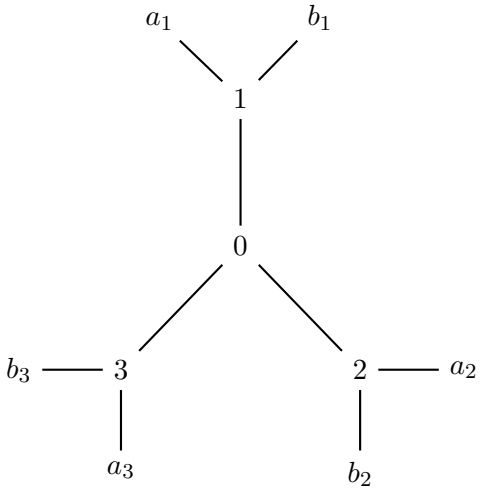

Figure 2.1: Graph Γ with automorphism group $Sym(2) \wr Sym(3)$.

Consider the automorphisms of graph Γ, $Aut(\Gamma)$. *Any* $\varphi \in Aut(\Gamma)$ *must satisfy the following conditions:*

- φ *fixes the vertex* 0,
- *it permutes the "inner ring" of vertices,* $\Delta = \{1,2,3\}$, *and*
- *for each* i, *either*
  - $\varphi(a_i) = a_{\varphi(i)}$ *and* $\varphi(b_i) = b_{\varphi(i)}$, *or*
  - $\varphi(a_i) = b_{\varphi(i)}$ *and* $\varphi(b_i) = a_{\varphi(i)}$.

We can see that the "outer ring" is $\{a_i, b_i : i \in \Delta\}$*, which is in correspondence with* $\Lambda \times \Delta$*, and that* $Sym(3)$ *acts on* $\Delta = \{1, 2, 3\}$ *and* $Sym(2)$ *acts on* $\Lambda = \{a, b\}$*. Thus, it is easy to conclude that this automorphism group is in fact the wreath product* $Sym(2) \wr Sym(3)$*.*

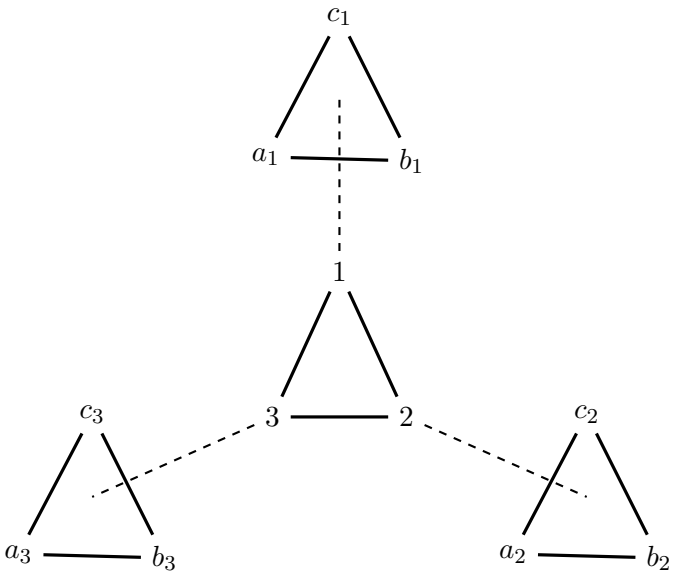

Figure 2.2: Γ with automorphism group $\mathbb{Z}_3 \wr \mathbb{Z}_3$.

Example. *A similar analysis of this diagram as in the previous example, will lead to the conclusion that its automorphism group are* $\mathbb{Z}_3 \wr \mathbb{Z}_3$*. Furthermore, all elements are of order divisible by 3. In fact, the subgroup of such automorphism is exactly all the elements of order* 3^n*.*

It is known that for any prime p the regular wreath product $\mathbb{Z}_p \wr \mathbb{Z}_p$ is isomorphic to the Sylow p-subgroup of $Sym(p^2)$ and has nilpotency class p [25].

We will also periodically reference the *Prüfer groups*, denoted $\mathbb{Z}(p^\infty)$. The Prüfer group is the quotient group $Q/\mathbb{Z}$, where, for a fixed prime p, Q is the additive group of those rational numbers whose denominators are powers of p. This group is also known as the p-quasicylic group. For an exploration of these groups, see [26].

A group is *solvable* if it has a normal series such that each normal factor is Abelian. A useful collection of solvable groups are the *free solvable groups of rank* 2 *and class*

n, denoted $F_2/F_2^{(n)}$, which is the quotient of the free group on 2 generators, F_2, and the n-th derived subgroup of F_2, $F_2^{(n)}$. The *derived subgroup of* F_2 is $F_2^{(1)} = [F_2, F_2]$, and the *n-th derived subgroup of* F_2 is defined recursively by $F_2^{(n)} = [F_2^{(n-1)}, F_2^{(n-1)}]$. This collection of groups of was shown to be uniformly computable by Myasnikov, Romankov, Ushakov, and Vershik in [42].

Some element level notational conventions that we will employ include: denoting the commutator, $x^{-1}y^{-1}xy$, of elements x and y from group G, by $[x, y]$; and using $=_G$ to denote equality in the group G.

It should be noted that in a recursively presented group, equality is not generally a computable predicate, but it is definable by a computable infinitary Σ_1 formula. To see this, first observe that words in F_G which evaluate to the identity in group G can be algorithmically enumerated. So when we write $w =_G v$ for some $w, v \in F_G$, we mean:

$$\bigvee_{s\in\mathbb{N}} (wv^{-1} \in 1_{G,s}),$$

where $1_{G,s}$ is the s-th finite approximation of the set of words in F_G which evaluate to the identity in G.

As a consequence, inequality is a Π_1^0 predicate.

2.2 General Result and Applications

Here we consider the complexity of detecting an arbitrary Markov property in the class of recursively presented groups.

Theorem 2.1. *Let P be a Markov property for recursively presented groups. Then detection of P is Π_2^0-hard in the class of recursively presented groups.*

Proof. We reduce the index set of the infinite c.e. sets, $INF = \{e : |W_e| = \aleph_0\}$, to the detection of P.

Let

$$G_+ = \langle x_0, x_1, \ldots | R_0, R_1, \ldots \rangle = \langle \mathbf{x} \mid \mathbf{R(x)} \rangle$$

and

$$G_- = \langle y_0, y_1, \ldots | S_0, S_1, \ldots \rangle = \langle \mathbf{y} \mid \mathbf{S(y)} \rangle$$

be groups witnessing that P is Markov for recursively presented groups with $G_+ \vDash P$ and $G_- \nvDash P$. For each $e \in \mathbb{N}$, we give a recursive presentation of a group G_e such that G_e has property P if and only if $e \in INF$.

We will need countably many distinct copies of the presentation of G_-, so we write

$$G_-(\mathbf{y_i}) = \langle y_{i,0}, y_{i,1}, \ldots | S_0, S_1, \ldots \rangle = \langle \mathbf{y_i} \mid \mathbf{S(y_i)} \rangle$$

to specify distinct generating sets.

Construction.

Stage 0. Begin by setting

$$G_{e,0} = G_+ * G_-(\mathbf{y_0}) * G_-(\mathbf{y_1}) * \cdots = \langle \mathbf{x}, \mathbf{y_0}, \mathbf{y_1}, \ldots \mid \mathbf{R(x)}, \mathbf{S(y_0)}, \mathbf{S(y_1)}, \ldots \rangle,$$

for every e and $n_e = 0$.

Stage s+1. Let $e \leq s$. We begin this stage with the presentation $G_{e,s} = G_{e,0}$ if $n_e = 0$. Otherwise, by previous stages of the construction, we have

$$G_{e,s} = \langle \mathbf{x}, \mathbf{y_0}, \mathbf{y_1}, \ldots \mid \mathbf{y_0}, \mathbf{y_1}, \ldots, \mathbf{y_{n-1}}, \mathbf{R(x)}, \mathbf{S(y_0)}, \mathbf{S(y_1)}, \ldots \rangle.$$

If $W_{e,s+1} - W_{e,s}$ is empty, set $G_{e,s+1} = G_{e,s}$. Otherwise, set

$$G_{e,s+1} = \langle \mathbf{x}, \mathbf{y_0}, \mathbf{y_1}, \ldots \mid \mathbf{y_0}, \mathbf{y_1}, \ldots, \mathbf{y_{n-1}}, \mathbf{y_n}, \mathbf{R(x)}, \mathbf{S(y_0)}, \mathbf{S(y_1)}, \ldots \rangle,$$

and increment n_e by 1.

End of construction.

Let G_e be the $\lim_{s\to\infty} G_{e,s}$.

Clearly, for every e, G_e is a recursively presented group.

If W_e has finite cardinality n_e, then G_e is the group with presentation

$$G_e = \langle \mathbf{x}, \mathbf{y_0}, \mathbf{y_1}, \ldots \mid \mathbf{y_0}, \mathbf{y_1}, \ldots, \mathbf{y_{n-1}}, \mathbf{R}(\mathbf{x}), \mathbf{S}(\mathbf{y_0}), \mathbf{S}(\mathbf{y_1}), \ldots \rangle,$$

which is a presentation of $G_+ * G_- * G_- * \cdots$, which obviously contains a subgroup G_- and thus does not have property P.

If W_e is infinite, the presentation resulting from the construction is

$$G_e = \langle \mathbf{x}, \mathbf{y_0}, \mathbf{y_1}, \ldots \mid \mathbf{y_0}, \mathbf{y_1}, \ldots, \mathbf{R}(\mathbf{x}), \mathbf{S}(\mathbf{y_0}), \mathbf{S}(\mathbf{y_1}), \ldots \rangle,$$

which is isomorphic to G_+ and so does have property P.

□

We have now established that no Markov property is algorithmically detectable in the class of recursively presented groups. It follows that detection of any Markov property that can be characterized by a finitary Π^0_2 formula, or a by a computable infinitary Π_2 formula, is a Π^0_2-complete decision problem. (See Section 1.4 for the definition of computable infinitary formulas.)

Recall that equality is a Σ^0_1 predicate and hence not computable in the class of recursively presented groups. Also, recall that for group G we write $=_G$, F_G, and 1_G to respectively denote equality in the group, the free group on the generators, and the group identity.

Corollary 2.1.1. *Detecting the following properties is Π^0_2-complete in the class of recursively presented groups.*

1. *Being abelian.*

2. *Being torsion-free.*[1]

3. *Being trivial.*

4. *Being divisible.*

5. *Being a torsion group.*

Proof. The proof for each property requires two components in order to satisfy Theorem 2.1: (i) verification that the property is Markov for the class of recursively presented groups, and (ii) a Π^0_2 defining formula for a group G satisfying the property.

1. Abelian groups are those which satisfy the formula:

$$\forall w \in F_G \; \forall v \in F_G \; (wv =_G vw),$$

   which is Π^0_2 as $=_G$ is a Σ^0_1 predicate. Set $G_+ = \langle x \mid \rangle$ and $G_- = \langle x, y \mid \rangle$ and apply Theorem 2.1.

2. Torsion-free groups are characterized by the formula:

$$\forall w \in F_G \Big(\bigwedge_{n \in \mathbb{N}} (w =_G 1_G \vee w^n \neq_G 1_G) \Big),$$

   which is computable Π_2 (again because $=_G$ is a Σ^0_1 predicate). Set $G_+ = \langle x \mid \rangle$ and $G_- = \langle y \mid y^2 \rangle$ and apply Theorem 2.1.

3. The characterizing formula for triviality is

$$\forall w \in F_G \; (w =_G 1_G),$$

[1]This result was shown by Lempp [31] in 1997. His approach uses more sophisticated combinatorial group theory.

which is Π^0_2. Let $G_+ = \langle x \mid x \rangle$ and $G_- = \langle y \mid \rangle$ and apply Theorem 2.1.

4. A group is divisible if and only if

$$\forall w \in F_G \left(\bigwedge\!\!\!\bigwedge_{n>0} (\exists v \in F_G \ (w =_G 1_G \vee v^n =_G w)) \right),$$

which is computable Π_2. Set $G_+ = \langle x_1, x_2, \ldots \mid x_1^p = 1_{G_+}, x_2^p = x_1, x_3^p = x_2, \ldots \rangle$, the Prüfer group, $\mathbb{Z}(p^\infty)$, for some prime p, and $G_- = \langle x \mid \rangle$ and apply Theorem 2.1.

5. Torsion groups are characterized by the formula:

$$\forall w \in F_G \bigvee_{n \in \mathbb{N}} (w^n =_G 1_G),$$

which is computable infinitary Π_2. Set $G_+ = \langle x \mid x^2 \rangle$ and $G_- = \langle y \mid \rangle$ and apply Theorem 2.1.

□

2.3 Results at Higher Levels

Many interesting Markov properties do not have descriptions at the Π^0_2 level, so the result in Theorem 2.1 is insufficient to precisely determine their complexity. In this section we analyze a selection of group-theoretic properties. While the properties studied are Markov for the recursively presented groups, we seldom use the full abstract definition because we can draw on concrete witnesses.

2.3.1 Finite Groups

We take group *finiteness* to mean the group is a finite group.

Theorem 2.2. *Detecting finiteness is Σ^0_3-complete in the class of recursively presented groups.*

Proof. A characterizing computable infinitary Σ_3 formula is

$$\bigvee_{n\in\mathbb{N}} \exists(w_1,\ldots,w_n) \in [F_G]^n \; \forall v \in F_G \; (v \in_G \{w_1,\ldots,w_n\}),$$

where $\in_G$ stands for the Σ^0_1 formula saying that v is equal, in G, to one w_i.

Recall the definition of the set $COF = \{e \in \mathbb{N} : |\overline{W_e}| < \aleph_0\}$, which is a Σ^0_3-complete set. To show Σ^0_3-completeness of detecting finiteness, we reduce COF to this detection problem. We construct a sequence of groups G_e such that G_e if finite if and only if $e \in COF$.

Construction

Stage $s = 0$. For all $e \in \mathbb{N}$, initialize

$$G_{e,0} = \langle x_0, x_1 \ldots \mid [x_i, x_j], x_i^2, \text{ for all } i,j \in \mathbb{N}\rangle,$$

where $[x,y] = x^{-1}y^{-1}xy$ the *commutator*. That is, $G_{e,0} \cong \mathbb{Z}_2^\omega$.

Stage $s+1$. In this stage we enumerate generators of $G_{e,s}$ into the relators, for all $e \leq s$, based on the enumeration of W_e.

If $W_{e,s-1} = \emptyset$, we begin this stage with $G_{e,s} = G_{e,0}$. If $W_{e,s} = \emptyset$, set $G_{e,s+1} = G_{e,s}$ and proceed to the next stage. Otherwise, if $k \in W_{e,s}$, set

$$G_{e,s+1} = \langle x_0, x_1 \ldots \mid x_k, [x_i, x_j], x_i^2, \text{ for all } i,j \in \mathbb{N}\rangle$$

and proceed to the next stage.

If we have already enumerated elements into $W_{e,s}$ at a previous stage, we begin with

$$G_{e,s} = \langle x_0, x_1 \ldots \mid x_k \text{ for } k \in W_{e,s}; [x_i, x_j], x_i^2, \text{ for all } i,j \in \mathbb{N}\rangle.$$

If $W_{e,s+1} - W_{e,s} = \emptyset$, we set $G_{e,s+1} = G_{e,s}$ and proceed. Otherwise, if $k_{s+1} \in W_{e,s+1} - W_{e,s}$, enumerate $x_{k_{s+1}}$ into the relators and proceed to the next stage with $G_{e,s+1} = \langle x_0, x_1 \ldots \mid x_k \text{ for } k \in W_{e,s+1}; [x_i, x_j], x_i^2, \text{ for all } i, j \in \mathbb{N} \rangle$.

End of construction.

Let $G_e = \lim_{s\to\infty} G_{e,s}$.

Then the resulting groups have presentations

$$G_e = \langle x_0, x_1 \ldots \mid x_k \text{ for } k \in W_e; [x_i, x_j], x_i^2, \text{ for all } i, j \in \mathbb{N} \rangle.$$

Observe, if $\overline{W_e}$ is finite of cardinality n, $G_e \cong \mathbb{Z}_2^n$, a finite group. If W_e is not cofinite, $G_e \cong \mathbb{Z}_2^\omega$ and is infinite. □

2.3.2 Cyclic Groups

Theorem 2.3. *Detecting whether a group is cyclic in the class of recursively presented groups is Σ_3^0-complete.*

Proof. The property of being cyclic is characterized by the computable infinitary Σ_3 formula:

$$\exists w \in F_G \; \forall v \in F_G \bigvee_{n\geq 1} (v =_G 1_G \vee w^n =_G v).$$

For completness we reduce COF to the detection problem as follows. We use the fact that the product of cyclic groups $\mathbb{Z}_n$ and $\mathbb{Z}_m$ is cyclic if and only if n and m are relatively prime. Let p_n be the nth prime number.

Construction.

Stage 0. Begin with

$$G_{e,0} = \langle x_0, x_1, \ldots \mid x_0^{p_0}, x_1^{p_1}, \ldots; [x_i, x_j] \text{ for } i, j \in \mathbb{N} \rangle$$

Stage s+1. If $W_{e,s+1} - W_{e,s}$ is empty, set $G_{e,s+1} = G_{e,s}$.

If $n \in W_{e,s+1} - W_{e,s}$, add x_n to the relators of $G_{e,s}$ and take that as the presentation for $G_{e,s+1}$.

End of construction.

Let $G_e = \lim_{s\to\infty} G_{e,s}$.

If the set W_e is cofinite, all but finitely many of the generators are "killed off" in the construction and we have a recursive presentation for a group that is a finite direct-product of cyclic groups of relatively prime orders. Otherwise, G_e is a recursively presented group that is isomorphic to an infinite direct product of the cyclic groups referenced above, and so is itself not cyclic.

□

2.3.3 Nilpotent and Solvable Groups

We use *nilpotency* to mean that a group has the property of being nilpotent.

Theorem 2.4. *Detecting nilpotency is Σ^0_3-complete in the recursively presented groups.*

Recall that a group G is *nilpotent* if it has a central series of finite length. Consider the lower central series of G:

$$G = G_0 \geq G_1 \geq \ldots \geq G_n = \{1_G\}$$

where $G_0 = G$, and $G_{i+1} = [G_i, G]$ for each $i \leq n$. The finiteness of this series can be expressed as a computable infinitary Σ_3 formula:

$$\bigvee_{n\in\mathbb{N}} \forall \vec{g} \in G^n \ \big([[\ldots[[g_0, g_1], g_2], \ldots], g_n] =_G 1_G\big),$$

where $[x, y]$, as usual, denotes the commutator. The (unique) length of the central series for any group is called the *nilpotency class* of the group.

For completeness, we build a presentation for a group G_e that is nilpotent if and only if W_e is cofinite.

Let $\{H_n\}_{n\in\omega}$ be a sequence of uniformly recursively presented nilpotent groups such that for each n, H_{n+1} has nilpotency class greater than that of H_n. For example, we can take H_n to be $\mathbb{Z}_{p_n} \wr \mathbb{Z}_{p_n}$, the *wreath product* of the group of integers modulo the nth prime by itself.

The full definition of the wreath product can be found in Definition 2.4. It is known that for any prime p the regular wreath product $\mathbb{Z}_p \wr \mathbb{Z}_p$ is isomorphic to the Sylow p-subgroup of $Sym(p^2)$ and has nilpotency class p [25]. Since these groups are finite, they have finite presentations. Thus for each n we can take a finite presentation for H_n, namely

$$H_n = \langle x_{n,1}, \ldots, x_{n,k_n} \mid R_{n,1}, \ldots, R_{n,j_n} \rangle \text{ for some } k_n, j_n \in \mathbb{N}.$$

We use these finite presentations in the proof of theorem 2.4.

Proof of Theorem 2.4. Detecting nilpotency is Σ^0_3 by the computable infinitary Σ_3 defining formula given above. To show completeness we reduce COF to detecting nilpotency.

Construction.

Stage 0. For all $e \in \mathbb{N}$, begin with

$$G_{e,0} = \bigoplus_{n\in\omega} H_n$$

given by the following recursive presentation, for all $j, k, m, n \in \mathbb{N}$,

$$\langle a_{m,k} \text{ for } k \leq k_m \mid R_{m,j} \text{ for } j \leq j_m; [a_{n,k}, a_{m,j}] \text{ for } n \neq m \rangle.$$

Stage s+1. Consider each $e \leq s+1$. When $n \in W_{e,s+1} - W_{e,s}$, enumerate all the generators of H_n into the relators of $G_{e,s}$ and take that as the presentation for $G_{e,s+1}$.

If $W_{e,s+1} - W_{e,s} = \emptyset$, do nothing.

End of construction.

Let $G_e = \lim_{s\to\infty} G_{e,s}$.

The construction yields a recursive presentation of group G_e. If $e \in COF$, G_e is the direct product of finitely many nilpotent groups, and thus itself nilpotent. If $e \notin COF$, we have a group that is residually nilpotent but not nilpotent since it contains subgroups of arbitrarily large nilpotency class.

□

The following corollary follows directly from the proof.

Corollary 2.4.1. *Detecting a nilpotent group is Σ^0_3-complete in the class of recursively presented residually nilpotent groups.*

Corollary 2.4.2. *Determining whether a recursively presented group is finitely presentable is Σ^0_3-complete.*

Proof. Σ^0_3-hardness follows immediately from the proof of Theorem 2.4, since G_e is finitely presentable if and only if $e \in COF$.

For a characterization of being finitely presentable, let

$$G = \langle x_1, x_2, \ldots \mid R_1, R_2, \ldots \rangle.$$

We write $\overline{g}(\overline{x})$ for a finite sequence of words in the generators and their inverses, $\{x_i^{\pm}\}_{i\in\mathbb{N}}$, $w(\overline{g})$ for a word on the elements of $\overline{g}$ and their inverses, and $\overline{w}(\overline{g})$ for a sequence of such words.

Then group G is finitely presentable if and only if the following Σ^0_3 formula holds.

$$(\exists \overline{g}(\overline{x}) \in F_G^{<\omega})(\exists \overline{w}(\overline{g}) \in F_{\overline{g}})(\forall h \in F_G)(\forall u, v \in F_G)(\forall s, t \in \mathbb{N})(\exists s', t' \in \mathbb{N})$$

$$[(h =_G v) \wedge (u \in 1_{G,s} \implies u \in 1_{\overline{g},s'}) \wedge (v \in 1_{\overline{g},t} \implies v \in 1_{G,t'})]$$

The result follows. □

Theorem 2.5. *Detecting a solvable group is Σ^0_3-complete in the class of recursively presented groups.*

Proof. A group G is *solvable* if its derived series is finite. That is, there is an $n \in \mathbb{N}$ such that

$$G = G_0 \geq G_1 \geq \cdots \geq G_n = \{1_G\},$$

where $G_{i+1} = [G_i, G_i]$ for each $i < n$. In the first-order logic, this can be expressed as

$$\bigvee_{n\in\mathbb{N}} \forall \vec{g} \in G^{2^n}\ [[\ldots[[g_1, g_2], [g_3, g_4]], \ldots], [\ldots, [[g_{2^n-3}, g_{2^n-2}], [g_{2^n-1}, g_{2^n}]]\ldots]] =_G 1_G.$$

This is an infinitary Σ_3 formula (as the equality of the iterated commutator is Σ^0_1).

The (unique) length of the derived series for any group is called the *solvability class* of the group.

To show completeness, we follow the construction in Theorem 2.4. We enumerate a presentation of group G_e, which is isomorphic to a direct product of finitely many solvable groups, and thus itself solvable, when $e \in COF$. If $e \notin COF$, G_e will contain subgroups of arbitrarily large solvability class.

For each $n \in \mathbb{N}$, let H_n be the *free solvable group* of rank 2 and solvability class n. That is, H_n will be the quotient of the free group F_2 by its nth derived subgroup. Then H_n is uniformly computable in n [42], and so has a recursive presentation.

As in the construction for Theorem 2.4, for each $e \in \mathbb{N}$ we begin with a presentation of the direct sum

$$G_{e,0} = \bigoplus_{n\in\mathbb{N}} H_n.$$

Whenever $n \in W_{e,s+1} - W_{e,s}$, we enumerate the generators of H_n into the relators of our presentation of G_e.

If $e \in COF$, G_e has solvability class $max(\overline{W_e})$, and is otherwise residually solvable but not solvable.

□

This corollary follows immediately from the proof.

Corollary 2.5.1. *Detecting a solvable group is Σ_3^0-complete in the class of recursively presented residually solvable groups.*

2.3.4 The Word Problem

We have predominantly considered *algebraic* properties of groups. That is, those properties that are concerned with the group operation. Let us turn our attention to more general Markov properties. Here we consider the Dehn's Word Problem and Conjugacy Problem, which are decision problems concerned with equality rather than an operation. (The Isomorphism Problem, the third of Dehn's group theory decision problems, is not always easily described in the first-order logic and since we have not developed the necessary machinery it will not be analyzed here.) The detection of both of these properties falls at the Σ_3^0 level.

Theorem 2.6. *Detecting a group with a decidable word problem is Σ_3^0-complete in the class of recursively presented groups.*

Proof. Let G be a recursively presented group. The property *G has a decidable word problem* is characterized by the computable infinitary Σ_3 formula:

$$\bigvee_{e\in\mathbb{N}} \forall w \in F_G \bigvee_{s\in\mathbb{N}} [\varphi_{e,s}(w)\downarrow \wedge (\varphi_{e,s}(w) = 1 \iff w =_G 1_G)].$$

For completeness, consider

$$G_e = \langle a, b, c, d \mid a^k b a^k =_G c^k d c^k, k \in W_e \rangle.$$

If the predicate $k \in W_e$ is decidable, then G_e has a decidable word problem since the set of relators is a computable set. Similarly, if $k \in W_e$ is not a decidable predicate, then G_e does not have a decidable word problem. Since the predicate $k \in W_e$ being decidable is equivalent to W_e being a computable set we can instead say that the group G_e has a decidable word problem if and only if e is in the Σ_3^0-complete set $REC = \{e \in \mathbb{N} \mid W_e \text{ is computable}\}$.

□

2.3.5 Future Work

Our results on the detecting Markov properties in the class of recursively presented groups are summarized in Figure 2.3.

Theorem 2.7. *Detecting a group with a decidable conjugacy problem is Σ_3^0 in the class of recursively presented groups.*

Proof. The conjugacy problem can be characterized most naturally as follows:

$$\bigvee_{e\in\mathbb{N}} \forall u, v \in F_G \bigvee_{s\in\mathbb{N}} [\varphi_{e,s}(u,v)\downarrow \wedge (\varphi_{e,s}(u,v) = 1 \iff \exists w \in F_G(wuw^{-1} =_G v))].$$

Rewriting in prenex normal form yields an equivalent formula which is clearly Σ_3^0:

$$\bigvee_{e\in\mathbb{N}} \forall u, v \in F_G \bigwedge_{t_2\in\mathbb{N}} \forall w_2 \in F_G \bigvee_{t_1\in\mathbb{N}} \exists w_1 \in F_G \bigvee_{s\in\mathbb{N}} \Big[(\varphi_{e,s}(u,v)\downarrow) \wedge$$

$$(\varphi_{e,s}(u,v) = 1 \implies w_1 u w_1^{-1} =_{G,t_1} v) \wedge (\varphi_{e,s}(u,v) \neq 1 \implies w_2 u w_2^{-1} =_{G,t_2} v)\Big]$$

where we write "$w =_{G,s} v$" for "$wv^{-1} \in 1_{G,s}$."

□

This is only a partial characterization of detecting a decidable conjugacy problem in the class of recursively presented groups. A sharp result would follow from the

following argument. Theorem 2.6 gives us the exact complexity of picking computable groups out of the recursively presented groups, since the property of *having a decidable word problem* is equivalent to *having a computable atomic diagram* and is, in fact, often what is taken as the definition of the latter. The complexity of detecting a group with a decidable conjugacy problem from the class of computable groups is also Σ_3^0 as the complexity of the defining relation does not decrease in the class of computable groups. Thus, if we were to obtain a sharp result in the class of computable groups, it would follow for the class of recursively presented groups. This follows because the intersection of two Σ_3^0-complete sets is Σ_3^0-complete.

Property	**Class of r.p. groups**
Markov property	Π_2^0-hard
Abelian	Π_2^0-complete
torsion-free	Π_2^0-complete
trivial	Π_2^0-complete
divisible	Π_2^0-complete
torsion	Π_2^0-complete
totally left-orderable [5]	Π_2^0-complete
totally bi-orderable [5]	Π_2^0-complete
finite	Σ_3^0-complete
decidable word problem	Σ_3^0-complete
cyclic	Σ_3^0-complete
nilpotent	Σ_3^0-complete
solvable	Σ_3^0-complete

Figure 2.3: Results in the class of recursively presented groups

Chapter 3

The Class of Computable Groups

3.1 Preliminaries

As was discussed in Chapter 2, recursively presented groups are a subset of the class of groups. Specifically, they are the groups that have a presentation with a computable set of generators and a set of relators that can be enumerated by a computable function. In other words, they are c.e. groups. Further tightening the computability restriction of the group descriptions gives a smaller subset: the *computable groups*.

Definition 3.1. *A group is* computable *if its domain is a computable set and the group operation is computable.*

Note that any computable group has a recursive presentation: you simply take the computable domain as the set of generators and the computable atomic diagram as the set of relators. The converse does not always hold.

Theorem 3.1 (Rabin, 1958)**.** *[43] A group has a computable copy if and only if it has a recursive presentation and a decidable word problem.*

Note that this means Theorem 2.6 determines the exact complexity of detecting groups that are computable within the class of recursively presented groups.

We now turn to restructuring our question for the class of computable groups. We once again ask:

> *Given a "nice" description of a group, how hard is it to determine if the group has a property P?*

In this chapter, the nice description of a group will be a computable copy and the types of properties we consider will again be *Markov properties*, but for the class of computable groups.

Definition 3.2. *A property, P, of groups is* Markov for the class of computable groups *if there is a computable group G_+ so that $G_+ \models P$, and there is a computable group G_- so that for any computable group H, if $G_- \hookrightarrow H$ then $H \not\models P$. Note that this implies $G_- \not\models P$.*

We call G_+ a positive witness *for the Markov property and G_- a* negative witness *for the Markov property.*

As before, the hardness of property detection will be captured by the *many-one reducibility* of the *index sets* for groups with the property. In computable structure theory, recursive recognizability of a property amounts to the index set of groups that exhibit that property being a computable set relative to the set of indices of all computable groups.

Definition 3.3. *Let Γ be a complexity class (say in the arithmetical hierarchy), $\mathcal{C}$ the class of computable groups, and A an index set for the collection of computable groups with (Markov) property P. That is, $A \subseteq I(\mathcal{C})$. We say detecting P is*

1. Γ within the class of computable groups*, if there is a set C in the complexity class Γ such that $A = C \cap I(\mathcal{C})$.*

2. Γ-hard in the class of computable groups*, if for any set S in Γ, there is a computable function $f : \omega \to B$ such that $f(n) \in A$ if and only if $n \in S$.*

3. Γ-complete in the class of computable groups *if* P *is* Γ *and* Γ*-hard in the class of computable groups.*

The same group theory which was used in Chapter 2 applies in this context, we will not repeat it, but the reader is referred to Section 2.1.1.

3.2 General Results and Applications

Theorem 3.2. *Let* P *be a Markov property for computable groups. Let* G_+ *be an infinite witness and* G_- *a negative witness that* P *is Markov, as in Definition 3.2. Then detection of* P *is* Π^0_1*-hard in the class of computable groups.*

Proof. Note that we can assume G_- is infinite as it is a subgroup of the direct product of itself with the (computable) additive group of integers, $G_- \times \mathbb{Z}$; this product is necessarily computable and fails to have property P by the definition of a Markov property.

We use the computable atomic diagrams of G_+ and G_- to build, uniformly in e, a computable atomic diagram of a group G_e such that

$$G_e \cong \begin{cases} G_+ & \text{if } \varphi_e(e) \text{ runs forever,} \\ G_+ \times G_- & \text{if } \varphi_e(e) \text{ eventually halts.} \end{cases}$$

If this strategy can be fulfilled, we will have

$$e \in \overline{K} \iff G_e \vDash P.$$

Since $\overline{K}$ is a Π^0_1-complete set, it follows that detecting P is Π^0_1-hard.

Take enumerations

$$G_+ = \langle g_0 = 1_+, g_1, \ldots \rangle$$

and

$$G_- = \langle h_0 = 1_-, h_1, \ldots \rangle$$

without repetitions of the groups witnessing that P is Markov. The universe of G_e will be $\mathbb{N}$. We give a coding map $\langle \cdot \rangle$, and enumerate the atomic diagram in stages below.

Construction.

Stage 0. Let $\langle (g_0, h_0) \rangle = 0$ and add $(0, 0, 0)$ to the atomic diagram, a triple of codes indicating that

$$(g_0, h_0) * (g_0, h_0) = (g_0, h_0)$$

in the group G_e that we are constructing, for all $e \in \mathbb{N}$.

Stage s+1. Consider $e \leq s + 1$. This stage begins with the range of the coding map for $G_{e,s}$ being an initial segment of the natural numbers and a finite set of triples in the atomic diagram of $G_{e,s}$. There are three cases.

1. $\varphi_{e,s+1}(e) \uparrow$. Let i be the least index of an element of G_+ for which (g_i, h_0) has not yet been assigned a code, and assign to it the least available code. Then let j be the least index of an element of G_+ for which there exists a $k \leq j$ such that there is no tuple in the diagram of $G_{e,s}$ indicating the product $(g_j, h_0) * (g_k, h_0)$. For each such $k \leq j$, assign both $(g_j g_k, h_0)$ and $(g_k g_j, h_0)$ codes if necessary, and add the corresponding triples to the diagram.

2. $\varphi_{e,s+1}(e) \downarrow$ but $\varphi_{e,s}(e) \uparrow$. This is the exact stage at which e enters the halting set. After we have executed this stage once, all subsequent stages will be of case (3) type.

   So far we have a partial diagram of a copy of $G_+ \times \{h_0 = 1_-\}$.

   Now we begin to build G_- in the second coordinate. Assign previously unused natural number codes systematically to $\langle (g_i, h_1) \rangle$ for all $i > 0$ for which (g_i, h_0)

has been assigned a code, and add triples to the diagram accordingly.

3. $\varphi_{e,s}(e)\downarrow$. Here e entered the halting set at some previous stage. Let i and j be the least indices for which (g_i, h_0) and (g_0, h_j) have not been assigned codes, and assign them codes. Add all tuples of the form

$$(\langle\langle(g_u, h_v)\rangle\rangle, \langle(g_x, h_y)\rangle, \langle(g_u g_x, h_v h_y)\rangle)$$

for $u, x \leq i$ and $v, y \leq j$, where codes are assigned as needed, to the diagram of G_e.

End of construction.

Let $G_e = \lim_{s\to\infty} G_{e,s}$.

It is clear from the construction that the group is computable and that when $e \in \overline{K}$, $G_e \cong G_+$, and so has property P. If $e \in K$, $G_e \cong G_+ \times G_-$, and will fail to satisfy P. □

Corollary 3.2.1. *Detection of the following properties is Π^0_1-complete in the class of computable groups.*

- *Being abelian.*
- *Being torsion-free.*

Proof. The characterizing formulae of these properties as given in the proof of Corollary 2.1.1 become Π^0_1, since equality (and inequality) is computable in the class of computable groups. We need only verify that the witnesses to each property are computable witnesses. Since both finite groups and free groups have computable copies and free groups are infinite, we can take computable instances of the same witnesses as in Corollary 2.1.1 and apply Theorem 3.2. □

Notice that not all the group-theoretic properties which follow from Theorem 2.1 in Corollary 2.1.1 hold in the class of computable groups. Being trivial is not a Markov property in the class of computable groups; there is no infinite positive witness to being a trivial group. Being divisible and being a torsion group are still Markov properties; however, even without the Σ^0_1 equality predicate, they are properties which can only be described by Π^0_2 defining formulae, so Theorem 3.2 does not apply. We will determine the exact complexity of detecting these properties in the class of computable groups in the next section.

3.3 Results at Higher Levels

3.3.1 Torsion Groups

Theorem 3.3. *The set of indices of torsion groups is Π^0_2-complete in the class of computable groups.*

Proof. The formula:

$$\forall g \in G \bigvee_{n \geq 1} g^n = 1_G$$

is a computable infinitary Π_2 formula characterizing torsion groups.

To show completeness, we reduce the Σ^0_2-complete set FIN of indices of finite sets to the index sets of non-torsion groups. We construct for each $e \in \omega$, a computable abelian group G_e that is non-torsion if and only if W_e is finite. At each stage s, we give a set $G_{e,s} = \{0, x_{\pm 1}, x_{\pm 2}, \ldots, x_{\pm k_s}\}$ of natural numbers indexed by integers as the sth approximation of G_e. In the end, the universe of the group will be $\mathbb{N}$.

The group we build will be isomorphic to a group of the form

$$\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k} \times \cdots$$

if W_e is infinite, or of the form

$$\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k} \times \mathbb{Z}$$

if W_e is finite, where $\mathbb{Z}_m$ denotes $\mathbb{Z}/m\mathbb{Z}$. The values n_i will be determined by the stages that elements appear in the enumeration of W_e. For example, if the m-th element appears at stage s and $(m+1)$-st element appears at stage t, then $n_{m+1} = 2^{(t-s)+1}$.

Each of the x_i's will behave like a tuple of integers under coordinate-wise addition. We will arrange that the inverse of x_i is x_{-i} for each i. To simplify discussion, we will denote the tuple (the behavior of which is) assigned to the natural number x_i by $[x_i]$.

So, for example, if the 17th and 18th elements added to the group are to "behave like" $(0, 1, 2)$ and $(0, -1, -2)$, we would have $[17] = [x_j] = (0, 1, 2)$ and $[18] = [x_{-j}] = (0, -1, -2)$ for some j, and observe that in our group, $17 + 18 = 0$, since we will set $[0] = (0)$.

When we speak of computing sums of tuples of different lengths, we will assume appended padding zeros at the end of the shorter tuple as necessary, i.e., $(2, 3, 4) + (1, 3, 0, 5, 6) = (3, 6, 4, 5, 6)$, modulo n_i in the ith component. The length of a tuple is the length of the sequence up to the last non-zero entry (e.g., the length of $(0, 2, 45, -11, 0, 0, \ldots)$ is 4).

At any given moment in the construction, we will have an element x that has not been assigned a finite order, so has the potential to wind up being a non-torsion element in the end. Whenever a new element enters $W_{e,s}$, we assign a finite order to x by declaring a multiple of it to be the identity in such a way that we do not interfere with any sums previously declared. Any time we add a new element to the group, we will assign it and its inverse names x_i for some $i \in \mathbb{Z}$.

Let $e \in \mathbb{N}$, and assume the approximating sets $W_{e,s}$ have $|W_{e,s+1} - W_{e,s}| \leq 1$.

Construction.

Stage 0: For all e, we will use $x_0 = 0$ as the identity for our groups, and begin the construction with

$$G_{e,0} = \{x_0 = 0, x_1 = 1, x_{-1} = 2\}$$

where $[x_0] = [0] = (0), [x_1] = [1] = (1)$, and $[x_{-1}] = [2] = (-1)$. In what follows, we generally conflate x_i and $[x_i]$, and so speak of tuples "in" $G_{e,s}$.

Stage $s+1$: Consider all $e \leq s$. We begin this stage with

$$G_{e,s} = \{x_0, x_{\pm 1}, x_{\pm 2}, \ldots x_{\pm k_s}\},$$

and each of these is mapped to some finite tuple of integers via the square bracket function. Let n_e be the length of the longest tuples in $G_{e,s}$, and let m_e be the largest positive value of the n_e-th components of elements of $G_{e,s}$. There are two cases:

Case 1: $W_{e,s+1} - W_e = \emptyset$.

In this case, we extend $G_{e,s}$ to $G_{e,s+1}$ by computing the coordinate-wise sums of all pairs of tuples in $G_{e,s}$, and assign fresh x_i's to sums that are not already in $G_{e,s}$ as needed. Note that the value in the n_eth component of the resulting sums will be no more than $2m_e$ and no less than $-2m_e$.

Case 2: $W_{e,s+1} - W_e \neq \emptyset$.

When this is the case, we need to introduce torsion. To do this, we extend $G_{e,s}$ to $G_{e,s+1}$ by adding sums of pairs of tuples in $G_{e,s}$ using modulo $4m_e$ addition in the n_eth component, but "shifted" by $2m_e$ from the usual notation. So, for example, if m_e is 4, then we shall perform additions modulo 16, but shifted by 8 (so 5+7 is -4 rather than 12) to avoid having to change the square bracket function. In the end, we have the values in the n_eth components between $-2m_e$ and $2m_e - 1$ only, and all subsequent additions in this component will be carried out modulo $4m_e$ in the same manner.

In Case 2, we also add two new tuples of length n_e+1, $(0, \ldots, 0, 1)$ and $(0, \ldots, 0, -1)$

to $G_{e,s+1}$.

End of construction.

Let $G_e = \bigcup_s G_{e,s}$.

We finish the proof of the theorem with a sequence of lemmas.

Lemma. *G_e is a computable group.*

Proof. It is clear that G_e is a group. To compute the sum of x_j and x_k, execute the construction to the stage s where both values have been added to $G_{e,s}$. At stage $s+1$, their sum will be defined (and will not be changed later).

□

Lemma. *If $e \in FIN$, then G_e has a non-torsion element.*

Proof. If $e \in FIN$, then there is a stage s so that $W_{e,s} = W_{e,s'}$ for all stages $s' \geq s$. From that stage on, only Case 1 in the construction will be executed, and the result is a group isomorphic to

$$\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k} \times \mathbb{Z}$$

for some $\{n_1, \ldots, n_k\} \subset \mathbb{N}$ where k is the cardinality of W_e. □

Lemma. *If $e \in INF$, then G_e is a torsion group.*

Proof. If $e \in INF$, then there are infinitely many stages s for which $W_{e,s} \neq W_{e,s+1}$ so the construction will execute Case 2 infinitely often.

Let x be a natural number that enters the group at stage s and $[x] = (x_1, x_2, \ldots, x_n)$ be of length n (so $x_n \neq 0$). Note that for each $i < n$, the order of $(0, \ldots, 0, x_i, 0, \ldots, 0)$ is determined by the stages t and t', at which the $(i-1)$-st and i-th elements entered W_e; in particular, the order divides $o_i = 2^{(t'-t)+1}$.

Now let $s' > s$ be the least so that $W_{e,s} \neq W_{e,s'}$. In this stage, Case 2 will be executed, so the element $(0, \ldots, 0, x_n)$ will have finite order that divides $2^{(s'-s'')+1}$, where $s'' < s$ is the largest so that $W_{e,s''} \neq W_{e,s}$.

At the end of stage s', the element x has finite order that divides $o_1 o_2 \cdots o_n$.

□

We have shown that $e \in INF$ if and only if G_e is a torsion group, and the proof is complete.

□

3.3.2 Divisible Groups

Theorem 3.4. *Detecting divisibility is Π^0_2-complete in the class of computable groups.*

Proof. For computable groups, the characterizing formula for divisibility is computable infinitary Π_2:

$$\forall g \in G \bigwedge_{n \geq 1} \exists h \in G (g = 1_G \vee h^n = g).$$

For completeness, we reduce INF, the Π^0_2-complete set of indices of infinite c.e. sets, to the index set of divisible groups. For each e we construct the atomic diagram M_e of a group G_e isomorphic to $(\mathbb{Q}, +)$ if $e \in INF$, and isomorphic to some non-divisible additive subgroup of $\mathbb{Q}$ otherwise.

The domain of the group G_e will be a computable copy of $\mathbb{N}$, here labeled $\{g_0, g_1, \ldots\}$, where each g_i is the name assigned to some rational number $[g_i]$. The atomic diagram M_e will be a set of triples (g_i, g_j, g_k) where g_k is the name assigned to the rational number $[g_i] + [g_j]$. We construct the coding function $[\cdot]$ and the atomic diagram M_e in stages.

Construction.

Stage 0. For all e, assign names g_0, g_1, g_2 to the rational numbers: $[g_0] = 0$, $[g_1] = 1$ and $[g_2] = -1$. Begin the construction of the atomic diagram with all triples (g_i, g_j, g_k)

for which no new names must be defined. That is, $M_{e,0}$ is the set of triples

$$\{(g_0, g_0, g_0), (g_0, g_1, g_1), (g_1, g_0, g_1), (g_0, g_2, g_2), (g_2, g_0, g_2), (g_1, g_2, g_0), (g_2, g_1, g_0)\}.$$

Stage s+1. We begin this stage with

$$dom(M_{e,s}) = \{g_0, g_1, \ldots, g_{n_s}\}$$

and some set of triples $M_{e,s}$, for all $e \leq s$. Regardless of the enumeration of W_e, extend $M_{e,s}$ to $M_{e,s+1}$ by adding the triples (g_i, g_j, g_k) for all g_i, g_j already in the domain of $M_{e,s}$, assigning new names, g_k, as needed.

If $W_{e,s+1} - W_{e,s} = \emptyset$, do nothing.

If $W_{e,s+1} - W_{e,s} \neq \emptyset$, assign the least name g_n which has yet to appear in the construction to the rational $\frac{1}{m+1}$, where $m = |W_{e,s+1}|$. Proceed to the next stage.
End of construction.

Observe, by the construction M_e and $dom(M_e)$ are computable.

If $e \in INF$, the resulting group is a computable copy of $(\mathbb{Q}, +)$, a divisible group. If $e \notin INF$, no element of the group is divisible by any $n > n_e = |W_e|$ and we have a computable copy of the subgroup of $\mathbb{Q}$ generated by $\{1, \frac{1}{2}, \ldots, \frac{1}{n_e+1}\}$. The result follows. □

3.3.3 Nilpotent and Solvable Groups

We use *nilpotency* to mean that a group has the property of being nilpotent.

Theorem 3.5. *Detecting nilpotency is Σ_2^0-complete in the class of computable groups.*

Proof. For computable groups the characterizing formula for nilpotency is computable infinitary Σ_2:

$$\bigvee_{n \geq 2} \forall \overline{g} \in G^n \ [[\ldots[[g_0, g_1], g_2]\ldots], g_n] =_G 1_G,$$

where $[x, y]$ denotes the commutator $x^{-1}y^{-1}xy$.

For completeness, we reduce FIN, the Σ_2^0-complete set of indices of finite c.e. sets, to the index set of the nilpotent computable groups. For each e we construct the atomic diagram, M_e, of a group G_e which is nilpotent if and only if $e \in FIN$.

Let $W(n) = \mathbb{Z}_{p_n} \wr \mathbb{Z}_{p_n}$, where $\wr$ denotes the wreath product and p_n is the n-th prime number. Note that $W(n)$ has nilpotency class p_n and that it is finite. For a more complete description of the wreath products of finite abelian groups see Definition 2.4 and the subsequent examples.

Our construction will yield a computable group G_e that is a direct product of the additive group of integers and $W(n)$'s so that

$$G_e \cong \begin{cases} \mathbb{Z} \times W(1) \times \cdots \times W(n), & \text{if } |W_e| = n, \\ \mathbb{Z} \times W(1) \times \cdots \times W(n) \times \cdots, & \text{if } |W_e| = \omega. \end{cases}$$

If $e \in FIN$, G_e will be nilpotent of class p_n, and residually nilpotent otherwise.

The domain of the group G_e we construct will be $\{g_0, g_1, \ldots\}$, a computable copy of $\mathbb{N}$, and we will approximate its diagram M_e by finite extension. Simultaneously, we build the isomorphism and, denoted by $[g_i]$, the tuple to which it corresponds. For each $n \geq 1$, we write 1_n for the identity in $W(n)$.

Construction.

Stage 0. For all e, set g_0 as the identity of G_e, that is, $[g_0] = (0, 1_1, 1_2, \ldots)$. To begin building a computable copy of the integers in the first component set $[g_1] = (1, 1_1, 1_2, \ldots)$ and $[g_2] = (-1, 1_1, 1_2, \ldots)$. Begin the construction of the atomic diagram with the set of triples for which no new names must be assigned. So $M_{e,0}$ is the set

$$\{(g_0, g_0, g_0), (g_0, g_1, g_1), (g_1, g_0, g_1), (g_0, g_2, g_2), (g_2, g_0, g_2), (g_1, g_2, g_0), (g_2, g_1, g_0)\}.$$

Stage s+1. We begin this stage with

$$dom(M_{e,s}) = \{g_0, g_1, \ldots, g_{n_s}\}$$

and some set of triples $M_{e,s}$. Consider all $e \leq s$.

First, extend $M_{e,s}$ with triples of the form (g_i, g_j, g_k) for all $g_i, g_j \in dom(M_{e,s})$, assigning new names, g_k, as needed.

If $W_{e,s+1} - W_{e,s} = \emptyset$, do nothing.

If $W_{e,s+1} - W_{e,s} \neq \emptyset$, let $n = |W_{e,s+1}|$. Assign fresh names, g_j, to $(0, 1_1, 1_2, \ldots, w, 1_{n+1}, \ldots)$ for each $w \in W(n)$, and add them to the domain. Note that there will be $p_n^{p_n+1} - 1$ such elements.

End of construction.

Let $M_e = \bigcup_s M_{e,s}$.

The group G_e is computable by the construction and if $e \in FIN$,

$$G_e \cong \mathbb{Z} \times W(1) \times \cdots \times W(n)$$

where $n = |W_e|$, and is a group of nilpotency class p_n.

Otherwise, if $e \notin FIN$,

$$G_e \cong \mathbb{Z} \times W(1) \times \cdots \times W(n) \times \cdots,$$

which is a residually nilpotent, but not nilpotent group. □

Theorem 3.6. *Detecting a solvable group is Σ_2^0-complete in the class of computable groups.*

Proof. First note that the description of being a solvable group from the proof of Theorem 2.5 is a computable infinitary Σ_2 formula in the class of computable groups as equality is a computable predicate.

The proof of completeness is essentially identical to the proof of Theorem 3.5 except that rather than using the finite groups $W(n)$ in the construction, we use the uniformly computable free solvable groups $H_n = F_2/F_2^{(n)}$, where F_2 is the free group on two generators, and $F_2^{(n)}$ is the n-th group in its derived series. These groups are infinite, so the construction involves routine dovetailing. □

3.3.4 Cyclic Groups

Theorem 3.7. *Detecting a cyclic group is 3-Σ_2^0-complete in the class of computable groups.*

The following is easy to check.

Lemma. *The 3-Σ_2^0 sets are those of the form $S_1 \cup (S_2 - S_3)$ where all S_i are Σ_2^0.*

Proof of Theorem 3.7. First we show the index set of cyclic groups is 3-Σ_2^0 within the class of computable groups. The usual characterization of a cyclic group (a group generated by a single element) is not optimal in this context. Instead, we recall that a group is cyclic if and only if one of the following things is true:

1. the group is generated by a single element of some finite order, or

2. the group is free and abelian.

The property of being a finite cyclic group can be expressed by the infinitary Σ_2 formula:

$$\bigvee_{n>0} \exists g \in F_G \; \forall h \in F_G \; \exists m \leq n (h =_G 1_g \vee g^m =_G h).$$

An optimal sentence expressing the property of a computable group being free and abelian was given by Carson et. al. in [9] to be d-Σ^0_2.

The disjunction of these two formulas gives us a 3-Σ^0_2 description of being cyclic.

For completeness, take any 3-Σ^0_2 set S, where $S = S_1 \cup (S_2 - S_3)$ and all S_i are Σ^0_2. We will build a sequence of computable groups $\{G_e\}_{e \in \omega}$ so that G_e is cyclic if and only if $e \in S$.

Our construction will result in a computable copy of $(\mathbb{Q}, +)$ if $e \notin S$. Otherwise we build

$$\langle 1, \frac{1}{2}, \ldots, \frac{1}{m} \rangle \leq (\mathbb{Q}, +),$$

which is a cyclic group isomorphic to $(\mathbb{Z}, +)$.

For each natural number e we will construct a map $[\cdot] : \mathbb{Q} \to \mathbb{N}$ assigning names in a computable copy of $\mathbb{N}$ to elements of the additive group of rationals, and a computable atomic diagram M_e, consisting of triples which encode the group operation.

Any triple $([q_i], [q_j], [q_k]) \in M_e$ will correspond to $q_i + q_j = q_k$ holding true for the rationals q_i, q_j, q_k.

For ease of discussion we introduce set D to keep track of which denominators of rationals are introduced at each stage of the construction. Note: D is a set of rationals, *not* of their names in our copy of G_e.

Construction.

Stage 0. Initialize the domain of the group as $G_{e,0} = \{g_0, g_1, g_2\}$ where

$$[0] = g_0, [1] = g_1, \text{ and } [-1] = g_2.$$

Begin the construction of the atomic diagram with the set of triples for which no

new names must be assigned. So $M_{e,0}$ is the set

$$\{(g_0, g_0, g_0), (g_0, g_1, g_1), (g_1, g_0, g_1), (g_0, g_2, g_2), (g_2, g_0, g_2), (g_1, g_2, g_0), (g_2, g_1, g_0)\}.$$

Then set $D = \{1\}$.

Stage s+1. We begin this stage with an initial segment $G_{e,s} = \{g_0, g_1, \ldots, g_m\}$, some finite collection of triples $M_{e,s}$, and the set $D = \{1, d_1, \ldots, d_M\}$.

Regardless of the approximations of the component sets of S, assign names $g_{m+1} = [s+1]$ and $g_{m+2} = [-(s+1)]$. Assign codes to all the triples (g_i, g_j, g_k) where each of $g_i, g_j,$ and g_k are already in $G_{e,s+1}$ and the corresponding equality holds true. Assign these codes in lexicographic order on the indices; for example, if (g_7, g_2, g_{12}), (g_6, g_{12}, g_2), and (g_7, g_2, g_6) first appear in this stage, and the next available code is N we would assign $\langle(g_6, g_{12}, g_2)\rangle = N$, $\langle(g_7, g_2, g_6)\rangle = N + 1$, and $\langle(g_7, g_2, g_{12})\rangle = N + 2$.

For every $d_1, d_2 \in D$, not necessarily distinct, add $d_1 d_2$ to D. (For example, if $D = \{1, 2, 3\}$ we add 4, 6, and 9 to D.) Then there is a collection of $\frac{p}{q} \in \mathbb{Q}$, call it K, so that:

1. $\frac{p}{q}$ has yet to appear in the construction,
2. $s < \frac{p}{q} < s + 1$ or $-(s+1) < \frac{p}{q} < -s$,
3. $gcd(p, q) = 1$, and
4. q is in the set D.

We define an ordering on these fractions so that $\frac{p_1}{q_1} \prec \frac{p_2}{q_2}$ if and only if

1. $q_1 < q_2$, or
2. $q_1 = q_2$, $|p_1| < |p_2|$, or
3. $q_1 = q_2$, $|p_1| = |p_2|$, $p_1 > 0$ and $p_2 < 0$.

We then assign names g_i to all these rationals, assigning the next available name to the least fraction under this ordering, and so on. Finally, before turning to the set S, we define all the triples (g_i, g_j, g_k) so that g_i, g_j, g_k have already been assigned and if $[a_i] = g_i, [a_j] = g_j, [a_k] = g_k$ then $a_i + a_j = a_k$ holds true in the rationals. We assign names to these triples again in lexicographic order on the indices.

Then consider the computable approximations of the component sets of S; a method of approximation for these sets is given in [3]. There are four cases:

1. $e \notin S_{1,s+1}$ and $e \notin S_{2,s+1}$.

2. $e \in S_{1,s+1}$.

3. $e \notin S_{1,s+1}$ and $e \in S_{2,s+1}$ and $e \notin S_{3,s+1}$.

4. $e \notin S_{1,s+1}$ and $e \in S_{2,s+1}$ and $e \in S_{3,s+1}$.

At this point we have a collection of named elements $\{g_0, g_1, \ldots, g_N\}$.

If case *2 or 3*: take $G_{e,s+1}$ to be $G_{e,s}$ union all new names assigned at this stage and $M_{e,s+1}$ to be $M_{e,s}$ union all the new triples. Proceed to the next stage.

If case *1 or 4*: assign names g_{N+i} for $1 \leq i \leq p_l$ for $\frac{i}{p_l} \in \mathbb{Q}$ where p_l is the smallest prime that has yet to appear in the construction. Assign names g_{N+p_l-1+i} for $1 \leq i \leq p_l - 1$ for $\frac{-i}{p_l} \in \mathbb{Q}$. Then for all $d \in D$ and p_l add dp_l to D.

There is now a new collection of rational numbers that satisfy the conditions of set K. Assign names to them in the ordering defined by $\prec$.

Yet again define all triples (g_i, g_j, g_k) that correspond to true sums in the rationals that have been assigned names. Do this in the lexicographic order on the indices.

Let $G_{e,s+1}$ be $G_{e,s}$ union all the new names, and $M_{e,s+1}$ be $M_{e,s}$ union all new triples. Proceed to the next stage.

End of construction.

Take $G_e = \bigcup_s G_{e,s}$ and $M_e = \bigcup_s M_{e,s}$.

If $e \notin S$, eventually only *case 1* or *case 4* will be implemented and $\frac{1}{p}$ for every prime p is added to the generators of group G_e. Thus G_e is isomorphic to the non-cyclic group $\mathbb{Q}$.

If $e \in S$, after some stage M only *case 2* or *case 3* will be implemented. Some collection of fractions $\{1, \ldots, \frac{1}{p_n}\}$ for $n \leq M$ will have been enumerated into the group, but no others will be added. Thus G_e is a copy of $\langle 1, \ldots, \frac{1}{p_n} \rangle$, a cyclic group isomorphic to $\mathbb{Z}$.

□

3.3.5 Future Work

The results from the previous two chapters are summarized in Figure 3.1. It would be remiss to discuss Markov properties and not mention the *conjugacy problem for groups*. At present, the exact complexity of this property is unknown for the class of computable groups, but we have an upper bound.

Theorem 3.8. *Detecting that a group has a decidable conjugacy problem is* Σ^0_3 *in the class of computable groups.*

Proof. See Section 2.7.

□

Property	**Class of r.p. groups**	**Class of computable groups**
Markov property	Π^0_2-hard	Π^0_1-hard (with infinite G_+)
Abelian	Π^0_2-complete	Π^0_1-complete
torsion-free	Π^0_2-complete	Π^0_1-complete
trivial	Π^0_2-complete	n/a
divisible	Π^0_2-complete	Π^0_2-complete
torsion	Π^0_2-complete	Π^0_2-complete
totally left-orderable [5]	Π^0_2-complete	Π^0_1-complete
totally bi-orderable [5]	Π^0_2-complete	Π^0_1-complete
finite	Σ^0_3-complete	n/a
decidable word problem	Σ^0_3-complete	n/a
cyclic	Σ^0_3-complete	3-Σ^0_2-complete
nilpotent	Σ^0_3-complete	Σ^0_2-complete
solvable	Σ^0_3-complete	Σ^0_2-complete

Figure 3.1: Results in classes of groups

Chapter 4

Other Classes of Computable Structures

4.1 Preliminaries

Our main result from the previous chapter relies only on the notion of embedding and finite direct product. Thus any class of structures where these two concepts are well-defined can withstand similar investigation. We extend the notion of a Markov property to the class of *computable relational structures*, extract conditions from our proofs in Chapters 2 and 3 that yield similar general results, and then apply them to a few interesting classes of relational structures.

A *relational structure* is any structure which only has relational symbols (that is, no function symbols and no constant symbols) in its signature. Examples of relational structures include graphs, trees, equivalence relations, partially ordered sets, and so on. We restrict ourselves to studying relational structures with only finitely many relation symbols. A relational structure, $\mathcal{A} = (A, R_1^{\mathcal{A}}, \ldots, R_n^{\mathcal{A}})$ is computable when its domain, A, is a computable set and the interpretation of each relation symbol, R_i, is a computable relation.

Definition 4.1. *For a class, $\mathcal{C}$, of computable relational structures, a property, P, of the structures is* Markov for $\mathcal{C}$ *if there is a $\mathcal{A}_+ \in \mathcal{C}$ which exhibits the property, and there is a $\mathcal{A}_- \in \mathcal{C}$ so that for any $\mathcal{B} \in \mathcal{C}$ if there is an injective homomorphism from $\mathcal{A}_-$ into $\mathcal{B}$, $\mathcal{B}$ fails to have property P.*

We call $\mathcal{A}_+$ a positive witness *for the Markov property and $\mathcal{A}_-$ a* negative witness *for the Markov property.*

We will be tailoring proof techniques from the previous two chapters to fit this context. However, unlike in algebraic structures, relational structures do not necessarily have a well-defined notion of a direct product. The *disjoint union* will replace the functionality of the direct product in much of what follows.

Relational structures are as much a source of computability-theoretic study as algebraic structures. Computable equivalence structures have been studied extensively. Namely, Calvert, Cenzer, Harizanov, and Morozov studied the complexity of isomorphisms between computable equivalence structures in [8]. This was extended to the study of structures within the Ershov difference hierarchy by Khoussainov, Stephan, and Yang in [27], and by Cenzer, LaForte, and Remmel in [10]. Cenzer, Harizanov, and Remmel studied Σ^0_1 and Π^0_1 equivalence structures in [11], and the properties of injection structures in [12]. Marshall extended this study to partial injection structures and nested equivalence classes in [37].

Marshall also studied these properties in classes of trees in [37]. Lempp, McCoy, Miller, and Solomon characterized computably categorical trees of finite height in [32], while Miller showed that no computable well-founded tree of infinite height is computably categorical in [41]. S.S. Goncharov was the first to give examples of graphs with finite computable dimension higher than one in [21]. Csima, Khoussainov, and Liu [16] gave conditions for certain classes of graphs to be computably categorical.

We present here the study of Markov properties for some of these classes of relational structures. This is not the only extension of the concept of a Markov

property beyond the class of groups. Ha showed some preliminary results in this direction when studying Markov properties of computable magmas [22].

4.2 General Results for Relational Structures

Theorem 4.1. *Let P be a Markov property for a class, $\mathcal{C}$, of computable relational structures which is closed under the usual disjoint union for relations. Given an infinite positive witness A_+ and any negative witness A_- to P, detecting P is Π^0_1-hard in the class $\mathcal{C}$.*

Proof. Assume the signature of structures in $\mathcal{C}$ is $\mathcal{L}_{\mathcal{C}} = \{R_0, R_1, \ldots, R_n\}$ where each R_i is an n_i-ary relation symbol (n_i a natural number).

Let A_+ be an infinite positive witness to P in $\mathcal{C}$, and A_- be a (possibly finite) negative witness. Since these are computable structures, there are computable enumerations without repetitions of each of their domains: $|A_+| = \{a_0, a_1, a_2, \ldots\}$ and $|A_-| = \{b_0, b_1, \ldots\}$. We can assume A_- is infinite because it can always be embedded into the disjoint union of itself with a computable copy of A_+.

In the following construction we will use the atomic diagrams of A_+ and A_- to build $A_e \in \mathcal{C}$ so that $A_e \cong A_+$ if $\varphi(e)$ runs forever, and $A_e \cong A_+ \sqcup A_-$ if $\varphi(e)$ eventually halts. That is,

$$A_e \models P \iff e \in \overline{K}.$$

Since $\overline{K} = \{e : \varphi_e(e) \uparrow\}$ is a Π^0_1-complete set, it follows that detecting P is Π^0_1-hard.

The universe of A_e will be $\mathbb{N}$. We give a coding map $\langle\cdot\rangle : A_+ \sqcup A_- \longrightarrow \mathbb{N}$ and enumerate the atomic diagram in stages below. We do not provide the explicit coding of the atomic diagram, since the usual coding scheme is used.

Throughout the construction we will use the following notation: for n-tuple $\overline{x} = (x_1, \ldots, x_n)$ and singleton y, the tuple $((x_1, y), (x_2, y), \ldots, (x_n, y))$ will be denoted

$(\overline{x}, \mathbf{y})$.

Construction.

Stage 0. Let $\langle(a_0, 0)\rangle = 0$ and proceed to the next stage.

Stage s+1. We begin this stage with a domain $A_{e,s}$, which is an initial segment of the natural numbers, namely $\{0, 1, \ldots, n_s\}$, and some finite atomic diagram.

There are three cases.

1. $\varphi_{e,s+1}(e) \uparrow$. Let $\langle(a_{s+1}, 0)\rangle = n_s + 1$.

   Consider all the n-ary relation symbols R_i where $i \leq s + 1$ and $n \leq s$. For each such R_i there may be some n-tuples $\overline{a} \in \{a_0, a_1, a_2, \ldots, a_{s+1}\}^n$, such that $A_+ \models R_i(\overline{a})$ is true and has not yet appeared in the construction. For all these, add $R((\overline{a}, \mathbf{0}))$ to the atomic diagram of A_e, according to the lexicographic order of the indices. For all n-tuples $\overline{a} \in \{a_0, a_1, a_2, \ldots, a_{s+1}\}^n$ such that $A_+ \not\models R_i(\overline{a})$ holds and has yet to appear, add $\neg R_i((\overline{a}, \mathbf{0}))$ to the atomic diagram of A_e, again following the lexicographic order of the indices.

   Proceed to the next stage.

2. $\varphi_{e,s+1}(e) \downarrow$ but $\varphi_{e,s}(e) \uparrow$. This is the exact stage at which e enters the halting set. After we have executed this stage once, all subsequent stages will be of case (3) type.

   In this case we begin by assigning and adding new codes, $\langle(a_{s+1}, 0)\rangle = n_s + 1$ and $\langle(b_i, 1)\rangle = n_s + i + 2$ for $0 \leq i \leq s + 1$, to A_e.

   Repeat the expansion of the atomic diagram exactly as in case (1), to deal with any statements made newly true by the addition of a_{s+1}. Then we mimic case (1) for the b_i, as follows.

   Consider all the n-ary relation symbols R_i where $i \leq s + 1$ and $n \leq s$. For each such R_i there may be some n-tuples $\overline{b} \in \{b_0, b_1, b_2, \ldots, b_{s+1}\}^n$, such that

$A_- \models R_i(\bar{b})$ is true and has not yet appeared in the construction. For all these, add $R((\bar{b}, \mathbf{1}))$ to the atomic diagram of A_e, again respecting the lexicographic order of the indicies. For all n-tuples $\bar{b} \in \{b_0, b_1, b_2, \ldots, b_{s+1}\}^n$ such that $A_- \not\models R_i(\bar{b})$ is true and has yet to appear, add $\neg R_i((\bar{b}, \mathbf{1}))$ to the atomic diagram of A_e, in the lexicographic order of the indices.

Proceed to the next stage.

3. $\varphi_{e,s}(e) \downarrow$ and e entered the halting set at some previous stage.

   Let $\langle (a_{s+1}, 0) \rangle = n_s + 1$ and $\langle (b_{s+1}, 1) \rangle = n_s + 2$. Repeat the addition of newly true sentences for R_i to the atomic diagram of A_e while maintaining the well-defined component-wise direct product as in case (2).

End of construction.

Let $A_e = \cup_s A_{e,s}$.

It is clear from the construction that the structure is computable and that

$$A_e = \begin{cases} A_+ \times \{0\} \cong A_+, & \text{if } e \in \overline{K}, \\ (A_+ \times \{0\}) \cup (A_- \times \{1\}) \cong A_+ \sqcup A_-, & \text{if } e \in K. \end{cases}$$

Thus if $e \notin K$, A_e has property P and if $e \in K$, A_e fails to exhibit property P.

□

Theorem 4.2. *Let P be a Markov property of some class, C, of computable relational structures, which is closed under the usual disjoint union. Suppose there is an infinite positive witness to P, A_+, and a sequence $(A_i)_{i \in \mathbb{N}}$ where for each $i \in \mathbb{N}$: $A_i \in C$ and (i) A_i is a substructure of A_{i+1}, (ii) $A_+ \sqcup A_i \models P$, and (iii) $A_- \not\models P$ where*

$$A_- \cong \bigcup_i A_i.$$

Then detecting P in the class $\mathcal{C}$ is Σ^0_2-hard.

Proof. The strategy here is to use the computable atomic diagrams of A_+ and the A_i's to build, uniformly in e, a computable atomic diagram for B_e, so that $B_e \cong A_+ \sqcup A_n$ if $|W_e| = n$, and $B_e \cong A_+ \sqcup A_-$ if $e \in INF$. Accomplishing this gives us

$$B_e \models P \iff e \in FIN.$$

Since $FIN = \{e : |W_e| < \omega\}$ is a Σ^0_2-complete set, it follows that detecting P is Σ^0_2-hard.

Let $A_+ = \{a_0, a_1, \ldots\}$ and for every $i \in \mathbb{N}$, let $A_i = \{b_0, b_1 \ldots\}$ be the domains of the structures enumerated without repetition of elements. The domains of A_+ and A_i are disjoint, for each i, but as the A_i's are nested substructures, the domain of A_i is a subset of the domain of A_{i+1}. It should be noted, some or even all the A_i's may be finite and while we never explicitly call on the computable atomic diagram of A_-, we will build a copy of it in the following construction.

For n-tuple $\overline{x} = (x_1, \ldots, x_n)$ and singleton y, the tuple $((x_1, y), (x_2, y), \ldots, (x_n, y))$ will be denoted $(\overline{x}, \mathbf{y})$.

The universe of B_e will be a computable copy of $\mathbb{N}$, and we give a coding map $\langle\cdot\rangle$ and enumerate the atomic diagram, a set of triples M_e, in stages below.

Construction.

Stage 0. Add $\langle(a_0, 0)\rangle = 0$ to the domain of $B_{e,0}$. Proceed to the next stage.

Stage s+1. We begin this stage with an initial segment of the natural numbers, $B_{e,s}$, and some finite segment of the atomic diagram, $M_{e,s}$. We have partially enumerated a copy of $A_+ \cup A_{m-1}$ by this stage where $m = |W_{e,s}|$. We begin by extending the copies of A_+ and A_{m-1}.

First, add codes to the domain of $B_{e,s+1}$ for $(a_{s+1}, 0)$ and for $(b_{s+1}, 1)$ that exists in A_{m-1} (recalling that some of the A_i's may be finite), assigning the least available code to a_{s+1} and the next one to b_{s+1}.

Then we extend the atomic diagram of $B_{e,s+1}$. For every n-ary relation R_i where $n < s+1$ and $0 \leq i \leq s+1$, add $R_i(\overline{a}, \overline{0})$ to the atomic diagram of $B_{e,s+1}$ if $R_i^{A_+}(\overline{a}) \models P$ for some $\overline{a} \in \{a_0, \ldots, a_{s+1}\}^n$ which has not already appeared. Do this in lexicographic order on the indices of the $\overline{a}$'s. Similarly add $\neg R_i(\overline{a}, \overline{0})$ to the atomic diagram of $B_{e,s}$, if $\neg R_i^{A_+}(\overline{a}) \models P$ and $\overline{a}$ has not already appeared.

Repeat this procedure for A_{m-1}.

Finally, consider the enumerations of W_e: if no new elements are enumerated into $W_{e,s+1}$, proceed to the next stage.

If $W_{e,s+1} - W_{e,s} \neq \emptyset$, we want to begin attaching a copy of A_m.

To do so, extend the domain of $B_{e,s+1}$ by adding a code for $(b_k, 1)$, for the least $b_k \in B_m - B_{m-1}$.

Then for every n-ary relation R_i, where $n < s+1$ and $0 \leq i \leq s+1$, add $R(\overline{b}, \mathbf{1})$ to the atomic diagram of $B_{e,s+1}$ if $R_i^{A_m}(\overline{b})$ for some $\overline{b} \in \{b_0, \ldots, b_{s+1}, b_k\}^n$ which has not already appeared. Do this in the lexicographic order on the indices of the $\overline{b}$'s. Similarly, add $\neg R_i(\overline{b}, \mathbf{1})$, if $\neg R_i^{A_m}(\overline{b}) \models P$ holds true and it has not already appeared. Do this in the lexicographic order on the indices of $\overline{b}$.

Proceed to the next stage.

End of construction.

Let $B_e = \bigcup_s B_{e,s}$.

It is clear from the construction that the structure is computable, and that when $e \in FIN$, $B_e \cong A_+ \sqcup A_m$, and so has the property P. If $e \notin FIN$, $B_e \cong A_+ \sqcup A_-$, and will fail to exhibit P. □

It should be noted that the results in this section do not apply to all classes of computable relational structures, only to those that share a common signature. However, this is a general enough restriction to allow for application of these results to the classes of computable partial orders, equivalence structures, trees, and graphs.

An analysis of properties in those classes follows.

4.3 Applications

4.3.1 Graphs

Definition 4.2. *A* graph *is a structure $\mathcal{G} = (V, E)$, where $V \subseteq \mathbb{N}$ is a set of vertices, and $E \subseteq V \times V$ is a set of edges, which is symmetric, and so can be viewed as a set of unordered pairs of vertices.*

A graph is *computable* when the set of vertices, V, is a computable set, and there is an algorithm for deciding whether or not $(v_1, v_2) \in E$ for any pair of vertices $v_1, v_2 \in V$. The disjoint union for two graphs is defined in the usual way and it is clear that the class of computable graphs is closed under this union, and even a disjoint union of countably many computable graphs. Thus, Theorems 4.1 and 4.2 apply.

Corollary 4.2.1. *The property of having no cycles is Π^0_1-complete in the class of computable graphs.*

Proof. A graph $\mathcal{G} = (V, E)$ having no edges can be expressed by the infinitary formula:

$$\bigwedge_{n\in\mathbb{N}} \forall v_1, v_2, \ldots, v_n \in V (\exists i \leq n-1)((v_i, v_{i+1}) \notin E).$$

Notice that the existential quantifier in this formula is bounded, and thus we have an infinitary Π_1 formula. A graph having no cycles is Π^0_1. For completeness we only need to give witnesses that this property is Markov for the class of computable graphs and then apply Theorem 4.1

An acceptable positive witness, $\mathcal{G}_+$, that is, an infinite computable graph with no cycles, is an infinite chain graph. So we take $\mathcal{G}_+ = (V_+, E_+)$ where $V_+ = \mathbb{N}$ and

Figure 4.1: Corollary 4.2.1 postive witness, $\mathcal{G}_+$

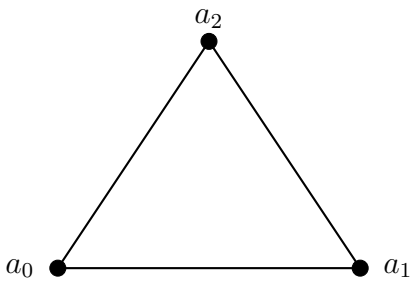

Figure 4.2: Corollary 4.2.1 negative witness, $\mathcal{G}_-$

$E_+ = \{(n, n+1) : n \in \mathbb{N}\}$, as in Figure 4.1. An acceptable negative witness is the (always computable) triangle graph: $\mathcal{G}_- = (V_-, E_-)$ where $V_- = \{a_0, a_1, a_2\}$ and $E_- = \{(a_0, a_1), (a_1, a_2), (a_2, a_0)\}$, as seen in Figure 4.2.

□

Corollary 4.2.2. *Detecting a complete graph is Π^0_1-complete in the class of computable graphs.*

Proof. A graph $\mathcal{G} = (V, E)$ being a complete graph can be expressed by the Π^0_1 formula:

$$\forall v_1, v_2 \in V\big((v_1, v_2) \in E\big).$$

For completeness we need only give witnesses that this property is Markov for the class of computable graphs and apply Theorem 4.1. An acceptable positive witness is the computable complete graph on countably many vertices. That is, $\mathcal{G}_+ = (V_+, E_+)$ where $V_+ = \mathbb{N}$ and $E_+ = \{(a, b) : a, b \in V_+\}$. An acceptable negative witness is the infinite chain graph, $\mathcal{G}_- = (V_-, E_-)$ where $V_- = \mathbb{N}$ and $E_- = \{(n, n+1) : \forall n \in \mathbb{N}\}$.

□

Corollary 4.2.3. *Detecting a connected graph is Π^0_2-complete in the class of com-*

putable graphs.

Proof. A graph $\mathcal{G} = (V, E)$ being connected can be expressed by the infinitary Π_2 formula:

$$\forall v_1, v_2 \in V \bigvee_{n\in\mathbb{N}} \exists v_3, \ldots, v_n \in V \ ((v_1, v_3), (v_3, v_4), \ldots, (v_n, v_2) \in E)$$

To show completeness we must show that the negation of this property, not being a connected graph, has an infinite positive witness, a negative witness, and a sequence of positive witnesses as required by the conditions of Theorem 4.2.

Let $\mathcal{G}_+ = (V_+, E_+)$, where $V_+ = \{a_0, a_1, \ldots\}$, and $E_+ = \{(a_{2i}, a_{2i+1}) : \text{ for } i \in \mathbb{N}\}$.

The sequence of positive witness are: $\mathcal{G}_0 = \mathcal{G}_+$, and $\mathcal{G}_i = (V_i, E_i)$, where $V_i = V_{i-1}$ and $E_i = E_{i-1} \cup \{(2i-1, 2i)\}$. The resulting negative witness is $\mathcal{G}_- = \cup_i \mathcal{G}_i$.

The graphs are shown in Figure 4.3.1.

□

Figure 4.3: Corollary 4.2.3 positive witness, $\mathcal{G}_+ = \mathcal{G}_0$, a disconnected graph

Figure 4.4: Corollary 4.2.3 positive witness, $\mathcal{G}_1$, a disconnected graph

Figure 4.5: Corollary 4.2.3 positive witness, $\mathcal{G}_2$, a disconnected graph

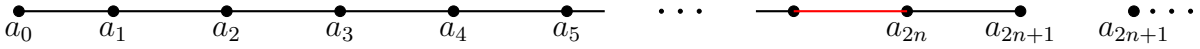

Figure 4.6: Corollary 4.2.3 positive witness, $\mathcal{G}_n$, a disconnected graph

Figure 4.7: Corollary 4.2.3 negative witness, $\mathcal{G}_- = \bigcup_n \mathcal{G}_n$, a connected graph

4.3.2 Nested Equivalence Structures

Definition 4.3. *A set $A \subseteq \mathbb{N}$ and a binary relation E on A is an* equivalence structure *if E is reflexive, symmetric, and transitive.*

Definition 4.4. *For two equivalence relations E and R on a set A, we say E is* nested inside *R if each equivalence class under E sits inside a corresponding equivalence class under R. That is,*

$$[a]_E \subseteq [a]_R \text{ for all } a \in A.$$

We say, R is coarser *and E is* finer.

Definition 4.5. *A* nested equivalence structure *is a tuple $\mathcal{A} = (A, E_1, \ldots, E_n)$ where the domain $A \subseteq \mathbb{N}$ and for each i, E_i is an equivalence relation and $E_{i-1} \subseteq E_i$ for $2 \leq i \leq n$.*

A nested equivalence structure is computable if its domain is a computable set and the atomic diagram of the structure is computable. A useful property of a computable equivalence structure, $\mathcal{A}$ is that we can computably enumerate all of its equivalence classes without repetition.

Corollary 4.2.4. *Detecting the property of having finitely many equivalence classes is Σ^0_2-complete in the class of computable equivalence structures.*

Proof. For an equivalence structure (A, E) having finitely many equivalence classes can be expressed by the infinitary Σ_2 formula:

$$\bigvee_{n\in\mathbb{N}} \exists a_1, \ldots, a_n \in A \; \forall b \in B \Big(b \in [a_1] \vee \cdots \vee b \in [a_n] \Big).$$

To show completeness we call on Theorem 4.2 by giving an infinite computable witness, $\mathcal{A}_+$, to the property and a sequence of nested equivalence classes, $\{\mathcal{A}_i\}_i$, in which the property holds but so that $\mathcal{A}_- \cong \bigcup_i \mathcal{A}_i$ is a negative witness to the

property. For all the witnesses E will be interpreted as the equivalence relation on pairs:

$$(a, b) \sim (c, d) \iff b = d.$$

With this relation, there are computable copies of the equivalence structures $\mathcal{A}_+ = (A_+, E)$ and $\mathcal{A}_i = (A_i, E)$ on the domains:

$$A_+ = \{(n, 0) : n \in \mathbb{N}\}$$

$$A_i = A_{i-1} \cup \{(n, i) : n \in \mathbb{N}\} \text{ for all } i \in \mathbb{N}_{\geq 1}.$$

Note that $\mathcal{A}_+$ has one equivalence class, $\mathcal{A}_i$ has i equivalence classes, but $\mathcal{A}_- \cong \bigcup_i \mathcal{A}_i$, which has a computable copy, has countably many equivalence classes.

□

The property of a nested equivalence structure $(A, E_1, \ldots, E_n)$ having finitely many equivalence classes for each equivalence relation is still expressible by an infinitary Σ_2 formulas, as it only differs from the non-nested version by a finite conjunction:

$$\bigwedge_{i=1}^{n} \bigvee_{k \in \mathbb{N}} \exists a_1, \ldots, a_k \in A \; \forall b \in B (b \in [a_1] \vee \cdots \vee b \in [a_k]).$$

Similarly, property of a nested equivalence structure $(A, E_1, \ldots, E_n)$ having finitely many equivalence classes for any equivalence relation is expressible by an infinitary Σ_2 formulas, as it only differs from the non-nested version by a finite disjunction:

$$\bigvee_{i=1}^{n} \bigvee_{k \in \mathbb{N}} \exists a_1, \ldots, a_k \in A \; \forall b \in B (b \in [a_1] \vee \cdots \vee b \in [a_k]).$$

Thus the following two corollaries hold, with the same (mutatis mutandis) proof as Corollary 4.2.4.

Corollary 4.2.5. *Detecting the property of having finitely many equivalence classes for every equivalence relation is Σ^0_2-complete in the class of computable nested equivalence structures with a fixed number of relations.*

Corollary 4.2.6. *Detecting the property of having finitely many equivalence classes for some equivalence relation is Σ^0_2-complete in the class of computable nested equivalence structures.*

The negation of each of the properties also holds.

Corollary 4.2.7. *Detecting the property of having infinitely many equivalence classes for some equivalence relation, is Π^0_2-complete in the class of computable nested equivalence structures with a fixed number of relations.*

Detecting the property of having infinitely many equivalence classes for every equivalence relation, is Π^0_2-complete in the class of computable nested equivalence structures with a fixed number of relations.

4.3.3 Future Work

A natural extension of our work in the class of relational structures is to more varied classes. In particular, it would be interesting to study partial injection structures as Cenzer, Harizanov, Remmel, and Marshall have in [12] and [37]. Our Theorems 4.1 and 4.2 do apply to both injection structures and partial injection structures, but finding properties that are Markov and of Π^0_1 or Σ^0_3 complexity will be worthwhile.

Another future project is extending the results of Chapter 2 into the class of relational structures. That is, we can ask:

> If we are using a description of a relational structure that is computably enumerable (rather than computable) what is the complexity of detecting a Markov property?

Bibliography

[1] S.I. Adian, *The unsolvability of certain algorithmic problems in the theory of groups*. Trudy Moskovskogo Mathematickogo Obshchestva, **6**, (1957) 231–298.

[2] S.I. Adian, *Finitely presented groups and algorithms*. Doklady Akademii Nauk SSSR, **117**, (1957) 9–12.

[3] C. Ash, J.F. Knight, *Computable Structures and the Hyperarithmetical Hierarchy*, Elsevier Science, Amsterdam, 2000.

[4] G. Baumslag, *Wreath products and finitely presented groups*. Math. Zeitschr. **75**, (1961) 22–28.

[5] I. Bilanovic, J. Chubb, and S. Roven, *Detecting properties from descriptions of groups*. Archive for Math. Logic, **59(3)**, (2020) 293–312.

[6] W.W. Boone and H. Rogers Jr., *On a problem of J.H.C. Whitehead and a problem of Alonzo Church*. Mathematica Scandinavica **19**, (1966) 185-192.

[7] W. Calvert, V. S. Harizanov, J. F. Knight, and S. Miller, *Index sets of computable structures*. Algebra and Logic **45**, (2006) 306–325.

[8] W. Calvert, D. Cenzer, V. Harizanov, and A. Morozov, *Effective categoricity of equivalence structures*. Annals of Pure and Applied Logic **141**, (2006) 61–78.

[9] J. Carson, V. Harizanov, J. Knight, K. Lange, C. McCoy, A. Morozov, S. Quinn, C. Safranski, and J. Wallbaum, *Describing free groups*. Transactions of the American Mathematical Society **365**, (2012) 5715–5728.

[10] D. Cenzer, G. LaForte, and J. Remmel, *Equivalence structures and isomorphisms in the difference hierarchy*. Journal of Symbolic Logic **74**, (2009) 535–556.

[11] D. Cenzer, V. Harizanov, and J. Remmel, Σ^0_1 *and* Π^0_1 *equivalence structures*. Annals of Pure and Applied Logic **162**, (2011) 490–503.

[12] D. Cenzer, V. Harizanov, and J. Remmel, *Computability-theoretic properties of injection structures*. Algebra and Logic **53**, (2014) 60–108 (Russian); 39–69 (English translation).

[13] C.C. Chang and H. Jerome Keisler, *Model Theory* Third Edition, Dover, New York, 2012.

[14] D.J. Collins, *On recognizing properties of groups which have solvable word problem*. Arch. Math. **21**, (1970) 31–39.

[15] S. Barry Cooper, *Computability Theory*, CRC Press, Boca Raton, 2004.

[16] B. Csima, B. Khoussainov, and J. Liu, *Computable categoricity of graphs with finite components*. Lecture Notes in Computer Science **5028**, (2008) 139–148.

[17] M. Dehn, *Über unendliche diskontinuierliche Gruppen*. Mathematische Annalen **71**, (1911) 116–144.

[18] M. Dehn, *Transformation der Kurven auf zweiseitigen Flächen*. Mathematische Annalen **72**, (1912) 413–421.

[19] L. Fuchs, *Note on ordered groups and rings*. Fund. Math. **46**, (1958) 167–174.

[20] L. Fuchs, *Partially Ordered Algebraic Systems*, Pergamon Press, 1963.

[21] S. S. Goncharov. *The problem of the number of nonautoequivalent constructivizations*. Algebra i Logika, **19 (6)**, (1980) 621–639, 745.

[22] T. Ha, *On Algorithmic Properties of Computable Magmas*, Ph.D. Thesis, The George Washington University, Washington, D.C., 2018.

[23] M. Harrison-Trainor and M.-C. Ho, *On optimal Scott sentences of finitely generated algebraic structures*. Proceedings of the American Mathematical Society **246**, (2018) 4473-4485.

[24] M. Ho, *Describing Groups*. Proceedings of the American Mathematical Society **145**, (2017) 2223–2239.

[25] L. Kaloujnine, *La structure des p-groupes de Sylow des groupes symetriques finis*. Annales Scientifiques de l'Ecole Normale Superiure **65 (3)**, (1948) 239–276.

[26] I. Kaplansky, *Infinite Abelian Groups*, University of Michigan Press, 1969.

[27] B. Khoussainov, F. Stephan, and Y. Yang, *Computable categoricity and the Ershov hierarchy*. Annals of Pure and Applied Logic **156**, (2008) 86–95.

[28] V.M. Kopytov and N.Ya. Medvedev, *Right-Ordered Groups*, Siberian School of Algebra and Logic, Consultants Bureau, New York, 1996.

[29] J.F. Knight and C. McCoy, *Index sets and Scott sentences*. Archive for Math. Logic **53**, (2014) 519–524.

[30] J.F. Knight and V. Saraph, *Scott sentences for certain groups*. Archive Math. Logic **57**, (2018) 453–472.

[31] S. Lempp, *The computation complexity of torsion-freeness of finitely presented groups*. Bull. Austr. Math. Soc. **56**, (1997) 273–277.

[32] S. Lempp, C. McCoy, R. Miller, and R. Solomon, *Computable categoricity of trees of finite height*. Journal of Symbolic Logic **70**, (2005) 151–215.

[33] J. Lockhart, *Decision problems in classes of group presentations with uniformly solvable word problem*. Arch. Math. **37**, (1981) 1–6.

[34] R.C. Lyndon and P.E. Schupp, *Combinatorial Group Theory*, Springer, Berlin, 2001.

[35] W. Magnus, *Das Identitäts problem für Gruppen mit einer definierenden Relation*. Math. Ann. **106**, (1932) 295–307.

[36] W. Magnus, A. Karrass, and D. Solitar, *Combinatorial Group Theory*, Interscience Publishers, New York, (1966).

[37] L. Marshall, *Computability-Theoretic Properties of Partial Injections, Trees, and Nested Equivalences*, Ph.D. Thesis, The George Washington University, Washington, D.C., 2015.

[38] Y. Matiyasevich, *Hilbert's Tenth Problem*, MIT Press, Cambridge, London, 1993.

[39] C. McCoy and J. Wallbaum, *Describing free groups, part II:* Π^0_4 *hardness and no* Σ^0_2 *basis*. Transactions of the American Mathematical Society **365**, (2012) 5729–5734.

[40] C.F. Miller, III, Decision problems for groups — survey and reflections, in *Algorithms and Classification in Combinatorial Group Theory* (G. Baumslag and C.F. Miller, III, eds.), MSRI Publications No. 23, Springer-Verlag, New York, 1992, 1–59.

[41] R. Miller, *The computable dimension of trees of infinite height*. Journal of Symbolic Logic **70**, (2005) 111–141.

[42] A. Myasnikov, V. Romankov, A. Ushakov, and A. Vershik, *The word and geodesic problems in free solvable groups*. Transactions of the American Mathematical Society **362**, (2010) 4655–4682.

[43] M.O. Rabin, *Recursive unsolvability of group theoretic problems*. Annals of Math. **67**, (1958) 172–194.

[44] J.J. Rotman, *An Introduction to the Theory of Groups*, Springer-Verlag, Berlin, 1991.

[45] R.I. Soare, *Recursively Enumerable Sets and Degrees*, Springer-Verlag, Berlin, 1987.

[46] R.I. Soare, *Turing Computability (Theory and Applications)*, Springer-Verlag, Berlin, 2016.

[47] A.M. Turing, *On computable numbers, with an application to the Entscheidungsproblem*. Proceedings of the London Mathematical Society **42**, (1936) 230–265.