Weisfeiler–Leman and Group Isomorphism

By **Nathaniel A. Collins**

Department of Mathematics

University of Colorado Boulder

United States

04/11/2023

Committee members

Thesis Advisor, Outside Reader: Joshua A. Grochow, Department of Computer Science

Honors Council Representative: Nathaniel Thiem, Department of Mathematics

Additional Committee Member: Peter Mayr, Department of Mathematics

Acknowledgements

There are many people I wish to thank for their help and support. I firstly want to thank Michael Levet, who took every opportunity to go above and beyond in supporting me throughout my entire undergraduate career. In addition to teaching math, he helped enable every research opportunity I had during my undergraduate and read my writing many more times than he needed to.

I additionally want to thank Joshua Grochow and Ryan Layer for their mentorship and for enabling me to try research before I found out I wanted to do this long-term.

I want to thank my friends and colleagues who were a constant source of encouragement. I finally want to thank Laura, who was with me through every up and down, encouraged me when I needed encouragement, and made sure I didn't procrastinate my writing, and my cat, who attacked my toes whenever I got distracted while writing.

Abstract

We investigate the power of the counting and count-free variants of the Weisfeiler–Leman (WL) Version I algorithm for groups (Brachter & Schweitzer, LICS 2020).

- We study the counting and count-free versions of the Weisfeiler–Leman algorithm when applied to so-called *CFI* groups, which arise from *CFI* graphs (Cai, Fürer, & Immerman, *Combinatorica*, 1992) via Mekler's construction (*J. Symb. Log.*, 1981). We use $O(\log \log n)$ rounds of WL Version I, improving upon the work of Brachter & Schweitzer, who used $O(\log n)$ rounds of WL Version II. As a consequence, we obtain improvements in both the parallel and descriptive complexities of identifying these groups.
- We further improve the parallel complexity using count-free WL Version I, bounded non-determinism, and limited counting. In particular, we obtain a $\beta_1\mathsf{MAC}^0(FOLL)$ isomorphism test for CFI groups

Contents

1 Introduction

1.1 Background and Motivation

Informally, an isomorphism problem takes as input two objects and asks if they are the same for some appropriately defined notion of equivalence. Efficient algorithmic solutions to isomorphism problems have applications in a number of areas such as chemistry [26], cryptography [17, 54], error-correcting codes [24, 13], computer algebra systems [50, 22], classifying quantum entanglement via known connections to tensors [37], and other areas in complexity theory [33].

Key motivation for the work in this thesis arises from the Graph Isomorphism problem (GI), which takes as input two graphs and asks if there is an isomorphism $\varphi : V(G) \to V(H)$. The computational complexity of Graph Isomorphism remains an intriguing open question. Indeed, in their seminal papers introducing the notions of NP and NP-completeness, both Cook [27] and Levin [78] asked if Graph Isomorphism was NP-complete.

In 1975, Ladner proved that if $\mathsf{P} \neq \mathsf{NP}$, then there exist problems in NP which are neither in P nor NP-complete. More precisely, there exists a strict infinite hierarchy of problems contained in NP [59]. Finding such a problem would imply that $\mathsf{P} \neq \mathsf{NP}$. There has been considerable effort in identifying NP-intermediate candidates. Many of these candidates, such as Linear Programming and Primality Testing, have been placed into P. The remaining candidates under (historical) consideration include isomorphism problems such as GI, as well as cryptographic problems such as Integer Factorization and Discrete Logarithm. There is a precise sense in which Integer Factorization and Discrete Logarithm might in fact be easier than GI. These problems have an upper bound of $\mathsf{NP} \cap \mathsf{coNP}$ while Graph Isomorphism has upper bounds of $\mathsf{NP} \cap \mathsf{coAM}$. To date, the best algorithmic upper bound is $n^{\Theta(\log^2 n)}$ [8] (Babai's analysis only provided a quasipolynomial bound. See [47] for a more careful analysis, which yields the exponent of $\Theta(\log^2 n)$).

There is considerable evidence that GI is not NP-complete. For instance, as GI $\in \mathsf{coAM}$, we have that if GI were NP-complete, then $\mathsf{PH} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}} = \mathsf{AM}$ (see for instance, [2]). Additionally, since the best upper bound is $n^{\Theta(\log^2 n)}$ [8], if this problem were to be NP-complete, then all of NP can be solved in quasipolynomial time. This would violate the Exponential Time Hypothesis [52] and imply that $\mathsf{EXP} = \mathsf{NEXP}$ [20]. It is believed that $\mathsf{EXP} \neq \mathsf{NEXP}$. In addition, GI is low for both PP [58] and SPP [3] and belongs in the second level $\mathsf{L}_2^{\mathsf{P}}$ of the Low Hierarchy [75] which is contained in NP. Hence, unless PH collapses to some level, GI is not reducible to any NP-complete problem under multiple notions of reducibility weaker than polynomial-time reductions. Finally, the decision variant of GI is polynomial-time equivalent to the counting variant #GI. No NP-complete problem is known to be polynomial-time equivalent to its counting version.

At the moment, further advances in resolving the complexity of Graph Isomorphism appear out of reach at the moment. It is thus natural to ask about special cases of GI that might be easier than the general case. One such problem is the Group Isomorphism (GpI) problem. GpI takes as input two finite groups and decides whether an isomorphism exists between them. When the two groups are given by their Cayley (multplication) tables, GpI is known to be reducible to Graph Isomorphism. Hence, GpI is also in $\mathsf{NP} \cap \mathsf{coAM}$. In particular, when the groups are given by their multiplication tables, GpI is AC^0-reducible to GI, and there is no AC^0-reduction from GI to GpI [23]. When the groups are given succinctly, such as by generating sets of permutations or matrices, GI reduces to GpI [23, 66, 37].

Similarly, the current best upper bound on GpI is $n^{\Theta(\log n)}$. It can be shown that a group of order n has a minimal generating set of size $\leq \lceil \log_p n \rceil$, where p is the smallest prime dividing n. Tarjan [69] and Lipton, Synder, & Zalcstein [65] independently observed that GpI admits a brute-force generator enumeration algorithm. This algorithm works by deriving the minimal generating set for both groups, giving each generator some index, and then testing all ways these generators can be matched together. Tarjan [69] gave an $n^{\log_p(n)+O(1)}$-time algorithm, and Lipton, Snyder & Zalcstein [65] gave a stronger bound of $\mathsf{DSPACE}(\log^2 n)$. This bound has escaped largely unscathed for 40 years, however, Rosenbaum improved this bound to $n^{(1/4)\log_p(n)+O(1)}$ [73] (see [32, Section 2.2]). Thus, in light of the $n^{\Theta(\log^2 n)}$-runtime bound on GI, GpI serves as a key barrier to placing GI into P. There has been significant work in the last 15 years developing efficient algorithms for special cases of GpI [36, 60, 9, 71, 80, 62, 12, 10, 72, 73, 19, 35]. These algorithms primarily leverage algebraic techniques. As GpI is strictly easier than GI under AC^0 reductions, it is natural to inquire as to whether combinatorial techniques from GI can be fruitfully leveraged in the setting of groups. We investigate this question in more detail using the Weisfeiler-Leman algorithm.

The k-dimensional Weisfeiler–Leman (WL) algorithm serves as the central combinatorial technique in GI. We refer readers to Section 2 of the original CFI [21] paper for a more detailed history of this algorithm. Published by Boris Weisfeiler and Andrei (Andrey) Leman [81, 82], and later generalized to its k-dimensional variant independently by Babai & Mathon [6] and Immerman & Lander [51], this algorithm colors k-tuples of vertices of both graphs in an isomorphism-invariant manner. This algorithm has two components; an initial coloring step that colors each k-tuple and an iterated refinement step that assigns every k-tuple a new color based on the colors of nearby tuples. See Section 2.7 for a more detailed description of how the algorithm works.

For fixed k, the k-dimensional Weisfeiler–Leman algorithm runs in polynomial-time [81, 82]. WL serves as a polynomial time nonisomorphism test for several families of graphs. These include trees [30, 51], graphs of bounded treewidth [42, 45, 55, 61], graphs of bounded rank width [43], planar graphs [41, 56], graphs of bounded genus [38, 40], interval graphs [31], and graphs with a forbidden minor [39]. In addition, just 1-WL is powerful enough to identify almost all graphs [74].

Despite the power and success of the Weisfeiler–Leman algorithm, it is not sufficiently powerful to place GI into P. We will discuss this here. Already, the simplest of these algorithms, the 1-dimensional WL algorithm, fails to identify regular graphs– for instance, 1-WL fails to distinguish C_6 and $2K_3$. More generally, two graphs are not distinguished by 1-WL if and only if they are fractionally isomorphic [77]. The 2-dimensional Weisfeiler-

Leman algorithm identifies nearly all regular graphs [14, 57], but fails to distinguish strongly regular graphs since every pair of vertices has the same number of shared neighbors [7]. More generally, Cai, Fürer, & Immerman [21] exhibited an infinite family of graph pairs $(G_n,\ H_n)$ that require $\Theta(\log n)$-dimensional WL to distinguish G_n from H_n. As a result, WL fails to place GI into P. Even the individualize-and-refine paradigm fails to resolve GI in polynomial-time [70]. On the other hand, the CFI graphs have bounded degree and so admit a polynomial-time isomorphism test using the group theoretic techniques of Babai & Luks [67, 11, 5, 44]. As a consequence, group theory appears to be a necessary ingredient to place GI into P

In light of the fact that GpI is strictly easier than GI under AC^0 reductions, it is natural to ask whether Weisfeiler-Leman will be more effective in the setting of groups. Previous works [63, 18] have attempted to use WL as a subroutine for GpI by reducing to a graph based on a group action over a vector space. Subsequently, Brachter & Schweitzer [15] formulated three variants of the Weisfeiler–Leman algorithm in the setting of groups. They showed that these three variants are equivalent in distinguishing power, up to a factor of 2 in the dimension. As a demonstration of the power of their model, Brachter & Schweitzer [15] showed that WL identifies the so-called *CFI groups*– class 2 p-groups of exponent $p > 2$ arising from the CFI graphs [21] via Mekler's construction [68]. As further evidence for the power of their model, Brachter & Schweitzer also showed that WL can detect many common isomorphism invariants [16], and Grochow & Levet show that WL for groups serves as a powerful parallel algorithm for determining isomorphism for several families of groups [34].

1.2 Summary of results

In this thesis, we use the Weisfeiler–Leman Version I algorithm for groups [15] to improve the parallel and descriptive complexities of identifying the CFI groups. This is joint work with Michael Levet [25].

We first establish the following.

Theorem 1.1. *Let Γ_0 be a 3-regular connected graph, and let $\Gamma_1 :=$ CFI(Γ_0) and $\Gamma_2 :=$*

$\widetilde{CFI}(\Gamma_0)$ *be the corresponding CFI graphs. For* $i = 1, 2$*, denote* $G_i := G_{\Gamma_i}$ *be the corresponding groups arising from Mekler's construction. We have that the* $(3, O(\log \log n))$*-WL Version I distinguishes* G_1 *from* G_2*. If furthermore* Γ_0 *is identified by the graph* $(3, r)$*-WL algorithm, then the* $(3, \max\{r, O(\log \log n)\})$*-WL Version I algorithm identifies* G_1 *as well as* G_2*.*

This improves the previous result of Brachter & Schweitzer, who established the analogous result using $O(\log n)$ rounds of the more powerful WL Version II algorithm. Our improved upper bound relied on two techniques. First, we observed that Brachter & Schweitzer [15] crucially leveraged the graph structure supporting the CFI groups. In particular, they did not rely on the full generated subgroup of a given 3-tuple. Thus, we observed that we could instead use the Weisfeiler–Leman Version I algorithm, which has a weaker initial coloring. Second, we demonstrate an improved pebbling strategy in the associated Ehrenfeucht–Fraïssé game which requires $O(\log \log n)$ rounds instead of $O(\log n)$. Hence, with an improvement to both the initial coloring and iterated refinement step, we have an improved upper bound on the complexity for 3-WL to distinguish CFI groups.

By the parallel implementation of WL by Grohe & Verbitsky [45] and the logical characterization of WL [21, 51, 15], we obtain improvements in both the parallel and descriptive complexities for identifying the CFI groups.

Corollary 1.2. *Let* Γ_0 *be a 3-regular connected graph, and let* $\Gamma_1 := CFI(\Gamma_0)$ *and* $\Gamma_2 := \widetilde{CFI}(\Gamma_0)$ *be the corresponding CFI graphs. For* $i = 1, 2$*, denote* $G_i := G_{\Gamma_i}$ *be the corresponding groups arising from Mekler's construction.* G_1 *can be distinguished from* G_2 *in using a logspace uniform* TC *circuit of depth* $O(\log \log n)$*. Furthermore, if the base graph* Γ_0 *is identifiable by 3-WL for graphs in* r *rounds, then we can decide isomorphism between* G_i $(i = 1, 2)$ *and an arbitrary group* H *using a logspace (uniform)* TC *circuit of depth* $\max\{r, O(\log \log n)\}$.

Corollary 1.3. *Let* $\mathcal{C}^I_{k,r}$ *be the Version I fragment of first order logic with counting quantifiers, where formulas are permitted at most* k *variables and quantifier depth at most* r*. Then* *there exists a formula φ in $\mathcal{C}^I_{3,O(\log\log n)}$ such that $G_1 \models \varphi$ and $G_2 \not\models \varphi$. Furthermore, if the base graph Γ_0 is identifiable by 3-WL for graphs in r rounds, then for any group $H \not\cong G$, there is a formula in φ_i in $\mathcal{C}^I_{3,O(\log\log n)}$ such that $G_i \models \varphi_i$ and $H \not\models \varphi_i$ for $i \in 1, 2$.*

Additionally, we utilize the weaker *count-free* variant of WL Version I in tandem with bounded non-determinism and limited counting to further improve the parallel complexity for isomorphism testing of the CFI groups. This technique was previously introduced by Grochow & Levet [34] in the setting of Abelian groups. We show:

Theorem 1.4. *Let Γ_0 be a 3-regular connected graph, and let $\Gamma_1 := \text{CFI}(\Gamma_0)$ and $\Gamma_2 := \widetilde{\text{CFI}(\Gamma_0)}$ be the corresponding CFI graphs. For $i = 1, 2$, denote $G_i := G_{\Gamma_i}$ to be the corresponding groups arising from Mekler's construction.*

(a) The multiset of colors computed by the count-free $(O(1), O(\log\log n))$-WL Version I distinguishes G_1 from G_2. In particular, we can decide whether $G_1 \cong G_2$ in $\beta_1\mathsf{MAC}^0(\mathsf{FOLL})$.

(b) If furthermore Γ_0 is identified by the graph count-free $(3, r)$-WL algorithm, then the multiset of colors computed by the count-free $(O(1), \max\{r, O(\log\log n)\})$-WL Version I will distinguish G_i $(i = 1, 2)$ from an arbitrary graph H. In particular, we can decide whether $G_i \cong H$ in $\beta_1\mathsf{MAC}^0(\mathsf{FOLL})$.

2 Background

We begin by recalling preliminary notions from graph theory and group theory.

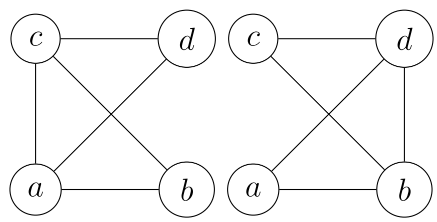
2.1 Graphs

A simple, undirected graph $G = (V, E)$ consists of a set V of vertices and a set E of edges $\{u, v\}$ for vertices $u, v \in V$. In this thesis, we restrict our attention to finite, simple, undirected graphs. For a vertex $v \in V(G)$, the *neighborhood* of a vertex v is $N(v) := \{u \in V(G) : \{u, v\} \in E(G)\}$. The *degree* of a vertex is $\deg(v) := |N(v)|$. Additionally, a graph is k-regular if each vertex has degree k. For two graphs G, H, we say that G and H are isomorphic, denoted $G \cong H$ if there exists a bijection $\varphi : V(G) \to V(H)$ such that

$$\{u, v\} \in E(G) \iff \{\varphi(u), \varphi(v)\} \in E(H)$$

For example, consider the two graphs below:

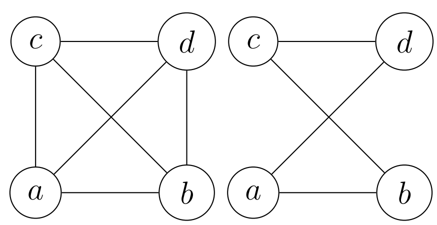
Example 2.1. *Let G and H be the graphs below.*



Observe that the bijection $a \mapsto b, b \mapsto a, c \mapsto d, d \mapsto c$ is an isomorphism of the two graphs.

However, not all pairs of graphs have such an isomorphism.

Example 2.2. *Let G and H be the graphs below.*



In this example, there is no isomorphism $\varphi : V(G) \to V(H)$ since G has eight edges while H only has six.

2.2 Groups

We assume familiarity with group theory at the level of an undergraduate Abstract Algebra course. We refer readers to a standard text for more information (see e.g., [28]).

Groups are considered by their Cayley (multiplication) tables. For a group of order n, its Cayley table has n^2 entries where each entry is represented by a binary string of size $\lceil \log_2 n \rceil$. A p-group is a group of order p^k for some prime p and positive integer $k \geq 1$. We say that a p-group has nilpotency class 2 if $G/Z(G)$ is abelian. In this thesis, we will be particularly interested in class 2 p-groups with exponent $p > 2$.

Remark 2.1. Class 2 p-groups of exponent p for $p > 2$ are considered to be the hard cases for Group Isomorphism. When p-groups are given by generating sets of matrices over $\mathbb{F}_p$, isomorphism for p-groups of class $c > 2, c < p$ and exponent p reduces to an isomorphism test for p-group of class 2 and exponent p. Additionally, every finite group has a maximal solvable normal subgroup denoted $\text{Rad}(G)$). $G/\text{Rad}(G)$ contains no abelian normal subgroups and an efficient isomorphism test is known to exist for these groups [10]. This indicates that solvable groups are the bottleneck case for GpI. In a different direction, Dietrich & Wilson exhibited a nearly-linear time isomorphism test for groups of almost all orders. Their algorithm notably did not handle groups of large prime power order, which suggests that groups whose orders are not large prime powers are not hard instances of GpI.

Definition 2.2. Let G be a group. For $a, b \in G$, define the *commutator* $[a, b] := aba^{-1}b^{-1}$.The *commutator subgroup* of G is the subgroup $G' = [G, G] = \langle\{[a, b] \mid a, b \in G\}\rangle$.

Definition 2.3. Let G be a group. The *Frattini subgroup* $\Phi(G)$ is the intersection of all maximal subgroups of G.

Theorem 2.4 (Burnside, see, e.g., [28])**.** *Let G be a finite p-group. We have that $\Phi(G) = G^p[G, G]$.*

Remark 2.5. In the case of class-2 p-groups, we have that $[G, G] \leq Z(G)$. If furthermore, G has exponent p, then $\Phi(G) = [G, G] \leq Z(G)$.

2.3 Mekler's Construction

We recall Mekler's construction [68] (recently improved by He & Qiao [46]), which allows us to encode a graph into a class 2 p-group ($p > 2$) of exponent p.

Definition 2.6. For $n \in \mathbb{N}$ and a prime $p > 2$, the relatively free class 2 p-group of exponent p is given by the presentation

$$F_{n,p} = \langle x_1, \ldots, x_n | R(p, n)\rangle,$$

where $R(p, n)$ consists of the following relations:

- For all $i \in [n]$, $x_i^p = 1$, and
- For all $i, j, k \in [n]$, $[[x_i, x_j], x_k] = 1$.

Thus, $F_{n,p}$ is generated by $x_1, \ldots, x_n$, each of these generators has order p, and the commutator of any two generators commutes with every generator (and thus, every group element). It follows that each element of $F_{n,p}$ can be written uniquely in the following normal form:

$$x_1^{d_1} \cdots x_n^{d_n} [x_1, x_2]^{d_{1,2}} \cdots [x_1, x_n]^{d_{1,n}} [x_2, x_3]^{d_{2,3}} \cdots [x_{n-1}, x_n]^{d_{n-1,n}}.$$

Here the exponents take on values in $\{0, \ldots, p-1\}$. In particular, $|F_{n,p}| = p^{n+n(n-1)/2}$.

Mekler's construction [68, 46] allows us to encode a graph as a class 2 p-group of exponent p as follows.

Definition 2.7 (Mekler's Construction)**.** Let $\Gamma(V, E)$ be a simple, undirected graph with $V = \{v_1, \ldots, v_n\}$, and let $p > 2$ be prime. We construct a class 2 p-group of exponent p as follows:

$$G_\Gamma = \langle x_1, \ldots, x_n \,\big|\, R(p, n), [x_i, x_j] = 1 : v_i v_j \in E \rangle.$$

So two generators of G_Γ commute precisely if the corresponding vertices form an edge of Γ. We identify x_i with the vertex v_i.

Remark 2.8. Mekler's construction provides a many-to-one reduction from GI to GpI (or in the language of category theory, an isomorphism-preserving functor from the category of graphs to the category of groups). In his original construction, Mekler first reduced arbitrary graphs to "nice" graphs via the use of gadgets. He & Qiao [46] subsequently showed that this gadgetry was unnecessary. In both Mekler's original construction [68] and the improvement due to He & Qiao [46], this reduction is polynomial-time computable when the inputs for the groups are given in the generator-relation model (c.f., [68, 46] and [15, Theorem 4.13]). While we will be given such groups by their Cayley (multiplication) tables, we are still able to reason about the groups using the underlying graph-theoretic structure.

We now recall some key properties about the groups arising via Mekler's construction.

Lemma 2.9 ([15, Lemma 4.3])**.** *We have that* $\Phi(G_\Gamma) = G'_\Gamma$*, and the vertices of* Γ *form a minimum-cardinality generating set of* G_Γ*.*

Lemma 2.10 ([15, Corollary 4.5])**.** *Let* Γ *be a simple, undirected graph. Then we have that* $|G_\Gamma| = p^{|V(\Gamma)|+|\binom{V}{2})-E(\Gamma)|}$*. In particular, every element of* G_Γ *can be written in the form:*

$$v_1^{d_1} \cdots v_n^{d_n} c_1^{d_{n+1}} \cdots c_k^{d_{n+k}},$$

where $\{c_1, \ldots, c_k\}$ *is the set of non-trivial commutators between generators (i.e., the non-edges of* Γ*) and each* d_i *is uniquely determined modulo* p*.*

Lemma 2.11 ([15, Lemma 4.7])**.** *Let* Γ *be a simple, undirected graph. We have* $Z(G_\Gamma) = G'_\Gamma \times \langle v : N[v] = V(\Gamma)\rangle$*. In particular, if no vertex of* Γ *is adjacent to every other vertex, then* $Z(G_\Gamma) = G'_\Gamma$*.*

Definition 2.12 ([15, Definition 4.8])**.** Let $x \in G_\Gamma$ be an element with normal form:

$$x := v_1^{d_1} \cdots v_n^{d_n} c_1^{e_1} \cdots c_m^{e_m}.$$

The *support* of x is $\{v_i : d_i \not\equiv 0 \pmod{p}\}$. For a subset $S = \{v_{i_1}, \ldots, v_{i_s}\} \subseteq V(\Gamma)$, let x_S be the subword $v_{i_1}^{d_{i_1}} \cdots v_{i_s}^{d_{i_s}}$, with $i_1 < \cdots < i_s$.

Lemma 2.13 ([15, Corollary 4.11])**.** *Let* Γ *be a simple, undirected graph. Let* $x := v_{i_1}^{d_1} \cdots v_{i_r}^{d_r} c$*, with* $i_1 < i_2 < \cdots < i_r$*,* c *central in* G_Γ*, and each* $d_i \not\equiv 0 \pmod{p}$*. Let* $C_1, \ldots, C_s$ *be the connected components of the complement graph* $co(\Gamma[\text{supp}(x)])$*. Then:*

$$C_{G_\Gamma}(x) = \langle x_{C_1}\rangle \cdots \langle x_{C_s}\rangle \langle\{v_m : [v_m, v_{i_j}] = 1 \textit{ for all } j\}\rangle G'_\Gamma.$$

2.4 Complexity

We first recall definitions for some standard circuit complexity classes. We assume familiarity with Turing machines, asymptotic complexity, and complexity classes P and NP.

We touch briefly on circuits as a computational model. For more information, we refer readers to some of the standard resources on circuit complexity [2, 1, 79]

We are interested in circuits consisting of only **AND**, **OR**, and **NOT** gates. In this thesis, we will consider polynomial-sized circuits of polylogarithmic depth. We will introduce several families of circuits we get various types of circuits. NC circuits are defined as those with fan-in 2. AC circuits generalize NC circuits, in that the **AND** and **OR** gates have unbounded fan-in. Finally, TC circuits in turn generalize AC circuits by allowing for the use of **MAJORITY** gates– which return true if $> \frac{n}{2}$ inputs are true.

We additionally want to understand the power of these circuits as computational models. This is done by defining a complexity class for languages computable by a circuit of each type.

Definition 2.14. For a fixed natural number k, A language L belongs to NC^k if there exists a (uniform) family of NC circuits $(C_n)_{n\in\mathbb{N}}$ such that C_n has depth $O(\log^k n)$ and polynomial-size, and for all strings x, we have that $x \in L$ if and only if $C_{|x|}(x) = 1$.

Definition 2.15. For a fixed natural number k, A language L belongs to AC^k if there exists a (uniform) family of AC circuits $(C_n)_{n\in\mathbb{N}}$ such that C_n has depth $O(\log^k n)$ and polynomial-size, and for all strings x, we have that $x \in L$ if and only if $C_{|x|}(x) = 1$.

Definition 2.16. For a fixed natural number k, A language L belongs to TC^k if there exists a (uniform) family of TC circuits $(C_n)_{n\in\mathbb{N}}$ such that C_n has depth $O(\log^k n)$ and polynomial-size, and for all strings x, we have that $x \in L$ if and only if $C_{|x|}(x) = 1$.

Each of the above series of classes is in fact a chain of containments. For example,

$$\mathsf{NC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{NC}^2 \subseteq \cdots$$

Since an NC circuit is a special case of an AC circuit and an AC circuit is a special case of a TC circuit, for any k, we have that

$$\mathsf{NC}^k \subseteq \mathsf{AC}^k \subseteq \mathsf{TC}^k \subseteq \mathsf{NC}^{k+1}$$

, where the last inclusion holds with an NC^1 simulation of a **MAJORITY** gate [79]. The complexity class FOLL is the set of languages decidable by (uniform) AC circuits with depth $O(\log \log n)$ and polynomial size.

We define complexity classes by the languages reducible to another complexity class using some particular circuit class. For complexity classes $\mathcal{C}_1, \mathcal{C}_2$, we define $\mathcal{C}_1(\mathcal{C}_2)$ to be the class of languages that are $\mathcal{C}_1$ Turing-reducible to languages in $\mathcal{C}_2$.

The above circuit complexity classes are closely related to small-space bounded classes such as logarithmic space. In order to formalize the notion of logspace computation, we introduce the notion of a logspace transducer.

Definition 2.17. A logspace transducer is a Turing machine M with three tapes. It contains a read-only input tape, a read/write work tape for which at most $O(\log n)$ symbols can be used, and a write-only output tape.

Then this definition leads to two very natural complexity classes. L is the complexity class of languages decidable by deterministic logspace transducers, and NL is the complexity class of languages decidable by a non-deterministic logspace transducer. These are the logspace analogs of P and NP. Containments for all of the complexity classes covered thus far are shown below.

$$\mathsf{NC}^0 \subsetneq \mathsf{AC}^0 \subsetneq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{AC}^1 \subseteq \mathsf{TC}^1 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P} \subseteq \mathsf{NP}$$

For a complexity class $\mathcal{C}$, we define $\beta_i\mathcal{C}$ to be the set of languages L such that there exists an $L' \in \mathcal{C}$ such that $x \in L$ if and only if there exists y of length at most $O(\log^i |x|)$ such that

$(x, y) \in L'$.

Definition 2.18. An MAC^0 circuit is an AC^0 circuit with at most one **MAJORITY** gate.

Remark 2.19. The class MAC^0 was introduced in [4] and given its name in [53].

In this paper, to prove that an isomorphism test is in $\beta_1\mathsf{MAC}^0(\mathsf{FOLL})$, we run count-free WL Version I for $O(\log\log n)$ rounds, which is FOLL-computable [45]. Then we build a distinguisher to analyze the resulting coloring. The distinguisher works as follows.

1. Using $O(\log n)$ nondeterministic bits, guess a color class C where the multiplicity of C in G is more abundant than that in H.

2. With a single AC^0 circuit, identify all k-tuples with color C. This is possible using the parallel WL implementation of Grohe & Verbitsky [45], which records this information at each round using indicator variables.

3. Using a single **MAJORITY** gate, input a 1 into the gate for every k-tuple in G with color C. Input a 0 into the **MAJORITY** gate for each k-tuple in H with color C. The **MAJORITY** gate will return true if more than half the inputs are true.

Then since count-free WL Version I runs in FOLL then running count-free WL in conjunction with this distinguisher yields a $\beta_1\mathsf{MAC}^0(\mathsf{FOLL})$ isomorphism test.

2.5 Weisfeiler–Leman

The main algorithm of study in this thesis is the Weisfeiler–Leman algorithm for graphs, which computes an isomorphism-invariant coloring. Let Γ be a graph, and let $k \geq 2$ be an integer. The k-dimension Weisfeiler–Leman, or k-WL, algorithm begins by constructing an initial coloring $\chi_0 : V(\Gamma)^k \to \mathcal{K}$, where $\mathcal{K}$ is our set of colors, by assigning each k-tuple a color based on its isomorphism type. That is, two k-tuples $(v_1, \ldots, v_k)$ and $(u_1, \ldots, u_k)$ receive the same color under χ_0 iff the map $v_i \mapsto u_i$ (for all $i \in [k]$) is an isomorphism of the induced subgraphs $\Gamma[\{v_1, \ldots, v_k\}]$ and $\Gamma[\{u_1, \ldots, u_k\}]$ and for all i, j, $v_i = v_j \Leftrightarrow u_i = u_j$.

For $r \geq 0$, the coloring computed at the rth iteration of Weisfeiler–Leman is refined as follows. For a k-tuple $\overline{v} = (v_1, \ldots, v_k)$ and a vertex $x \in V(\Gamma)$, define

$$\overline{v}(v_i/x) = (v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_k).$$

The coloring computed at the $(r+1)$st iteration, denoted χ_{r+1}, stores the color of the given k-tuple $\overline{v}$ at the rth iteration, as well as the colors under χ_r of the k-tuples obtained by substituting a single vertex in $\overline{v}$ for another vertex x. We examine this multiset of colors over all such vertices x. This is formalized as follows:

$$\chi_{r+1}(\overline{v}) = (\chi_r(\overline{v}), \{\!\{(\chi_r(\overline{v}(v_1/x)), \ldots, \chi_r(\overline{v}(v_k/x)) \big| x \in V(\Gamma)\}\!\}),$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset.

The *count-free* variant of WL considers the set rather than the multiset of colors at each round. Precisely:

$$\chi_{r+1}(\overline{v}) = (\chi_r(\overline{v}), \{(\chi_r(\overline{v}(v_1/x)), \ldots, \chi_r(\overline{v}(v_k/x)) \big| x \in V(\Gamma)\}).$$

Note that the coloring χ_r computed at iteration r induces a partition of $V(\Gamma)^k$ into color classes. The Weisfeiler–Leman algorithm terminates when this partition is not refined, that is, when the partition induced by χ_{r+1} is identical to that induced by χ_r. The final coloring is referred to as the *stable coloring*, which we denote $\chi_\infty := \chi_r$.

Brachter & Schweitzer introduced three variants of WL for groups [15]. WL Versions I and II are both executed directly on the groups, where k-tuples of group elements are initially colored. For WL Version I, two k-tuples $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$ receive the same initial color iff (a) for all $i, j, \ell \in [k]$, $g_i g_j = g_\ell \iff h_i h_j = h_\ell$, and (b) for all $i, j \in [k]$, $g_i = g_j \iff h_i = h_j$. For WL Version II, $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$ receive the same initial color iff the map $g_i \mapsto h_i$ for all $i \in [k]$ extends to an isomorphism of the generated subgroups $\langle g_1, \ldots, g_k \rangle$ and $\langle h_1, \ldots, h_k \rangle$. For both WL Versions I and II, refinement is performed in the classical manner as for graphs. Namely, for a given k-tuple $\overline{g}$ of group elements,

$$\chi_{r+1}(\overline{g}) = (\chi_r(\overline{g}), \{\!\{(\chi_r(\overline{g}(g_1/x)), \ldots, \chi_r(\overline{g}(g_k/x)) \big| x \in G\}\!\}).$$

We will not use WL Version III, and so we refer the reader to [15] for details.

2.6 Logics

We recall the central aspects of first-order logic. We have a countable set of variables $\{x_1, x_2, \ldots, \}$. Formulas are defined inductively. As our basis, $x_i = x_j$ is a formula for all pairs of variables. Now if φ, ψ are formulas, then so are the following: $\varphi \wedge \psi, \varphi \vee \psi, \neg\varphi, \exists x_i\, \varphi$, and $\forall x_i\, \varphi$. In order to define logics on groups, it is necessary to define a relation that relates the group multiplication. We recall the two different logics introduced by Brachter & Schweitzer [15].

- **Version I:** We add a ternary relation R where $R(x_i, x_j, x_\ell) = 1$ if and only if $x_i x_j = x_\ell$ in the group. In keeping with the conventions of [21], we refer to the first-order logic with relation R as $\mathcal{L}^I$ and its k-variable fragment as $\mathcal{L}^I_k$. We refer to the logic $\mathcal{C}^I$ as the logic obtained by adding counting quantifiers $\exists^{\geq n} x_i\, \varphi$ (there exist at least n distinct x_i that satisfy φ) and $\exists! n\, \varphi$ (there exist exactly n distinct x_i that satisfy φ) and its k-variable fragment as $\mathcal{C}^I_k$. If furthermore we restrict the formulas to have quantifier depth at most r, we denote this fragment as $\mathcal{C}^I_{k,r}$.

- **Version II:** We add a relation R, defined as follows. Let $w \in (\{x_{i_1}, \ldots, x_{i_t}\} \cup \{x_{i_1}^{-1}, \ldots, x_{i_t}^{-1}\})^*$. We have that $R(x_{i_1}, \ldots, x_{i_t}; w) = 1$ if and only if multiplying the group elements according to w yields the identity. For instance, $R(a, b; [a, b])$ holds precisely if a, b commute. Again, in keeping with the conventions of [21], we refer to the first-order logic with relation R as $\mathcal{L}^{II}$ and its k-variable fragment as $\mathcal{L}^{II}_k$. We refer to the logic $\mathcal{C}^{II}$ as the logic obtained by adding counting quantifiers $\exists^{\geq n} x_i\, \varphi$ and $\exists! n\, \varphi$

and its k-variable fragment as $\mathcal{C}_k^{II}$. If furthermore we restrict the formulas to have quantifier depth at most r, we denote this fragment as $\mathcal{C}_{k,r}^{II}$.

Remark 2.20. Brachter & Schweitzer [15] refer to the logics with counting quantifiers as $\mathcal{L}_I$ and $\mathcal{L}_{II}$. We instead adhere to the conventions in [21].

Let $J \in \{I, II\}$. Brachter & Schweitzer [15] established that two groups G, H are distinguished by (k, r)-WL Version J if and only if there exists a formula $\varphi \in \mathcal{C}_{k+1,r}^J$ such that $G \models \varphi$ and $H \not\models \varphi$. Following the techniques of Brachter & Schweitzer, Grochow & Levet [34] established the analogous result for count-free WL Version J and the logic $\mathcal{L}^J$. In the setting of graphs, the equivalence between Weisfeiler–Leman and first-order logic with counting quantifiers is well known [51, 21].

2.7 Weisfeiler–Leman as a Parallel Algorithm

Grohe & Verbitsky [45] previously showed that for fixed k, the classical k-dimensional Weisfeiler–Leman algorithm for graphs can be effectively parallelized. Precisely, each iteration of the classical counting WL algorithm (including the initial coloring) can be implemented using a logspace uniform TC^0 circuit, and each iteration of the *count-free* WL algorithm can be implemented using a logspace uniform AC^0 circuit. As they mention ([45, Remark 3.4]), their implementation works for any first-order structure, including groups. However, because here we have three different versions of WL, we explicitly list out the resulting parallel complexities, which differ slightly between the versions.

- **WL Version I:** Let $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$ be two k-tuples of group elements. We may test in AC^0 whether (a) for all $i, j, m \in [k]$, $g_i g_j = g_m \iff h_i h_j = h_m$, and (b) $g_i = g_j \iff h_i = h_j$. So we may decide if two k-tuples receive the same initial color in AC^0. Comparing the multiset of colors at the end of each iteration (including after the initial coloring), as well as the refinement steps, proceed identically as in [45]. Thus, for fixed k, each iteration of k-WL Version I can be implemented using a logspace uniform TC^0 circuit. In the setting of the count-free k-WL Version I, we are comparing the set rather than multiset of colors at each iteration. So each iteration (including the initial coloring) can be implemented using a logspace uniform AC^0 circuit.

- **WL Version II:** Let $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$ be two k-tuples of group elements. We may use the marked isomorphism test of Tang [76] to test in L whether the map sending $g_i \mapsto h_i$ for all $i \in [k]$ extends to an isomorphism of the generated subgroups $\langle g_1, \ldots, g_k \rangle$ and $\langle h_1, \ldots, h_k \rangle$. So we may decide whether two k-tuples receive the same initial color in L. Comparing the multiset of colors at the end of each iteration (including after the initial coloring), as well as the refinement steps, proceed identically as in [45]. Thus, for fixed k, the initial coloring of k-WL Version II is L-computable, and each refinement step is TC^0-computable. In the case of the count-free k-WL Version II, the initial coloring is still L-computable, while each refinement step can be implemented can be implemented using a logspace uniform AC^0 circuit.

2.8 Pebble Game

In practice, we find that multisets of colors of k-tuples can difficult to work with. We heavily utilize the bijective pebble game introduced by [48, 49] for WL on graphs. This game is often used to show that two graphs X and Y cannot be distinguished by k-WL. The game is an Ehrenfeucht–Fraïssé bijective pebble game (c.f., [29, 64]), with two players: Spoiler and Duplicator. The game begins with $k+1$ pairs of pebbles, which are placed beside the graph. Each round proceeds as follows.

1. Spoiler picks up a pair of pebbles (p_i, p_i').
2. We check the winning condition, which will be formalized below.
3. Duplicator chooses a bijection $f : V(X) \to V(Y)$.
4. Spoiler places p_i on some vertex $v \in V(X)$. Then p_i' is placed on $f(v)$.

Let $v_1, \ldots, v_m$ be the vertices of X pebbled at the end of step 1, and let $v_1', \ldots, v_m'$ be the corresponding pebbled vertices of Y. Spoiler wins precisely if the map $v_\ell \mapsto v_\ell'$ does not extend to an isomorphism of the induced subgraphs $X[\{v_1, \ldots, v_m\}]$ and $Y[\{v'_1, \ldots, v'_m\}]$. Duplicator wins otherwise. Spoiler wins, by definition, at round 0 if X and Y do not have the same number of vertices. X and Y are not distinguished by the first r rounds of k-WL if and only if Duplicator wins the first r rounds of the $(k+1)$-pebble game [48, 49, 21].

Versions I and II of the pebble game are defined analogously, where Spoiler pebbles group elements. We first introduce the notion of marked equivalence. Let $\overline{u} := (u_1, \ldots, u_k), \overline{v} := (v_1, \ldots, v_k)$ be k-tuples of group elements. We say that $\overline{u}$ and $\overline{v}$ are *marked equivalent* in WL Version I iff (i) for all $i, j \in [k]$, $u_i = u_j \iff v_i = v_j$, and (II) for all $i, j, \ell \in [k]$, $u_i u_j = u_\ell \iff v_i v_j = v_\ell$. We say that $\overline{u}$ and $\overline{v}$ are marked equivalent in WL Version II if the map $u_i \mapsto v_i$ extends to an isomorphism of the generated subgroups $\langle u_1, \ldots, u_k \rangle$ and $\langle v_1, \ldots, v_k \rangle$.

We now turn to formalizing Versions I and II of the pebble game. Precisely, for groups G and H, each round proceeds as follows.

1. Spoiler picks up a pair of pebbles (p_i, p'_i).
2. We check the winning condition, which will be formalized below.
3. Duplicator chooses a bijection $f : G \to H$.
4. Spoiler places p_i on some vertex $g \in G$. Then p'_i is placed on $f(g)$.

Suppose that $(g_1, \ldots, g_\ell) \mapsto (h_1, \ldots, h_\ell)$ have been pebbled. Duplicator wins at the given round if this map is a marked equivalence in the corresponding version of WL. Brachter & Schweitzer established that for $J \in \{I, II\}$, (k, r)-WL Version J is equivalent to version J of the $(k+1)$-pebble, r-round pebble game [15].

Remark 2.21. In our work, we explicitly control for both pebbles and rounds. In our theorem statements, we state explicitly the number of pebbles on the board. So if Spoiler can win with k pebbles on the board, then we are playing in the $(k+1)$-pebble game. Note that k-WL corresponds to k-pebbles on the board.

Brachter & Schweitzer [15, Theorem 3.9] also previously showed that WL Version I, II, and III are equivalent up to a factor of 2 in the dimension, though they did not control for rounds. Following the proofs of Brachter & Schweitzer [15] for the bijective games, Grochow & Levet [34, Appendix A] showed that only $O(\log n)$ additional rounds are necessary.

There exist analogous pebble games for count-free WL Versions I-III. The count-free $(k+1)$-pebble game consists of two players: Spoiler and Duplicator, as well as $(k+1)$ pebble pairs (p, p'). In Versions I and II, Spoiler wishes to show that the two groups G and H are not isomorphic; and in Version III, Spoiler wishes to show that the corresponding graphs Γ_G, Γ_H are not isomorphic. Duplicator wishes to show that the two groups (Versions I and II) or two graphs (Version III) are isomorphic. Each round of the game proceeds as follows.

1. Spoiler picks up a pebble pair (p_i, p_i').

2. The winning condition is checked. This will be formalized later.

3. In Versions I and II, Spoiler places one of the pebbles on some group element (either p_i on some element of G or p_i' on some element of H). In Version III, Spoiler places one of the pebbles on some vertex of one of the graphs (either p_i on some vertex of Γ_G or p_i' on some element of Γ_H).

4. Duplicator places the other pebble on some element of the other group (Versions I and II) or some vertex of the other graph (Version III).

Let $v_1, \ldots, v_m$ be the pebbled elements of G (resp., Γ_G) at the end of step 1, and let $v_1', \ldots, v_m'$ be the corresponding pebbled vertices of H (resp., Γ_H). Spoiler wins precisely if the map $v_\ell \mapsto v_\ell'$ does not extend to a marked equivalence in the appropriate version of count-free WL. Duplicator wins otherwise. Spoiler wins, by definition, at round 0 if G and H do not have the same number of elements. We note that G and H (resp., Γ_G, Γ_H) are not distinguished by the first r rounds of the count-free k-WL if and only if Duplicator wins the first r rounds of the count-free $(k+1)$-pebble game. Grochow & Levet [34] established the

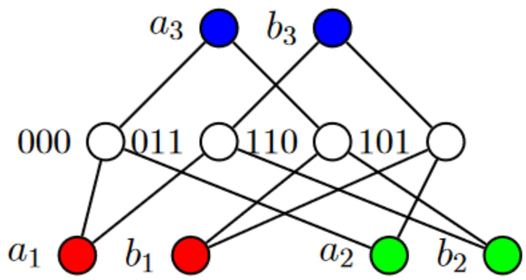


Figure 1: The CFI gadget F_3 [21, 15].

equivalence between Versions I and II of the count-free pebble game and the count-free WL algorithm for groups.

The count-free r-round, k-WL algorithm for graphs is equivalent to the r-round, $(k+1)$-pebble count-free pebble game [21]. Thus, the count-free r-round, k-WL Version III algorithm for groups introduced in Brachter & Schweitzer [15] is equivalent to the r-round, $(k+1)$-pebble count-free pebble game on the graphs Γ_G, Γ_H associated to the groups G, H.

2.9 CFI Graphs

Cai, Fürer, & Immerman [21] previously established that for every k, there exist a pair of graphs that are indistinguishable by k-WL. We recall their construction, which we denote the CFI construction, here. We begin with a connected base graph Γ. In Γ, each vertex is replaced by a particular gadget, and the gadgets are interconnected according to the edges of Γ as follows. For a vertex of degree d, we define the gadget F_d to be the graph whose vertex set consists of a set of external vertices $O_d = \{a_1^v, b_1^v, \ldots, a_d^v, b_d^v\}$ and a set of internal vertices M_d which are labeled according to the strings in $\{0,1\}^d$ that have an even number of 1's. For each i, each internal vertex u of M_d is adjacent to exactly one of $\{a_i^v, b_i^v\}$; namely u is adjacent to a_i if the ith bit of the string is 0 and b_i otherwise. An example of F_3 is included here (see Figure 1).

We now discuss how the gadgets are interconnected. Let $xy \in E(\Gamma)$. For each pair of external vertices (a_i^x, b_i^x) on the gadget corresponding to x and each pair of external vertices

(a_j^y, b_j^y) on the gadget corresponding to y, we add parallel edges $a_i^x a_j^y, b_i^x b_j^y$. We refer to the resulting graph as CFI(Γ). The *twisted CFI-graph* $\widetilde{\text{CFI}(\Gamma)}$ is obtained by taking one pair of parallel edges $a_i^x a_j^y, b_i^x b_j^y$ from CFI(Γ) and replacing these edges with the twist $a_i^x b_j^y, b_i^x a_j^y$. Up to isomorphism, it does not matter which pair of parallel edges we twist [21]. For a subset of edges $E' \subseteq E(\Gamma)$ of the base graph, we can define the graph obtained by twisting exactly the edges in E'. The resulting graph is isomorphic to CFI(Γ) if $|E'|$ is even and isomorphic to $\widetilde{\text{CFI}(\Gamma)}$ otherwise.

In the original construction [21], the base graph is generally taken to be a vertex-colored graph where each vertex has a different color. As a result, all of the gadgets in the CFI construction are distinguishable. The colors can be removed by instead attaching gadgets to each vertex, and these gadgets can be attached in such that the base graph is identified by 2-WL. In particular, it is possible to choose a 3-regular base graph to have WL-dimension 2 [15, Observation 2.2].

3 Results

3.1 Weisfeiler–Leman and the CFI Groups

In this section, we establish the following.

Theorem 3.1. *Let Γ_0 be a 3-regular connected graph, and let $\Gamma_1 := CFI(\Gamma_0)$ and $\Gamma_2 := \widetilde{CFI(\Gamma_0)}$ be the corresponding CFI graphs. For $i = 1, 2$, denote $G_i := G_{\Gamma_i}$ be the corresponding groups arising from Mekler's construction. We have that the $(3, O(\log \log n))$-WL Version I distinguishes G_1 from G_2. If furthermore Γ_0 is identified by the graph $(3, r)$-WL algorithm, then the $(3, \max\{r, O(\log \log n)\})$-WL Version I algorithm identifies G_1 as well as G_2.*

Remark 3.2. We compare Thm. 3.1 to [15, Theorem 6.1], where Brachter & Schweitzer established the analogous result in WL Version II without controlling for rounds. A careful analysis of their work shows that they only use $O(\log n)$ rounds. This yields an upper bound of TC^1. In contrast, Thm. 3.1 improves the depth of the circuit from $O(\log n)$ to $O(\log \log n)$. Additionally, the analysis of CFI groups in [15] relies almost exclusively on the underlying graph structure of CFI graphs. Hence, we do not need the full power of deciding whether two 3-tuples of group elements generate isomorphic subgroups and we can get away using WL Version I instead of Version II. The best upper bound for the initial coloring of Version II is L [76] while Version I's initial coloring is known to be TC^0-computable. Both refinement steps are TC^0-computable. With an improvement to the number of rounds and to the initial coloring, these two observations give us a $\mathsf{TC}^{o(1)}$ upper bound.

We begin with some preliminary lemmas.

Lemma 3.3. *Let G, H be groups. Suppose Duplicator selects a bijection $f : G \to H$ such that $|C_G(x)| \neq |C_H(f(x))|$. Then Duplicator can win with 3 pebbles and 3 rounds.*

Proof. Without loss of generality, suppose that $|C_G(x)| > |C_H(f(x))|$. Spoiler begins by pebbling $x \mapsto f(x)$. Let $f' : G \to H$ be the bijection that Duplicator selects the next round. As $|C_G(x)| > |C_H(f(x))|$, there exists $y \in C_G(x)$ such that $f'(y) \notin C_H(f(x))$. At the next two rounds, Spoiler pebbles y, xy and wins. □

Remark 3.4. Brachter & Schweitzer [15] established that for the CFI groups G_1, G_2, group elements with single-vertex support have centralizers of size $p^4 \cdot |Z(G)|$, while all other group elements have centralizers of size at most $p^3 \cdot |Z(G)|$. So by Lem. 3.3, if Duplicator does not preserve single-support vertices, then Spoiler can win with 2 additional pebbles and 2 additional rounds.

Lemma 3.5 (Compare rounds c.f. [15, Lemma 6.3])**.** *Let G_1, G_2 be the CFI groups. Let $f : G_1 \to G_2$ be the bijection that Duplicator selects. If there exists $x \in G_1$ such that $|\operatorname{supp}(x)| \neq |\operatorname{supp}(f(x))|$, then Spoiler can win with 3 pebbles and $O(\log \log |G_1|)$ rounds.*

Proof. Without loss of generality, suppose that $|\operatorname{supp}(x)| < |\operatorname{supp}(f(x))|$. Spoiler begins by pebbling $x \mapsto f(x)$. Write $x = x_{i_1}^{d_1} \cdots x_{i_r}^{d_r} \cdot c$, where the $x_{i_1}, \ldots, x_{i_r}$ correspond to vertices $v_{i_1}, \ldots, v_{i_r} \in V(\Gamma_1)$ as per Mekler's construction, and $c \in Z(G_1)$. Let $m := \lceil r/2 \rceil$. At the next two rounds, Spoiler pebbles $x_{i_1}^{d_1} \cdots x_{i_m}^{d_m} \mapsto u$ and $x_{i_{m+1}}^{d_{i_{m+1}}} \cdots x_{i_r}^{d_r} \cdot c \mapsto v$. Now if $f(x) \neq uv$, then Spoiler wins. So suppose $f(x) = uv$. As $|\operatorname{supp}(x)| \neq |\operatorname{supp}(f(x))|$, we have that either:

$$|\operatorname{supp}(x_{i_1}^{d_1} \cdots x_{i_m}^{d_m})| < |\operatorname{supp}(u)| \text{ or},$$
$$|\operatorname{supp}(x_{i_{m+1}}^{d_{i_{m+1}}} \cdots x_{i_r}^{d_r} \cdot c)| < |\operatorname{supp}(v)|.$$

Without loss of generality, suppose that: $|\operatorname{supp}(x_{i_1}^{d_1} \cdots x_{i_m}^{d_m})| < |\operatorname{supp}(u)|$. Spoiler iterates on the above argument starting from $x_{i_1}^{d_1} \cdots x_{i_m}^{d_m} \mapsto u$ and reusing the pebbles on $x, x_{i_{m+1}}^{d_{i_{m+1}}} \cdots x_{i_r}^{d_r} \cdot c$. Now as $|\operatorname{supp}(x_{i_1}^{d_1} \cdots x_{i_m}^{d_m})| \leq |\operatorname{supp}(x)/2|$, we iterate on this argument at most $\log_2(|\operatorname{supp}(x)|) + 1 \leq \log |V(\Gamma_1)| \leq \log \log |G_1| + O(1)$ times until we reach a base case where our pebbled element $x' \in G_1$ has support size 1, but the corresponding pebbled element $y' \in H$ has support size > 1. In this case, Spoiler at the next round reuses one of the other two pebbles on the board to pebble some element in $C_{G_1}(x')$ whose image does not belong to $C_{G_2}(y')$. The result now follows. $\square$

Lemma 3.6 (Compare rounds c.f. [15, Lemma 6.4])**.** *Let $u \in V(\Gamma_1)$, and let $g_u \in G_1$ be a single-support element that is supported by u.*

(a) *Suppose $v \in V(\Gamma_1)$ belongs to the same gadget as u, and let $g_v \in G_1$ be a single-support element that is supported by v. Let $f : G_1 \to G_2$ be the bijection that Duplicator selects. If $f(g_u g_v)$ is not supported by exactly two vertices x, y on the same gadget of $V(\Gamma_2)$, then Spoiler can win with 3 pebbles and $O(1)$ rounds.*

(b) *Suppose that $g_u \mapsto x$ has been pebbled. Let $f : G_1 \to G_2$ be the bijection that Duplicator selects at the next round. Let $v \in V(\Gamma_1)$ be a vertex on the same gadget as u, and suppose that for some single-support element $g_v \in G_1$ that is supported by v, that $f(g_v)$ belongs to a different gadget than $f(g_u) = x$. Then Spoiler can win with 3 pebbles and $O(1)$ rounds.*

(c) *Suppose that $u \in V(\Gamma_1)$ is an internal vertex on some gadget, and let $g_u \in G_1$ be a single-support element that is supported by u. Suppose that Duplicator selects a bijection*

$f : G_1 \to G_2$ where $f(g_u)$ is a single-support vertex is supported by an external vertex of some gadget. Then Spoiler can win with 3 pebbles and $O(1)$ rounds.

Proof. We proceed as follows.

(a) Spoiler begins by pebbling $g_u g_v \mapsto f(g_u g_v)$. Now if $|\text{supp}(f(g_u g_v))| > 2$, Spoiler pebbles $g_u \mapsto x, g_v \mapsto y$ at the next two rounds. If $f(g_u g_v) \neq xy$, then Spoiler immediately wins. So suppose $f(g_u g_v) = xy$. As $|\text{supp}(f(g_u g_v))| > 2$, either $|\text{supp}(x)| > 1$ or $|\text{supp}(y)| > 1$. Without loss of generality, suppose $|\text{supp}(x)| > 1$. In this case, Spoiler wins by the argument in the proof of Lem. 3.3, reusing the pebbles on $g_v, g_u g_v$.

So suppose now that $|\text{supp}(f(g_u g_v))| = 2$. At the next two rounds, Spoiler pebbles $g_u \mapsto x, g_v \mapsto y$. Now by the CFI construction [21], the graphs Γ_1, Γ_2 have the property that for every 6-cycle and every 8-cycle, there exists a single gadget that contains said cycle. That is, no 6-cycle and no 8-cycle span two gadgets. Moreover, every pair of vertices lying on the same gadget lie on some common 6-cycle or same 8-cycle. Thus, Spoiler may reuse the pebble on $g_u g_v$ and trace along the cycle containing g_u, g_v starting from g_u. Within at most 4 additional rounds, Spoiler will have moved this third pebble to a neighbor of g_v, while the corresponding pebble will not be along a neighbor of y. Spoiler now wins.

(b) Spoiler begins by pebbling $g_v \mapsto f(g_v)$. Using a third pebble, we now proceed identically as in the second paragraph of part (a). The result follows.

(c) Spoiler begins by pebbling $g_u \mapsto f(g_u)$. We note that if $f(g_u)$ is not supported by a single vertex, then by Lem. 3.3, Spoiler can win with 2 additional pebbles and 2 additional rounds. So suppose $f(g_u)$ is supported by the vertex $x \in V(\Gamma_2)$. Let $f' : G_1 \to G_2$ be the bijection that Duplicator selects at the next round. Let $y \in V(\Gamma_2)$ be an external vertex adjacent to x, and let $h_y \in G_2$ be a single-support group element that is supported by y. Let $g \in G_1$ such that $f'(g) = h_y$. Spoiler pebbles $g \mapsto h_y$.

We may again assume that g has single support, or Spoiler wins in one additional pebble (beyond reusing the pebble on g_u) and two additionals rounds. By part (b), we may assume that g_u, g belong to different gadgets, or Spoiler wins with 1 additional pebble and $O(1)$ additional rounds. But as g_u is supported by an internal vertex, so the vertices supporting g_u, g are not adjacent in Γ_1. By Mekler's construction, this implies that g_u, g do not commute. However, $f(g_u), h_y$ are adjacent and do commute. So Spoiler pebbles $g_u g$ and wins.

□

For convenience, we pull out the following construction.

Definition 3.7. Let G_i $(i = 1, 2)$ be a CFI group. We first define a set $\mathcal{V}$ of vertices in Γ_1 as follows. For each gadget, we include a single, arbitrary internal vertex and all adjacent external vertices. Let $v \in G_1$ denote the ordered product of all the vertices in $\mathcal{V}$.

We now prove Thm. 3.1.

Proof of Thm. 3.1 . We follow the strategy of [15, Theorem 6.1], carefully analyzing the number of rounds. We first define a set $\mathcal{V}$ of vertices in Γ_1 according to Def. 3.7. Let $v \in G_1$ denote the ordered product of all the vertices in $\mathcal{V}$. By Lem. 3.5, Duplicator must choose a bijection $f : G_1 \to G_2$ in which $f(v)$ has the same support size as v. Otherwise, Spoiler can win with 3 pebbles and $O(\log \log n)$ rounds.

Spoiler begins by pebbling $v \mapsto f(v)$. Now by Lem. 3.5, Duplicator must select bijections that map (setwise) $\text{supp}(v) \mapsto \text{supp}(f(v))$; otherwise, Spoiler can win with 2 additional pebbles and $O(\log \log n)$ rounds. By Lem. 3.6, $\text{supp}(f(v))$ must be composed exactly as $\text{supp}(v)$; otherwise, Spoiler wins with 2 additional pebbles and $O(1)$ rounds. That is, $\mathcal{V}' := \text{supp}(f(v))$ must contain exactly one internal vertex and all adjacent external vertices from each gadget of Γ_2 (i.e., $\mathcal{V}'$ must also satisfy Def. 3.7).

Now in the proof of [15, Theorem 6.1], Brachter & Schweitzer showed that the induced subgraphs $\Gamma_1[\mathcal{V}]$ and $\Gamma_2[\mathcal{V}']$ have a different number of edges modulo 2. In particular, $\Gamma_1[\mathcal{V}]$ and $\Gamma_2[\mathcal{V}']$ disagree in exactly one edge: the twisted link.

Let $f' : G_1 \to G_2$ be the bijection that Duplicator selects at the next round. As the number of edges in $\Gamma_1[\mathcal{V}]$ and $\Gamma_2[\mathcal{V}']$ disagree, there exists a single-support vertex $g \in G_1$ such that the vertex supporting g has degree in $\Gamma_1[\mathcal{V}]$ that is different than the degree of the vertex supporting $f'(g)$ in $\Gamma_2[\mathcal{V}']$. Spoiler pebbles $g \mapsto f'(g)$. At the next round, Duplicator must select a bijection $f'' : G_1 \to G_2$ that maps some element u of $\mathcal{V}$ that commutes with g to some element $f''(u)$ of $\mathcal{V}'$ that does not commute with $f'(g)$ (or vice-versa). Spoiler pebbles $u \mapsto f''(u)$. Then at the next round, moves the pebble on v to gu and wins. In total, Spoiler used at most 3 pebbles on the board and $O(\log \log n)$ rounds.

Furthermore, suppose that Γ_0 is identified by the graph $(3, r)$-WL. Brachter & Schweitzer [15] previously established that all single-support group elements of G_1, G_2 have centralizers of size $p^4 \cdot |Z(G_1)|$, and all other group elements have centralizers of size at most $p^3 \cdot |Z(G_1)|$. Now let H be an arbitrary group, and suppose 3-WL Version I fails to distinguish G_i $(i = 1, 2)$ from H in $\max\{r, O(\log \log n)\}$ rounds. Then H has the same number of group elements with centralizers of size $p^4 \cdot |Z(G_1)|$. Furthermore, as 3-WL Version I fails to distinguish G_i $(i = 1, 2)$ from H in $\max\{r, O(\log \log n)\}$ rounds, the induced commutation graph on these elements in $H/Z(H)$ is indistinguishable from Γ_i. Furthermore, by Lem. 3.6, $(3, O(1))$-WL Version I identifies internal vertices. So given G_i $(i = 1, 2)$, we can recover the base graph Γ_0. Furthermore, we can reconstruct the base graph Γ underlying H. Precisely, any bijection $f : G_i \to H$ induces a bijection $\tilde{f} : V(\Gamma_0) \to V(\Gamma)$, and so we may simulate the 4-pebble, r-round strategy to identify Γ_0 in the graph pebble game, by pebbling the appropriate elements of G_i $(i = 1, 2)$. But since Γ_0 is identified by the graph $(3, r)$-WL, we have that $\Gamma_0 \cong \Gamma$. So H is isomorphic to either G_1 or G_2. The result now follows. □

3.2 Count-Free Strategy and the CFI Groups

In this section, we establish the following theorem.

Theorem 3.8. *Let Γ_0 be a 3-regular connected graph, and let $\Gamma_1 := \mathit{CFI}(\Gamma_0)$ and $\Gamma_2 := \widetilde{\mathit{CFI}(\Gamma_0)}$ by the corresponding CFI graphs. For $i = 1, 2$, denote $G_i := G_{\Gamma_i}$ to be the corresponding groups arising from Mekler's construction.*

(a) The multiset of colors computed by the count-free $(O(1), O(\log \log n))$-WL Version I distinguishes G_1 from G_2. In particular, we can decide whether $G_1 \cong G_2$ in $\beta_1\mathsf{MAC}^0(\mathsf{FOLL})$.

(b) If furthermore Γ_0 is identified by the graph count-free $(3, r)$-WL algorithm, then the multiset of colors computed by the count-free $(O(1), \max\{r, O(\log \log n)\})$-WL Version I will distinguish G_i $(i = 1, 2)$ from an arbitrary graph H. In particular, we can decide whether $G_i \cong H$ in $\beta_1\mathsf{MAC}^0(\mathsf{FOLL})$.

We proceed similarly as in the case of counting WL. We begin with the following lemma.

Lemma 3.9. *Let G_i be a (twisted) CFI group $(i = 1, 2)$. Let $u, v \in G_i$ where $|\operatorname{supp}(u)| = 1$ and $|\operatorname{supp}(v)| > 1$. Suppose that $u \mapsto v$ has been pebbled. Spoiler can win with $O(1)$ additional pebbles and $O(\log \log n)$ additional rounds.*

Proof. Brachter & Schweitzer [15] previously established that $|C_{G_i}(u)| = p^4 \cdot |Z(G_i)|$ and $|C_{G_i}(v)| \leq p^3 \cdot |Z(G)|$. Now by [15, Lemma 4.7] (recalled as Lem. 2.11), we have that $Z(G_i) = \Phi(G_i) = [G_i, G_i]$. In particular, as G_i is a class 2 p-group of exponent $p > 2$, we have that $G_i/Z(G_i)$ is elementary Abelian. So $C_{G_i}(u)/Z(G_i) \cong (\mathbb{Z}/p\mathbb{Z})^4$ and $C_{G_i}(v)/Z(G_i) \cong (\mathbb{Z}/p\mathbb{Z})^d$ for some $d \leq 3$. Spoiler now pebbles a representative g_1, g_2, g_3, g_4 of each coset for $C_{G_i}(u)/Z(G_i)$. Let h_1, h_2, h_3, h_4 be the corresponding elements Duplicator pebbles. Necessarily, one such element either does not belong to $C_{G_i}(v)$ or belongs to $\langle h_1, h_2, h_3\rangle \cdot Z(G_i)$. Without loss of generality, suppose this element is h_4. If $h_4 \not\in C_{G_i}(v)$, Spoiler wins by pebbling $h_4, h_4 v$ at the next two rounds.

So suppose that $h_4 \in C_{G_i}(v)$. Now each element of $\langle h_1, h_2, h_3\rangle \cdot Z(G_i)$ can be written as $h_1^{e_1} h_2^{e_2} h_3^{e_3} \cdot z$ for some $z \in Z(G_i)$. Using 3 additional pebbles, Spoiler can pebble $g_j^{e_j}$ $(j = 1, 2, 3)$. If Duplicator does not respond by pebbling $h_j^{e_j}$, then by [34, Lemma 3.12], Spoiler can win with $O(1)$ additional pebbles in $O(\log \log n)$ rounds. Spoiler now pebbles $z \in Z(G_i)$ such that $h_4 = h_1^{e_1} h_2^{e_2} h_3^{e_3} \cdot z$ and wins. The result now follows. □

We now show that the count-free WL algorithm will distinguish group elements with different support sizes.

Lemma 3.10. *Let G_i be a (twisted) CFI group ($i = 1, 2$). Let $u, v \in G_i$ where $|\operatorname{supp}(u)| \neq |\operatorname{supp}(v)| > 1$. Suppose that $u \mapsto v$ has been pebbled. Spoiler can win with 4 additional pebbles and $O(\log \log n)$ additional rounds.*

Proof. Aside from Spoiler selecting an element x to pebble, the proof of Lem. 3.5 did not rely on Duplicator selecting a bijection at each round. Thus, as $u \mapsto v$ has been pebbled, we may proceed identically as in Lem. 3.5. The result now follows. □

Our next goal is to show that count-free WL can detect the gadget structure of the underlying CFI graphs.

Lemma 3.11. *Let $u \in V(\Gamma_i)$ ($i = 1, 2$), and let $g_u \in G_i$ be a single-support element that is supported by u.*

(a) Let $v \in V(G_i)$ be on the same gadget as u, and let $g_v \in G_i$ be a single-support element that is supported by v. Suppose that $(g_u, g_v) \mapsto (h_u, h_v)$ have been pebbled, and that $h_u h_v$ is not supported by exactly two vertices on the same gadget. Then Spoiler can win with $O(1)$ additional pebbles and $O(1)$ rounds.

(b) Suppose now that u is an internal vertex. Suppose that $g_u \mapsto h_u$ has been pebbled, and that h_u is a single-support element supported by some x that is an external vertex. Then Spoiler can win with $O(1)$ additional pebbles and $O(1)$ rounds.

Proof. We proceed as follows.

(a) We note that if $|\operatorname{supp}(h_u h_v)| \neq 2$, then either h_u or h_v are not single-support elements. In this case, Spoiler can win with $O(1)$ additional pebbles and $O(1)$ additional rounds by Lem. 3.9. So suppose $|\mathrm{supp}(h_u h_v)| = 2$. Let $\mathrm{supp}(h_u) = \{x\}$ and $\mathrm{supp}(h_v) = \{y\}$. Suppose that x, y belong to different gadgets. Now the CFI graphs have the property that two vertices belong to a common gadget if and only if they are on common 6-cycle or common 8-cycle [21]. Using two additional pebbles and $O(1)$ additional rounds, Spoiler wins by tracing around the cycle containing u, v.

(b) As x is an external vertex, x is adjacent to some other external vertex y. Let h_y be a single-support element that is supported by y. Spoiler pebbles h_y, and Duplicator responds by pebbling some single-support element g_v that is supported by the vertex v. If $uv \notin E(\Gamma_i)$, Spoiler immediately wins, as $h_u h_y$ commute, and $g_u g_v$ do not. So suppose $uv \in E(\Gamma_i)$. But as u is internal, u, v belong to the same gadget, while x, y do not. Spoiler now wins by part (a).

□

We now establish the relationship between the group elements arising from the construction in Def. 3.7 and the induced subgraphs from $\mathcal{V}$.

Lemma 3.12. *Let $v \in G_i$ $(i = 1, 2)$ such that v satisfies the construction in Def. 3.7. We have the following.*

(a) Let $v' \in G_{i'}$ $(i' = 1, 2)$ such that v' is not constructed according to Def. 3.7. Then the count-free $(O(1), O(\log \log n))$-WL Version I will distinguish v from v'.

(b) Let $v' \in G_{i'}$ $(i' = 1, 2)$ such that v' is constructed according to Def. 3.7. Let $u \in \mathrm{supp}(v)$, and let $g_u \in G_i$ be a single-support element that is supported by u. Let $h \in G_{i'}$. If h is not a single-support element satisfying $\mathrm{supp}(h) \subseteq \mathrm{supp}(v')$, then the count-free $(O(1), O(\log \log n))$-WL Version I will distinguish (v, g_u) from (v', h).

(c) Let v', g_u, h be as defined in part (b). Let $\mathrm{supp}(h) = \{u'\}$, and relabel $h_{u'} := h$. Suppose that $|N(u) \cap \mathrm{supp}(v)| \neq |N(u') \cap \mathrm{supp}(v')|$. That is, suppose that the degree of

u in $\Gamma_i[\text{supp}(v)]$ is different than the degree of u' in $\Gamma_{i'}[\text{supp}(v')]$. Then the count-free $(O(1), O(\log \log n))$-WL Version I will distinguish (v, g_u) from $(v', h_{u'})$.

Proof. We proceed as follows.

(a) By Lem. 3.10, if $v' \in G_i$ $(i = 1, 2)$ satisfies $|\,\text{supp}(v')| \neq |\,\text{supp}(v)|$, then the $(O(1), O(\log \log n))$-WL Version I algorithm will distinguish v and v'.

Now suppose $\text{supp}(v')$ contains two vertices a, b on the same gadget. We claim that the count-free $(O(1), O(\log \log n))$-WL Version I will distinguish v from v'. Consider the pebble game, starting from the configuration $v \mapsto v'$. Spoiler pebbles group elements h_a, h_b supported by a, b respectively. Duplicator responds by pebbling g_x, g_y. By Lem. 3.10, we may assume that g_x, g_y are single-support elements; otherwise, Spoiler wins with $O(1)$ pebbles and $O(1)$ rounds. Let x, y be the vertices of Γ_i supporting g_x, g_y respectively. We may assume that $x, y \in \text{supp}(v)$. Otherwise, by Lem. 3.10, Spoiler wins with $O(1)$ pebbles and $O(\log \log n)$ rounds. By construction of v, x, y lie on different gadgets. Thus, by Lem. 3.11, Spoiler can win with $O(1)$ additional pebbles and $O(1)$ additional rounds. It now follows that any group element v' whose support does not consist of a single arbitrary internal vertex from each gadget and all external vertices adjacent to the internal vertices selected, that the count-free $(O(1), O(\log \log n))$-WL Version I will distinguish v and v'.

(b) If h is not single support or $\text{supp}(h) \not\subseteq \text{supp}(v')$, then by Lem. 3.10, the count-free $(O(1), O(\log \log n))$-WL Version I algorithm will distinguish (v, g_u) from (v', h).

(c) By Lem. 3.10, if $\text{supp}(h_u) \not\subseteq \text{supp}(v')$, then the count-free $(O(1), O(\log \log n))$-WL Version I will distinguish (v, g_u) and $(v', h_{u'})$. Now by the CFI construction [21], both $\Gamma_i, \Gamma_{i'}$ have maximum degree at most 4. So with $O(1)$ pebbles, Spoiler can pebble the neighbors of u in Γ_i. By construction, the adjacency relation in $\Gamma_i, \Gamma_{i'}$ determines the commutation relation in $G_i, G_{i'}$. Thus, only $O(1)$ additional pebbles and $O(1)$ additional rounds are needed to determine commutation. Thus, the count-free $(O(1), O(\log \log n))$-WL Version I will distinguish (v, g_u) and $(v', h_{u'})$.

$\square$

We now prove Thm. 3.8.

Proof of Thm. 3.8 . We proceed similarly as in the proof of Thm. 3.1.

(a) Let $v \in G_1$ be defined as in Def. 3.7. Now suppose that $G_1 \not\cong G_2$. Now take an arbitrary $v' \in G_2$ such that v' is defined according to Def. 3.7. Again take $\mathcal{V} = \text{supp}(v)$ and $\mathcal{V}' = \text{supp}(v')$. Brachter & Schweitzer [15, Proof of Theorem 6.1] argued that the induced subgraphs $\Gamma_1[\mathcal{V}]$ and $\Gamma_2[\mathcal{V}']$ have different degree sequences.

    Now suppose that $u \in \mathcal{V}$ and $u' \in \mathcal{V}'$ have different degrees. Let $g_u \in G_1$ be a single-support element supported by u, and let $h_{u'} \in G_2$ be a single-support element supported by u'. As u, u' have different degrees, we have by Lem. 3.12 (c) that the count-free $(O(1), O(\log \log n))$-WL Version I will distinguish (v, g_u) from $(v', h_{u'})$. In particular, it follows that the multiset of colors produced by the count-free $(O(1), O(\log \log n))$-WL Version I algorithm will be different for G_1 than for G_2.

    By Grohe & Verbitsky, we have that the count-free $(O(1), O(\log \log n))$-WL Version I can be implemented using an FOLL circuit. We will now use a $\beta_1\mathsf{MAC}^0$ circuit to distinguish G_1 from G_2. Using $O(\log n)$ non-deterministic bits, we guess a color class C where the multiplicity differs. At each iteration, the parallel WL implementation due to Grohe & Verbitsky records indicators as to whether two k-tuples receive the same color. As we have already run the count-free WL algorithm, we may in AC^0 decide whether two k-tuples have the same color. For each k-tuple in G_1^k having color C, we feed a 1 to the **MAJORITY** gate. For each k-tuple in G_2^k having color C, we feed a 0 to the **MAJORITY** gate. The result now follows.

(b) Suppose furthermore that the base graph Γ_0 is identified by the count-free $(O(1), r)$-WL algorithm for graphs. Brachter & Schweitzer [15] previously established that single-support elements of G_1, G_2 have centralizers of size $p^4 \cdot |Z(G_1)|$, and all other group elements have centralizers of size at most $p^3 \cdot |Z(G_1)|$. By Lem. 3.9, the count-free $(O(1), O(1))$-WL Version I will distinguish in G_i $(i = 1, 2)$ single-support group elements from those group elements g with $|\operatorname{supp}(g)| > 1$. Now let H be an arbitrary group, and suppose the multiset of colors produced by the count-free $(O(1), O(\log \log n))$-WL Version I is the same for G_i $(i = 1, 2)$ as for H. Then G_i $(i = 1, 2)$ and H has the same number of elements of order $p^4 \cdot |Z(G_1)|$. Furthermore, as the multiset of colors arising from the count-free $(O(1), O(\log \log n))$-WL Version I fails to distinguish G_i $(i = 1, 2)$ and H, the induced commutation graph on these elements in $H/Z(H)$ is indistinguishable from Γ_i. Furthermore, by Lem. 3.11 (b), the count-free $(O(1), O(1))$-WL Version I will distinguish internal and external vertices. So given G_i $(i = 1, 2)$, we can reconstruct the base graph Γ_0. Furthermore, we can reconstruct the base graph Γ underlying H. Precisely, the vertices of Γ_0 correspond to gadgets of Γ_i. Now the count-free WL Version I for groups can simulate the count-free WL for graphs in the following manner. When Spoiler or Duplicator pebble a single-support element of G_i, that induces placing a pebble on the corresponding vertex v of Γ_i. In turn, this induces placing a pebble on the vertex corresponding to the gadget of Γ_0 containing v. So we may simulate the $(3, r)$-pebble strategy to identify Γ_0 in the graph pebble game, by pebbling the appropriate elements of G_i $(i = 1, 2)$. But since Γ_0 is identified by the graph $(3, r)$-WL, we have that $\Gamma_0 \cong \Gamma$. So H is isomorphic to either G_1 or G_2. The result now follows.

□

4 Conlcusion

In this thesis, we presented an improved upper bound on the parallel and descriptive complexities on the identification of CFI groups. In particular, we improved upon the TC^1 upper bound presented in [15]. We demonstrated a $\mathsf{TC}^{o(1)}$ bound using WL Version I and a $\beta_1\mathsf{MAC}^0(FOLL)$ bound using count-free WL.

We conclude with several open problems.

Question 4.1. Can 2-WL distinguish the CFI groups?

Question 4.2. Can the 3-dimensional count-free WL algorithm distinguish CFI groups without the postprocessing illustrated in 2.4?

The central open question remains whether Weisfeiler–Leman solves GpI in polynomial time. Precisely:

Question 4.3. Does WL resolve Group Isomorphism in polynomial time? That is, does there exist a fixed k such that k-WL can distinguish any two nonisomorphic finite groups?

References

[1] Complexity zoo. URL: `https://complexityzoo.net`.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. 01 2009. `doi:10.1017/CBO9780511804090`.

[3] V. Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. *Information and Computation*, 204(5):835–852, 2006. `doi:10.1016/j.ic.2006.02.002`.

[4] James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, page 402–409, New York, NY, USA, 1991. Association for Computing Machinery. `doi:10.1145/103418.103461`.

[5] L. Babai, W. M. Kantor, and E. M. Luks. Computational complexity and the classification of finite simple groups. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 162–171, 1983. `doi:10.1109/SFCS.1983.10`.

[6] László Babai. Lectures on graph isomorphism. In *Mimeographed lecture notes*, 1979.

[7] László Babai. On the complexity of canonical labeling of strongly regular graphs. *SIAM Journal on Computing*, 9(1):212–216, 1980. arXiv:10.1137/0209018, doi:10.1137/0209018.

[8] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *STOC'16—Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 684–697. ACM, New York, 2016. Preprint of full version at arXiv:1512.03547v2 [cs.DS]. doi:10.1145/2897518.2897542.

[9] László Babai, Paolo Codenotti, Joshua A. Grochow, and Youming Qiao. Code equivalence and group isomorphism. In *Proceedings of the Twenty-Second Annual ACM–SIAM Symposium on Discrete Algorithms (SODA11)*, pages 1395–1408, Philadelphia, PA, 2011. SIAM. doi:10.1137/1.9781611973082.107.

[10] László Babai, Paolo Codenotti, and Youming Qiao. Polynomial-time isomorphism test for groups with no abelian normal subgroups - (extended abstract). In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 51–62, 2012. doi:10.1007/978-3-642-31594-7_5.

[11] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 171–183, New York, NY, USA, 1983. Association for Computing Machinery. doi:10.1145/800061.808746.

[12] László Babai and Youming Qiao. Polynomial-time isomorphism test for groups with Abelian Sylow towers. In *29th STACS*, pages 453 – 464. Springer LNCS 6651, 2012. doi:10.4230/LIPIcs.STACS.2012.453.

[13] Anton Betten, Michael Braun, Harald Fripertinger, Adalbert Kerber, Axel Kohnert, and Alfred Wassermann. *Error-correcting linear codes classification by isometry and applications.* Springer Berlin, 2014.

[14] Béla Bollobás. Distinguishing vertices of random graphs. In Béla Bollobás, editor, *Graph Theory*, volume 62 of *North-Holland Mathematics Studies*, pages 33–49. North-Holland, 1982. doi:10.1016/S0304-0208(08)73545-X.

[15] Jendrik Brachter and Pascal Schweitzer. On the Weisfeiler–Leman dimension of finite groups. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 287–300. ACM, 2020. doi:10.1145/3373718.3394786.

[16] Jendrik Brachter and Pascal Schweitzer. A Systematic Study of Isomorphism Invariants of Finite Groups via the Weisfeiler-Leman Dimension. 244:27:1–27:14, 2022. doi:10.4230/LIPIcs.ESA.2022.27.

[17] Gilles Brassard and Moti Yung. One-way group actions. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology-CRYPTO' 90*, pages 94–107, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[18] Peter A. Brooksbank, Joshua A. Grochow, Yinan Li, Youming Qiao, and James B. Wilson. Incorporating Weisfeiler–Leman into algorithms for group isomorphism. arXiv:1905.02518 [cs.CC], 2019.

[19] Peter A. Brooksbank, Joshua Maglione, and James B. Wilson. A fast isomorphism test for groups whose Lie algebra has genus 2. *Journal of Algebra*, 473:545–590, 2017. `doi:10.1016/j.jalgebra.2016.12.007`.

[20] Harry Buhrman and Steven Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In R. K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings*, volume 652 of *Lecture Notes in Computer Science*, pages 116–127. Springer, 1992. `doi:10.1007/3-540-56287-7\_99`.

[21] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. Originally appeared in SFCS '89. `doi:10.1007/BF01305232`.

[22] John J. Cannon and Derek F. Holt. Automorphism group computation and isomorphism testing in finite groups. *Journal of Symbolic Computation*, 35(3):241–267, 2003. `doi:10.1016/S0747-7171(02)00133-5`.

[23] Arkadev Chattopadhyay, Jacobo Torán, and Fabian Wagner. Graph isomorphism is not AC^0-reducible to group isomorphism. *ACM Trans. Comput. Theory*, 5(4):Art. 13, 13, 2013. Preliminary version appeared in FSTTCS '10; ECCC Tech. Report TR10-117. `doi:10.1145/2540088`.

[24] Charles J. Colbourn and Jeffrey H. Dinitz. *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2006.

[25] Nathaniel A. Collins and Michael Levet. Count-free weisfeiler–leman and group isomorphism, 2022. `doi:10.48550/ARXIV.2212.11247`.

[26] Diane J. Cook and Lawrence B. Holder. *Mining Graph Data*. 01 2009. `doi:10.1017/CBO9780511804090`.

[27] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery. `doi:10.1145/800157.805047`.

[28] David S. Dummit and Richard M. Foote. *Abstract algebra*. Wiley, New York, 3rd ed edition, 2004.

[29] Heinz-Dieter Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer, 2 edition, 1994. doi:10.1007/978-1-4757-2355-7.

[30] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.

[31] Sergei Evdokimov and Ilia Ponomarenko. Separability number and schurity number of coherent configurations. *Electr. J. Comb.*, 7, 05 2000. doi:10.37236/1509.

[32] François Le Gall and David J. Rosenbaum. On the group and color isomorphism problems. *CoRR*, abs/1609.08253, 2016. arXiv:1609.08253.

[33] Joshua A. Grochow. Matrix isomorphism of matrix lie algebras. In *2012 IEEE 27th Conference on Computational Complexity*, pages 203–213, 2012. doi:10.1109/CCC.2012.34.

[34] Joshua A. Grochow and Michael Levet. On the parallel complexity of group isomorphism via Weisfeiler–Leman. *CoRR*, abs/2112.11487, 2022. arXiv:2112.11487.

[35] Joshua A. Grochow and Youming Qiao. Polynomial-time isomorphism test of groups that are tame extensions - (extended abstract). In *Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, pages 578–589, 2015. doi:10.1007/978-3-662-48971-0_49.

[36] Joshua A. Grochow and Youming Qiao. Algorithms for group isomorphism via group extensions and cohomology. *SIAM J. Comput.*, 46(4):1153–1216, 2017. Preliminary version in IEEE Conference on Computational Complexity (CCC) 2014 (DOI:10.1109/CCC.2014.19). Also available as arXiv:1309.1776 [cs.DS] and ECCC Technical Report TR13-123. doi:10.1137/15M1009767.

[37] Joshua A. Grochow and Youming Qiao. Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions, 2019. doi:10.48550/ARXIV.1907.00309.

[38] Martin Grohe. Isomorphism testing for embeddable graphs through definability. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, page 63–72, New York, NY, USA, 2000. Association for Computing Machinery. doi:10.1145/335305.335313.

[39] Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *J. ACM*, 59(5), November 2012. doi:10.1145/2371656.2371662.

[40] Martin Grohe and Sandra Kiefer. A Linear Upper Bound on the Weisfeiler-Leman Dimension of Graphs of Bounded Genus. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP*

2019), volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 117:1–117:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.117.

[41] Martin Grohe and Sandra Kiefer. Logarithmic Weisfeiler-Leman Identifies All Planar Graphs. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 134:1–134:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.134.

[42] Martin Grohe and Julian Mariño. Definability and descriptive complexity on databases of bounded tree-width. In Catriel Beeri and Peter Buneman, editors, *Database Theory — ICDT'99*, pages 70–82, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[43] Martin Grohe and Daniel Neuen. Canonisation and definability for graphs of bounded rank width. *CoRR*, abs/1901.10330, 2019. arXiv:1901.10330.

[44] Martin Grohe, Daniel Neuen, and Pascal Schweitzer. Towards faster isomorphism tests for bounded-degree graphs. *CoRR*, abs/1802.04659, 2018. arXiv:1802.04659.

[45] Martin Grohe and Oleg Verbitsky. Testing graph isomorphism in parallel by playing a game. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I*, volume 4051 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2006. doi:10.1007/11786986_2.

[46] Xiaoyu He and Youming Qiao. On the Baer–Lovász–Tutte construction of groups from graphs: Isomorphism types and homomorphism notions. *Eur. J. Combin.*, 98:103404, 2021. doi:10.1016/j.ejc.2021.103404.

[47] Harald Andrés Helfgott, Jitendra Bajpai, and Daniele Dona. Graph isomorphisms in quasi-polynomial time, 2017. doi:10.48550/ARXIV.1710.04574.

[48] Lauri Hella. Definability hierarchies of generalized quantifiers. *Annals of Pure and Applied Logic*, 43(3):235 – 271, 1989. doi:10.1016/0168-0072(89)90070-5.

[49] Lauri Hella. Logical hierarchies in PTIME. *Information and Computation*, 129(1):1–19, 1996. doi:10.1006/inco.1996.0070.

[50] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brian. *Handbook of Computational Group theory*. CRC Press, 2020.

[51] Neil Immerman and Eric Lander. Describing graphs: A first-order approach to graph canonization. In Alan L. Selman, editor, *Complexity Theory Retrospective: In Honor of Juris Hartmanis on the Occasion of His Sixtieth Birthday, July 5, 1988*, pages 59–81. Springer New York, New York, NY, 1990. doi:10.1007/978-1-4612-4478-3_5.

[52] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

[53] Jeffrey C. Jackson, Adam R. Klivans, and Rocco A. Servedio. Learnability beyond AC^0. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 776–784, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/509907.510018.

[54] Zhengfeng Ji, Youming Qiao, Fang Song, and Aaram Yun. General linear group action on tensors: A candidate for post-quantum cryptography. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography*, pages 251–281, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-36030-6_11.

[55] Sandra Kiefer and Daniel Neuen. The power of the Weisfeiler–Leman algorithm to decompose graphs. *SIAM Journal on Discrete Mathematics*, 36(1):252–298, 2022. arXiv:10.1137/20M1314987, doi:10.1137/20M1314987.

[56] Sandra Kiefer, Ilia Ponomarenko, and Pascal Schweitzer. The weisfeiler-leman dimension of planar graphs is at most 3. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, jun 2017. doi:10.1109/lics.2017.8005107.

[57] Ludek Kucera. Canonical labeling of regular graphs in linear average time. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 271–279, 1987. doi:10.1109/SFCS.1987.11.

[58] Johannes Köbler, Uwe Schöning, and Jacobo Torán. Graph isomorphism is low for pp. volume 577, 02 1992. doi:10.1007/3-540-55210-3_200.

[59] Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, January 1975. doi:10.1145/321864.321877.

[60] François Le Gall. Efficient isomorphism testing for a class of group extensions. In *Proc. 26th STACS*, pages 625–636, 2009. doi:10.4230/LIPIcs.STACS.2009.1830.

[61] Michael Levet, Puck Rombach, and Nicholas Sieger. Logarithmic weisfeiler–leman and treewidth, 2023. arXiv:2303.07985.

[62] Mark L. Lewis and James B. Wilson. Isomorphism in expanding families of indistinguishable groups. *Groups - Complexity - Cryptology*, 4(1):73–110, 2012. doi:10.1515/gcc-2012-0008.

[63] Yinan Li and Youming Qiao. Linear algebraic analogues of the graph isomorphism problem and the erdős-rényi model. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 463–474, 2017. doi:10.1109/FOCS.2017.49.

[64] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004. doi:10.1007/978-3-662-07003-1_1.

[65] R. J. Lipton, L. Snyder, and Y. Zalcstein. The complexity of word and isomorphism problems for finite groups. Yale University Dept. of Computer Science Research Report # 91, 1977.

[66] Eugene Luks. Permutation groups and polynomial-time computation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 11, 09 1993. doi:10.1090/dimacs/011/11.

[67] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.

[68] Alan H. Mekler. Stability of nilpotent groups of class 2 and prime exponent. *The Journal of Symbolic Logic*, 46(4):781–788, 1981.

[69] Gary L. Miller. On the $n^{\log n}$ isomorphism technique (a preliminary report). In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 51–58, New York, NY, USA, 1978. Association for Computing Machinery. doi:10.1145/800133.804331.

[70] Daniel Neuen and Pascal Schweitzer. An exponential lower bound for individualization-refinement algorithms for graph isomorphism. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 138–150. ACM, 2018. doi:10.1145/3188745.3188900.

[71] Youming Qiao, Jayalal M. N. Sarma, and Bangsheng Tang. On isomorphism testing of groups with normal Hall subgroups. In *Proc. 28th STACS*, pages 567–578, 2011. doi:10.4230/LIPIcs.STACS.2011.567.

[72] David J. Rosenbaum. Breaking the $\mathrm{n}^{\log \mathrm{n}}$ barrier for solvable-group isomorphism. *CoRR*, abs/1205.0642, 2012. arXiv:1205.0642.

[73] David J. Rosenbaum. Bidirectional collision detection and faster deterministic isomorphism testing. arXiv:1304.3935 [cs.DS], 2013.

[74] Benjamin Rossman. Ehrenfeucht-Fraïssé Games on Random Structures. In Hiroakira Ono, Makoto Kanazawa, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*, volume 5514 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2009. doi:10.1007/978-3-642-02261-6_28.

[75] Uwe Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312 – 323, 1988. doi:10.1016/0022-0000(88)90010-4.

[76] Bangsheng Tang. *Towards Understanding Satisfiability, Group Isomorphism and Their Connections.* PhD thesis, Tsinghua University, 2013.

[77] G Tinhofer. Graph isomorphism and theorems of birkhoff type. *Computing*, 36:285–300, 1986. doi:10.1007/BF02240204.

[78] B.A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.

[79] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach.* Springer Verlag, 1999.

[80] Fabian Wagner. On the complexity of group isomorphism. *Electron. Colloquium Comput. Complex.*, TR11, 2011.

[81] B. Yu. Weisfeiler and A. A. Leman. Reduction of a graph to a canonical form and an algebra arising during this reduction, 1968. English translation available at https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf.

[82] Boris Weisfeiler. On construction and identification of graphs. 1976.