On the Parallel Complexity of Group Isomorphism

Michael Levet
Department of Computer Science, University of Colorado Boulder, USA

Abstract

In this paper, we exhibit efficient parallel isomorphism tests for several families of groups, including groups with $O(1)$ generators, Hamiltonian groups, and a common generalization of these (direct products of Abelian groups, with non-Abelian groups with $O(1)$ generators). In order to accomplish this, we use the L marked isomorphism procedure of Tang (ibid., 2013) to extend the parallel Weisfeiler–Leman implementation of Grohe & Verbitsky (ICALP 2006) from graphs to the setting of groups. We next extend the $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ isomorphism test for Abelian groups of Chattopadhyay, Torán, and Wagner (*ACM Trans. Comput. Theory*, 2013), to the setting of both Hamiltonian groups and groups of the form $G = A \times B$, where A is Abelian, B is a non-Abelian group of bounded order, and $Z(G)$ is a direct product of elementary Abelian groups. We also combine this $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ isomorphism test, Tang's L marked isomorphism procedure, and the NC^2 procedure to compute the Smith Normal Form due to Villard (*Appl. Alg. in Eng., Comm. and Comp.*, 1997), to obtain an NC^2 isomorphism test for groups of the form $G = A \times B$, where A is Abelian, B is a non-Abelian group with $O(1)$ generators, and $Z(G)$ is a direct product of elementary Abelian groups.

In the process of trying to find effective parallel marked isomorphism tests, we also adapted the $\beta_2\mathsf{L} \cap \beta_2\mathsf{FOLL}$ Quasigroup Isomorphism test of Chattopadhyay, Torán, and Wagner to solve the Latin Square Isotopy problem in $\beta_2\mathsf{L} \cap \beta_2\mathsf{FOLL}$. We obtain as a corollary that the Latin Square Graph Isomorphism problem is in $\beta_2\mathsf{L}$. These results improve upon the previous bound of $\beta_2\mathsf{NC}^2$ from Wolf (*Theoret. Comput. Sci.*, 1994).

2012 ACM Subject Classification Theory of computation → Parallel algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Group Isomorphism, Isomorphism Testing, Weisfeiler–Leman, Descriptive Complexity, Circuit Complexity

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Michael Levet*: This research was partially supported by Joshua A. Grochow's startup funds.

Acknowledgements I wish to thank Josh Grochow for helpful discussions and feedback on this paper. I also wish to thank Keith Kearnes for his insights on infinitary logics, which led me to a better understanding of the Hella style pebble game. I wish to thank Richard Lipton for helpful discussions regarding his previous results on Group Isomorphism.

1 Introduction

The Group Isomorphism problem (GpI) takes as input two finite groups G and H, and asks if there exists an isomorphism $\varphi : G \to H$. When the groups are given by their multiplication (a.k.a. Cayley) tables, it is known that GpI belongs to $\mathsf{NP} \cap \mathsf{coAM}$. The generator-enumerator algorithm, attributed to Tarjan in 1978 [45], has time complexity $n^{\log_p(n)+O(1)}$, where n is the order of the group and p is the smallest prime dividing n. In more than 40 years, this bound has escaped largely unscathed. The best upper bound to date on GpI is $n^{(1/4)\log_p(n)+O(1)}$, due to Rosenbaum in 2013 [48] (see [37, Sec. 2.2]). Even the impressive body of work on practical algorithms for this problem, led by Eick, Holt, Leedham-Green and O'Brien (e.g., [11, 22, 10, 17]) still results in an $n^{\Theta(\log n)}$-time algorithm in the general case (see [58, Page 2]).

In practice, such as working with computer algebra systems, the Cayley model is highly unrealistic. Instead, the groups are given by generators as permutations or matrices, or as black-boxes. For these models, GpI belongs to PromiseΣ_2^p [3]; it remains open as to whether GpI belongs to NP or coNP in such succinct models. In the past several years, there have been significant advances on algorithms with worst-case guarantees on the serial runtime for special cases of this problem [32, 55, 50, 36, 5, 47, 49, 38, 7, 6, 48, 14, 24, 21, 20].

The Group Isomorphism problem is closely related to the Graph Isomorphism problem (GI). In the Cayley (verbose) model, GpI reduces to GI [41], while GI reduces to the succinct GpI problem [28, 44]. In light of Babai's breakthrough result that GI is quasipolynomial-time solvable [4], GpI in the Cayley model is a key barrier for resolving GI. Both verbose GpI and GI are considered to be candidate NP-intermediate problems; that is, problems that belong to NP, but are neither in P nor NP-complete [35]. There is considerable evidence suggesting that GI is not NP-complete [51, 15, 31, 4, 33, 2]. As verbose GpI reduces to GI, this evidence also suggests that GpI is not NP-complete. It is also known that GI is strictly harder than GpI under AC^0 reductions [19]. Torán showed that GI is DET-hard [53], which provides that Parity is AC^0-reducible to GI. On the other hand, Chattopadhyay, Torán, and Wagner showed that Parity is not AC^0-reducible to GpI [19]. To the best of our knowledge, there is no literature on lower bounds for GpI.

In contrast with the work on serial runtime complexity, the literature on the space and parallel complexity for GpI is quite minimal. Around the same time as Tarjan's $n^{\log_p(n)+O(1)}$-time algorithm for GpI [45], Lipton, Snyder, and Zalcstein showed that $\mathsf{GpI} \in \mathsf{SPACE}(\log^2(n))$ [39]. Wolf improved the upper bound from $\mathsf{SPACE}(\log^2(n))$ to $\beta_2\mathsf{NC}^2$ (NC^2 circuits that receive $O(\log^2(n))$ non-deterministic bits as input) [59]. In 2010, Chattopadhyay, Torán, and Wagner improved this bound to $\beta_2\mathsf{L} \cap \beta_2\mathsf{FOLL}$. Here, FOLL is the class of languages that are decided by (uniform) AC circuits of depth $O(\log(\log(n))$ [9]. In 2013, Tang showed that $\mathsf{GpI} \in \beta_2\mathsf{NSC}^2$, where NSC^k is the set of languages decidable by a non-deterministic Turing Machine in polynomial time using $O(\log^k(n))$ space [52]. In the case of Abelian groups, Chattopadhyay, Torán, and Wagner showed that $\mathsf{GpI} \in \mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ [19]. To the best of our knowledge, no other specific family of groups is known to have an efficient parallel isomorphism test prior to our paper.

In light of the fact that GI lies between verbose and succinct GpI, it is natural to ask whether techniques that have been useful for GI can be adapted to GpI. The k-dimensional Weisfeiler–Leman algorithm [57] is one such key subroutine in isomorphism testing for graphs. It works by iteratively coloring k-tuples of the vertices in an isomorphism-invariant manner, where the coloring at each subsequent iteration is updated to account for a given k-tuple's current color and the colors of nearby k-tuples, where distance is measured using the Hamming metric. The k-WL procedure serves as a polynomial-time computable and useful invariant for graph non-isomorphism, with runtime $O(n^{k+1}\log(n))$, where n is the number of vertices. For any fixed k, k-WL only distinguishes certain pairs of non-isomorphic graphs, though not all of them. This motivates the notion of the Weisfeiler–Leman dimension of a graph G: the smallest k such that k-WL distinguishes G from any non-isomorphic graph. An important class of graphs that have bounded Weisfeiler–Leman dimension includes graphs with forbidden minors [26]. However, Cai, Fürer, and Immerman [16] constructed an infinite family of graphs for which the Weisfeiler–Leman dimension was linear in the number of vertices. This rules out the Weisfeiler–Leman algorithm as a general polynomial-time isomorphism test for graphs. Brachter & Schweitzer [12] introduced an adaptation of the Weisfeiler–Leman algorithm for groups, which colors a k-tuple $(g_1, \ldots, g_k)$ of group elements according to the (marked) isomorphism type of the subgroup $\langle g_1, \ldots, g_k\rangle$. The coloring is then refined in the same manner as for graphs.

Main Results. In this paper, use the L marked isomorphism test of Tang [52, Proposition 6.1] to extend the parallel Weisfeiler–Leman implementation of Grohe & Verbitsky [27] to the setting of groups. We note that marked isomorphism of graphs can be computed in AC^0. However, in the setting of groups, the number of elements in a subgroup may be exponential in the number of generators. So it is not immediately clear how to test marked isomorphism of k-tuples in a group efficiently in parallel. Precisely, Tang's marked isomorphism test allows us to compute the initial coloring of Weisfeiler–Leman in L. We then apply the color-refinement circuit of Grohe & Verbitsky [27] to obtain a parallel WL implementation for groups. In the process of implementing the initial coloring of Weisfeiler–Leman, we make the following observation.

▶ **Observation 1.** *For groups with $O(1)$ generators, $\mathsf{GpI} \in \mathsf{L}$.*

Using this parallel WL implementation, we obtain L isomorphism tests for both Abelian and Hamiltonian groups. While the L bound for Abelian groups was previously known (in fact, Chattopadhyay, Torán, and Wagner had previously shown that GpI for Abelian groups belongs to $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ [19]), the fact that Weisfeiler–Leman achieves this bound is new. For Hamiltonian groups, the previous best known bound was P (precisely, a linear-time serial algorithm) due to Das & Sharma [20].

▶ **Theorem 2.** *For both Abelian and Hamiltonian groups, the first two rounds of the 2-WL algorithm serve as an L isomorphism test.*

In our next result, we extend the $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ isomorphism test for Abelian groups of Chattopadhyay, Torán, and Wagner [19] to the setting of Hamiltonian groups. In addition to providing a second proof that GpI for Hamiltonian groups is in L, it also provides a $\mathsf{TC}^0(\mathsf{FOLL})$ upper bound.

▶ **Theorem 3.** *For the class of Hamiltonian groups, $\mathsf{GpI} \in \mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$.*

We next consider groups of the form $G = A \times B$, where A is an Abelian group, B is a group that is bounded in some way, and $Z(G)$ is the direct product of elementary Abelian groups. If B is Abelian, then so is G. In this case, we apply the $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ isomorphism test for Abelian groups due to Chattopadhyay, Torán, and Wagner [19]. We may test in AC^0 whether G is Abelian. So without loss of generality, we assume that B is non-Abelian.

In order to test G, we use the $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ of Chattopadhyay, Torán, and Wagner [19] to analyze $Z(G)$ and the generator enumeration algorithm to analyze a candidate for B (which we call C). As $Z(G)$ is the direct product of elementary Abelian groups, we need only determine a candidate for B that has a direct complement. We need not select *the* group B as in the direct decomposition $A \times B$. If B is of bounded order, we may test isomorphism using a constant size (and therefore, constant depth) circuit. We then test whether $Z(G)$ and C indeed generate G.

If instead B is only assumed to have $O(1)$ generators, we use the L marked isomorphism test of Tang [52] to analyze C. Once we have identified C, we compute the Smith Normal Form of G/C and test whether $G/C = Z(G)/Z(C)$ in NC^2 [56]. If $G/C = Z(G)/Z(C)$, then as $Z(G)$ is the direct product of elementary Abelian groups, we have identified an Abelian direct complement for C. We record this isomorphism test in the following theorem.

▶ **Theorem 4.** *Suppose that $G = A_1 \times B_1$ and $H = A_2 \times B_2$, where A_1 and A_2 are Abelian, and B_1 and B_2 are non-Abelian. Furthermore, suppose that $Z(G)$ and $Z(H)$ are direct products of elementary Abelian groups.*

(a) *Fix $d \in \mathbb{N}$. If B_1 and B_2 have order at most d, then we may decide in $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ whether $G \cong H$.*

(b) *Suppose that B_1 and B_2 have generating sets of size $O(1)$. Then we may decide in NC^2 whether $G \cong H$.*

Method. Along the way to proving Theorem 2, we adapt the counting variant of Weisfeiler–Leman for groups and the corresponding counting logics introduced by Brachter & Schweitzer [12] to the count-free setting. We then adapt the $\mathcal{C}_k$ (counting) and $\mathcal{L}_k$ (count-free) pebbling games from Cai–Fürer–Immerman [16] to the setting of groups. Next, we establish the equivalence of the count-free variants of the Weisfeiler–Leman algorithm, our adaptation of the CFI $\mathcal{L}_k$ pebbling game to the setting of groups, and the first-order logics of group theory without counting quantifiers (in analogy with the results of Cai–Fürer–Immerman for graphs). Similarly, we establish the equivalence of the counting variants as well.

With regards to the both the $\mathcal{C}_{k+1}$ and $\mathcal{L}_{k+1}$ pebble games, we show that if $\chi_r(\overline{g}) = \chi_r(\overline{h})$, then Spoiler (who seeks to establish that G and H are not isomorphic) cannot win the pebble game in r rounds starting from the pebbled configuration $(\overline{g}, \overline{h})$ (that is, $\overline{g}$ pebbled by Spoiler and $\overline{h}$ by Duplicator). In establishing equivalence with the $\mathcal{C}_k$ logics and the $\mathcal{C}_k$ pebble game, we effectively compare the sizes of the color classes of G^k and H^k after a given iteration. Intuitively, the logics with counting quantifiers effectively express the combinatorial properties of these groups. In contrast, when establishing the equivalence of the $\mathcal{L}_k$ logics and the $\mathcal{L}_k$ pebble game, we effectively compare which color classes of G and H are present in a given iteration. So in some sense, the k-dimensional counting Weisfeiler–Leman algorithm computes the combinatorial properties of a given group that can be expressed using $(k+1)$ variables in the first order logic with counting quantifiers, while the count-free k-WL algorithm only detects which group properties are present without considering multiplicity.

Bonus Result. In the process of trying to find effective parallel marked isomorphism tests, we came upon a way to adapt the $\beta_2\mathsf{L} \cap \beta_2\mathsf{FOLL}$ Quasigroup Isomorphism test of Chattopadhyay, Torán, and Wagner [19] to solve the Latin Square Isotopy problem. This improves upon the previous bound of $\beta_2\mathsf{NC}^2$ from Wolf [59]. An *isotopy* of Latin squares (quasigroups) L_1 and L_2 is a triple (α, β, γ) of bijections $\alpha, \beta, \gamma : L_1 \to L_2$ such that whenever $ab = c$, $\alpha(a)\beta(b) = \gamma(c)$. Albert showed that a quasigroup Q is isotopic to a group G if and only if Q is isomorphic to G. So in the setting of groups, isomorphism and isotopy are equivalent conditions. However, isotopic quasigroups need not be isomorphic [1].

► **Theorem 5.** *The Latin Square Isotopy problem is in $\beta_2\mathsf{L} \cap \beta_2\mathsf{FOLL}$.*

Miller showed that Latin Square Isotopy problem reduces to isomorphism testing of a family of graphs, which are referred to as *Latin square graphs.* Given a Latin square L, a Latin square graph $G(L)$ has n^2 nodes g_{ij}, where $i, j \in [n]$. Here, g_{ij} corresponds to L_{ij}, the ij entry in L. Two vertices g_{ij} and $g_{k\ell}$ are adjacent if one of the following holds: (i) $i = k$, (ii) $j = \ell$, or (iii) $L_{ij} = L_{k\ell}$. By using this reduction, Miller obtained an $n^{\log(n)+O(1)}$ time algorithm for the Latin Square Graph Isomorphism problem [45]. Wolf improved this bound to $\beta_2\mathsf{NC}^2$, which (to the best of our knowledge) is the best known upper bound [59]. In the process, Wolf showed (see [59, Lemma 4.11]) that Latin squares can be recovered from Latin square graphs in NC^1. In light of Theorem 22 and Wolf's result, we obtain the following corollary.

► **Corollary 6.** *The Latin Square Graph Isomorphism problem is in $\beta_2\mathsf{L}$.*

Further Related Work. The *marked isomorphism* problem accepts two k-tuples of group elements $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$, and asks whether the map $g_i \mapsto h_i$ for all $i \in [k]$ extends to an isomorphism of $\langle g_1, \ldots, g_k \rangle$ and $\langle h_1, \ldots, h_k \rangle$. Testing for marked isomorphism is a key component of almost every general isomorphism test for groups that uses the generator enumeration strategy [45, 39, 59, 19, 52], as well as the Weisfeiler–Leman algorithm for groups, recently introduced by Brachter & Schweitzer [12]. The best known upper bound marked isomorphism is L, due to Tang [52]. In the special case of cube generating sequences (those where every group element can be represented as $g_1^{e_1} g_2^{e_2} \cdots g_k^{e_k}$ with all $e_i \in \{0, \pm 1\}$), Chattopadhyay, Torán, and Wagner showed that marked isomorphism could be tested in $\mathsf{L} \cap \mathsf{FOLL}$ [19].

To the best of our knowledge, WL was first utilized for group isomorphism testing by Brooksbank, Grochow, Li, Qiao, and Wilson [13]. Here, the authors encoded the group structure into a hypergraph data structure. The Weisfeiler–Leman algorithm for graphs was then applied to the hypergraph structure, and the resulting coloring was then utilized to reason about the group in question. The relationship between this and the group WL of Brachter & Schweitzer [12] remains an intriguing question.

Examining the distinguishing power of $\mathcal{C}_k$ serves as a measure of descriptive complexity for groups. In the setting of graphs, the descriptive complexity has been extensively studied, with [25] serving as a key reference in this area. There has been recent work relating first order logics and groups [46, 42], as well as work examining the descriptive complexity of finite abelian groups [23]. However, the work on the descriptive complexity of groups is scant compared to the algorithmic literature on GpI.

2 Preliminaries

2.1 Complexity Classes

We assume familiarity with the complexity classes $\mathsf{P}, \mathsf{NP}, \mathsf{L}, \mathsf{NL}, \mathsf{NC}^k, \mathsf{AC}^k$, and TC^k. Throughout the paper, we also consider the following complexity classes.

- SAC^k, the set of languages decidable by uniform circuit families using AND, OR, and NOT gates, where each circuit has depth $O(\log^k(n))$ and polynomial size. Here, the AND gates have fan-in 2 and the OR gates have unbounded fan-in.
- FOLL, the set of languages decidable by uniform circuit families with AND, OR, and NOT gates of depth $O(\log(\log(n)))$, polynomial size, and unbounded fan-in. It is known that $\mathsf{AC}^0 \subsetneq \mathsf{FOLL} \subsetneq \mathsf{AC}^1$, and it is open as to whether FOLL is contained in NL [9].

For a complexity class $\mathcal{C} \in \{\mathsf{NC}^k, \mathsf{AC}^k, \mathsf{L}, \mathsf{NL}, \mathsf{FOLL}\}$, $\beta_i \mathcal{C}$ denotes class of languages recognized by a (uniform) family of $\mathcal{C}$ circuits with $O(\log^i(n))$ non-deterministic input bits.

2.2 Weisfeiler–Leman

We begin by recalling both the counting and count-free variants of the Weisfeiler–Leman algorithm in the setting of graphs, which computes an isomorphism-invariant coloring. Let Γ_1, Γ_2 be graphs, and let $k \geq 2$ be an integer. The k-dimensional Weisfeiler–Leman algorithm (k-WL) begins by constructing an initial coloring $\chi_0 : V(\Gamma_1)^k \cup V(\Gamma_2)^k \to \mathcal{K}$, where $\mathcal{K}$ is our set of colors, by assigning each k-tuple a color based on its isomorphism type. Precisely, two k-tuples $(v_1, \ldots, v_k), (u_1, \ldots, u_k)$ receive the same color under χ_0 precisely if the map $v_i \mapsto u_i$ for all $i \in [k]$ induces an isomorphism of the induced subgraphs $\Gamma_1[\{v_1, \ldots, v_k\}] \cong \Gamma_2[\{u_1, \ldots, u_k\}]$ and for all $i, j \in [k]$, $v_i = v_j \iff u_i = u_j$. Note that the latter condition is necessary to ensure that, if the k-tuples have repeated vertices, the map $v_i \mapsto u_i$ is a well-defined bijection on the induced subgraphs.

For $r \geq 0$, the coloring computed at the rth iteration of WL, denoted χ_r, is refined as follows. Fix $i \in [2]$. For a k-tuple $\overline{v} = (v_1, \ldots, v_k) \in V(\Gamma_i)^k$ and a vertex $x \in V(\Gamma_i)$, define

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

The coloring χ_{r+1} computed at the $(r+1)$st iteration of WL stores the color of the given k-tuple $\overline{v} \in V(\Gamma_i)^k$, as well as the colors under χ_r of the k-tuples obtained by substituting a single element in $\overline{v}$ for another vertex $x \in V(\Gamma_i)$. The *counting* variant of WL examines the multiset of colors over all such vertices x. The *count-free* variant of WL only stores the set of colors rather than the full multiset.

Note that the coloring χ_r computed at the rth iteration induces a partition of both $V(\Gamma_1)^k$ and $V(\Gamma_2)^k$ into color classes. The Weisfeiler–Leman algorithm terminates when the partition is not refined; that is, if when the partition induced by χ_{r+1} is identical to the partition induced by χ_r. The final coloring is referred to as the *stable coloring*, denoted $\chi_\infty := \chi_r$.

We consider the analogous counting variant of WL for groups, proposed by Brachter & Schweitzer [12]. Again, let $k \geq 2$. Let G and H be finite groups of order n. Two k-tuples $(g_1, \ldots, g_k), (h_1, \ldots, h_k) \in G^k \cup H^k$ receive the same initial coloring precisely if the map $g_i \mapsto h_i$ for all $i \in [k]$ induces an isomorphism of $\langle g_1, \ldots, g_k \rangle \cong \langle h_1, \ldots, h_k \rangle$ and for all $i, j \in [k]$, $g_i = g_j \iff h_i = h_j$. The coloring at subsequent iterations is then refined in the same manner as for graphs.

▶ Remark 7. Brachter & Schweitzer [12] only considered the counting variant of WL for groups, not the count-free version.

2.3 Logics

We recall a first-order language of group theory $\mathcal{L}$ from [12]. This first-order language is built using the variables $x_1, x_2, \ldots$, the logical connectives $\wedge, \vee, \neg, \implies$, and the quantifiers $\forall$ and $\exists$. Consider the set of variables $S = \{x_{i_1}, \ldots, x_{i_j}\}$ and a word $w \in (S \cup S^{-1})^*$. The relation $R(x_{i_1}, \ldots, x_{i_j}; w)$ holds precisely if multiplying the elements according to w yields the trivial element in the group.

Denote $\mathcal{L}_k$ to be the set of formulas from $\mathcal{L}$, where only the variables $x_1, \ldots, x_k$ are used, though the variables may be requantified. For $m \in \mathbb{N}$, denote $\mathcal{L}_{k,m}$ to be the subset of $\mathcal{L}_k$ where all the formulas have quantifier depth at most m, sometimes referred to as *quantifier rank.* Note that quantifier depth counts the total number of nested quantifiers, and not just quantifier alternations.

We denote $\mathcal{C}$ to be the extension of $\mathcal{L}$ with the counting quantifiers $\exists N$ and $\exists! N$, where $N \in \mathbb{N}$. Here, $(\exists N\, x_i)\varphi$ indicates that there exist at least N elements x_i that satisfy φ. The expression $(\exists! N\, x_i)\varphi$ indicates that there exist *exactly* N elements x_i that satisfy φ. Denote $\mathcal{C}_k$ to be the restriction of $\mathcal{C}$, where only the variables $x_1, \ldots, x_k$ are used, though the variables may be requantified. Similarly, for $m \in \mathbb{N}$, $\mathcal{C}_{k,m}$ denotes the subset of $\mathcal{C}_k$ where all the formulas have quantifier depth at most m.

We next introduce the notion of distinguishing and defining sentences.

▶ **Definition 8.** *Let Φ be a sentence from either $\mathcal{L}$ or $\mathcal{C}$. We say that Φ* distinguishes *a group G from a group H if $G \models \Phi$, but $H \not\models \Phi$. We say that Φ* defines *the group G if $G \models \Phi$, and for every group $H \not\cong G$, $H \not\models \Phi$.*

▶ **Definition 9.** *Let G and H be groups. Denote $D(G, H)$ as the minimum quantifier depth of a logical sentence in $\mathcal{L}$ that distinguishes G and H. The logical depth of a group G, denoted $D(G)$, is the minimum quantifier depth over all such formulas from $\mathcal{L}$ that define G.*

We similarly define $D^k(G, H)$ to be the minimum quantifier depth of a logical sentence in $\mathcal{L}_k$ that distinguishes G and H, and $D^k(G)$ to be the minimum quantifier depth of a logical sentence in $\mathcal{L}_k$ that defines G. If $\mathcal{L}_k$ is too weak to define G (respectively, distinguish G and H), we define $D^k(G) := \infty$ (respectively, $D^k(G, H) = \infty$).

The analogues over $\mathcal{C}$ are denoted $CountD(G)$, $CountD^k(G)$, $CountD(G, H)$, and $CountD^k(G, H)$.

▶ Remark 10. As there are only finitely many pairwise inequivalent first order sentences about finite groups with k variables and quantifier depth at most r, we have that $D^k(G) = \max\{D^k(G, H) : H \not\cong G\}$. Similarly, we have that $\text{Count}D^k(G) = \max\{\text{Count}D^k(G, H) : H \not\cong G\}$ [27].

3 Weisfeiler–Leman for Groups

3.1 Marked Isomorphism Testing

Let G and H be finite groups of order n. Now let $(g_1, \dots, g_k) \in G^k$ and $(h_1, \dots, h_k) \in H^k$. In this section, we consider two problems. The first problem is the *marked isomorphism* problem, which asks whether the map $g_i \mapsto h_i$ for all $i \in [k]$ extends to an isomorphism of $\langle g_1, \dots, g_k \rangle$ and $\langle h_1, \dots, h_k \rangle$. For the second problem, we ask whether $G = \langle g_1, \dots, g_k \rangle$. Both of these are key subroutines in our later isomorphism tests. In particular, marked isomorphism testing is the key subroutine in computing the initial coloring of Weisfeiler–Leman. Tang showed that both of these problems are in L [52]. We recall Tang's result here.

▶ **Theorem 11** (Tang [52, Proposition 6.1])**.** *Let G and H be groups of order n, and let $\overline{g} = (g_1, \dots, g_k) \in G^k$ and $\overline{h} = (h_1, \dots, h_k)$. The following problem is in L: decide whether the map $g_i \mapsto h_i$ for all $i \in [k]$ extends to an isomorphism of $\langle g_1, \dots, g_k \rangle$ and $\langle h_1, \dots, h_k \rangle$. We may also decide in L whether $G = \langle g_1, \dots, g_k \rangle$ and $H = \langle h_1, \dots, h_k \rangle$.*

We obtain as a corollary that isomorphism testing of groups with $O(1)$ generators is in L.

▶ **Corollary 12.** *Let G be a group with $O(1)$ generators, and let H be an arbitrary group. Testing whether $G \cong H$ is in L.*

Proof. Let $k := d(G)$, the minimum number of generators for G. For each k-element subset $\{g_1, \dots, g_k\}$ of G, we test in L whether $\langle g_1, \dots, g_k \rangle = G$, using Theorem 11. There are $\binom{n}{k}$ such subsets to examine, and so this needs only $\binom{n}{k} = \Theta(n^k)$ iterations. Storing a given subset requires $k \log(n) = O(\log(n))$ space. Thus, computing a k-element generating set for G is in L. Fix such a generating set $\{g_1^*, \dots, g_k^*\}$.

For each $(h_1, \dots, h_k) \in H^k$, we test whether $(g_1^*, \dots, g_k^*)$ and $(h_1, \dots, h_k)$ have the same marked isomorphism type. There are n^k such k-tuples, and so we require n^k iterations. Storing both $(g_1^*, \dots, g_k^*)$ and $(h_1, \dots, h_k)$ requires $2k \log(n) = O(\log(n))$ space. By Theorem 11, testing marked isomorphism is in L. So our algorithm runs using $O(\log(n))$ space. The result follows. ◀

▶ Remark 13. The proof of Corollary 12 can easily be adapted to compute the initial coloring of Weisfeiler–Leman. Namely, we compare all $\binom{2n^k}{2} \sim \Theta(n^{2k})$ pairs of k-tuples. As there are a polynomial number of k-tuples to compare, only $O(\log(n))$ space is required.

3.2 Parallelizing Weisfeiler–Leman

In this section, we extend the parallel implementation of Weisfeiler–Leman from Grohe & Verbitsky [27] to the setting of groups. We begin by recalling key relationships between Weisfeiler–Leman and the logics. Unless explicitly specified, k-WL refers to the counting version, while we will explicitly specify the count-free variant of k-WL.

▶ **Proposition 14.** *We have the following.*

(a) The r-round, k-WL algorithm identifies the group G if and only if $r \geq \text{CountD}^{k+1}(G)$.
(b) The r-round, k-WL algorithm distinguishes the groups G and H if and only if $r \geq \text{CountD}^{k+1}(G, H)$.
(c) The r-round, count-free k-WL algorithm identifies G if and only if $r \geq D^{k+1}(G)$.
(d) The r-round, count-free k-WL algorithm distinguishes the groups G and H if and only if $r \geq D^{k+1}(G, H)$.

Proposition 14(a)-(b) follow from Theorem 29. Proposition 14(c)-(d) follow from Theorem 28. Theorems 29 and 28 are proven in Appendix A.

We next show that Weisfeiler–Leman can be effectively parallelized, which extends the result of Grohe & Verbitsky [27] for graphs to the setting of groups. We note that marked isomorphism testing of graphs can be done in AC^0, and so the parallel Weisfeiler–Leman implementation for graphs requires depth $O(r)$, where r is the number of rounds. We use the L marked isomorphism test from Theorem 11 in the parallel WL implementation to compute the initial coloring, which results in circuit depth of $O(r + \log(n))$. As a result, we obtain the following.

▶ **Theorem 15.** *Let $k \geq 2$ be constant, and let $r := r(n)$ be a function where n is the order of the input groups. We have the following.*

(a) The r-round, k-WL algorithm can be implemented by a logspace uniform family of TC circuits of depth $O(r + \log(n))$ and size $O(r \cdot n^{3k})$.
(b) The r-round, count-free k-WL algorithm can be implemented by a logspace uniform family of AC circuits of depth $O(r + \log(n))$ and size $O(r \cdot n^{3k})$.

Proof. The proof is similar as in the case of graphs (see [27]). Full details are in Appendix B. ◀

We obtain the following immediate corollary to Theorem 15, which is analogous to that of Grohe & Verbitsky, in the case of graphs [27].

▶ **Corollary 16.** *Fix $k \geq 2$, and let $i \geq 1$. We have the following.*

(a) Let $\mathcal{C}$ be a class of groups where for each $G \in \mathcal{C}$, $\text{CountD}^{k+1}(G) = O(\log^i(|G|))$. Then GpI for $\mathcal{C}$ is in TC^i.
(b) Let $\mathcal{C}$ be a class of groups where for each $G \in \mathcal{C}$, $D^{k+1}(G) = O(\log^i(|G|))$. Then GpI for $\mathcal{C}$ is in AC^i.

3.3 Weisfeiler–Leman and Abelian Groups

In this section, we examine the parallel complexity of isomorphism testing for Abelian groups. We first apply Theorem 15 to show that GpI for Abelian groups is in L. While this does not improve the best known upper bound of Chattopadhyay, Torán, & Wagner that Abelian group isomorphism is in $\mathsf{L} \cap \mathsf{TC}^0(\mathsf{FOLL})$ [19], it does show that Weisfeiler–Leman is sufficiently powerful to acheive the bound of L.

▶ **Proposition 17.** *Let G be a finite Abelian group of order n, and let H be an arbitrary (not necessarily Abelian) group of order n. The first two rounds of 2-WL correctly decides whether $G \cong H$, in* L

Proof. We construct a three-variable formula distinguishing G from H. Define: $\varphi = \forall a \forall b, R(a, b; aba^{-1}b^{-1})$. Note that if H is not Abelian, $H \not\models \varphi$. We next recall the standard fact that finite Abelian groups are determined by the orders of their elements. In particular, if H is an Abelian group not isomorphic to G, then there exists a divisor $d \big| n$ such that G has more elements of order d than H. Let m_d denote the number of elements of order d in G. We define this using the formula ψ, below:

$$\psi := \exists m_d \, x_d, \left(R(x_d; y^d) \wedge \left(\bigwedge_{i=1}^{d-1} \neg R(x_d; y^i) \right) \right).$$

The conjunction $\varphi \wedge \psi$ distinguishes G from H. As H was arbitrary, it follows that $\text{Count}D^2(G) = 2$. Note that this requires us to requantify the variable x_d to be used in φ. So by Theorem 15, the counting variant 2-WL distinguishes G and H using only 2 rounds. Note that we neither need to know d nor m_d, since 2-WL captures all sentences on at most three variables. So only the existence of a distinguishing sentence is sufficient. The initial coloring is computed using in L circuit, and the coloring is refined using a TC^0 circuit (see Appendix B). As $\mathsf{TC}^0 \subsetneq \mathsf{L}$, we have that the two-round 2-WL algorithm can be implemented in L. ◀

3.4 Hamiltonian Groups

In this section, we extend the techniques for isomorphism testing of Abelian groups to the setting of Hamiltonian groups. We recall that a non-Abelian group G is said to be *Hamiltonian* if every subgroup of G is normal. We show that Hamiltonian groups are identified by 2-WL in two rounds, which provides that Weisfeiler–Leman is already sufficiently powerful to improve upon the linear-time isomorphism test of Das & Sharma [20]. We also extend the $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ isomorphism test for Abelian groups due to Chattopadhyay, Torán, and Wagner [19] to the setting of Hamiltonian groups.

Finite Hamiltonian groups have a nice characterization [18, Page 114].

▶ **Lemma 18** ([18]). *G is a finite Hamiltonian group if and only if $G \cong Q_8 \times A \times B$, where Q_8 is the Quaternion group of order 8, A is an Abelian group of odd order, and B is an elementary Abelian 2-group. Note that A or B (or both) be trivial.*

Observe that the Sylow 2-subgroup of a Hamiltonian group G is $Q_8 \times (\mathbb{Z}/2\mathbb{Z})^k$ for some $k \in \mathbb{N}$. The other Sylow subgroups of G are subgroups of the Abelian group of odd order A. It follows that G can be written as a direct product of its Sylow subgroups, and so finite Hamiltonian groups are nilpotent.

▶ **Theorem 19.** *The Weisfeiler–Leman dimension of Hamiltonian groups is 2. Consequently, deciding whether two finite Hamiltonian groups are isomorphic is in* L.

Proof. Let $G = Q_8 \times (\mathbb{Z}/2\mathbb{Z})^k \times A$ and $H = Q_8 \times (\mathbb{Z}/2\mathbb{Z})^\ell \times B$ be finite Hamiltonian groups of order n, where A and B are Abelian groups of odd order (possibly trivial). By Lemma 18, we note that the following is a complete invariant for G and H: whether they are Abelian, the number of elements of order 2, and the number of elements of each odd order. We note that whether a group is Abelian is captured by the sentence φ in Proposition 17, and we may count elements of a given order using the sentence ψ in Proposition 17.

As G and H have order n, we note that $G \cong H$ if and only if $A \cong B$. In particular, finite Abelian groups are determined by the orders of their elements. So if A and B are non-isomorphic Abelian groups of the same order, then there exists a divisor $d \big| |A|$ such that A has more elements of order d than B. We note that the elements of odd order correspond precisely to the elements of A and B, respectively. So we proceed as in Proposition 17. ◀

In their $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ isomorphism test for Abelian groups, Chattopadhyay, Torán, and Wagner relied on the fact that Abelian groups were determined by their orders [19], and that computing and comparing orders was in $\mathsf{FOLL} \cap \mathsf{L}$ [9]. We extend their result to Hamiltonian groups, which provides a new upper bound of $\mathsf{TC}^0(\mathsf{FOLL})$ and a second proof that isomorphism testing of Hamiltonian groups is in L.

▶ **Theorem 20.** *Let $G = Q_8 \times (\mathbb{Z}/2\mathbb{Z})^k \times A$ and $H = Q_8 \times (Z/2\mathbb{Z})^\ell \times B$ be finite Hamiltonian groups of order n, where A and B are Abelian groups of odd order and $k \geq 0$. We can decide whether $G \cong H$ in $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$.*

Proof. We note that $G \cong H$ if and only if $A \cong B$. By the same reasoning as in the proof of Theorem 19, it suffices to compute and compare the multiset of orders. We may do this in $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ using the isomorphism test of Chattopadhyay, Torán, and Wagner [19]. ◀

4 Groups with a Bounded Non-Abelian Direct Factor

In this section, we consider groups that can be written as a direct product $G = A \times B$, where A is an Abelian group, B is a non-Abelian group that is bounded in some way, and $Z(G)$ is the direct product of elementary Abelian groups. We consider both the cases when B is of bounded order, and when B has a bounded size generating set. When B has bounded order, we show that isomorphism testing is in $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$, which improves the upper bound of P from Das & Sharma [20] for groups where $Z(G)$ is the direct product of elementary Abelian groups. We note that the result of Das & Sharma does not make assumptions about $Z(G)$, so our result only applies to a proper subset of the groups captured by the result of Das & Sharma. Our result also extends the $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ isomorphism test of Chattopadhyay, Torán, and Wagner [19] for Abelian groups. When B is only guaranteed to have a generating set of bounded size, we show that isomorphism testing is in NC^2.

Let $G = A_1 \times B_1$ and $H = A_2 \times B_2$, where A_1 and A_2 are Abelian, and B_1 and B_2 are non-Abelian groups of bounded size (respectively, having bounded-size generating sets). Our strategy is to first test isomorphism of B_1 and B_2. To do this, we use a non-deterministic circuit accepting $O(\log(n))$ non-deterministic bits. The idea is to guess elements in $A_i \times B_i$ that correspond to generating sequences of $\{1\} \times B_i$. As we are unable to easily verify whether the generating sequences precisely generate the groups $\{1\} \times B_i$, we effectively guess candidates for B_1 and B_2 (which we call C_1 and C_2, respectively). Our goal is to find direct complements for C_1 and C_2.

We use the generator enumeration algorithm to test isomorphism of C_1 and C_2 (see Corollary 12). In the case when the non-Abelian factors are of bounded order, we may do this using a constant size (and therefore, constant depth) circuit. If the non-Abelian factors are only guaranteed to have bounded-size generating sets, then we handle the marked isomorphism testing in L using Theorem 11. If $C_1 \cong C_2$, we then compute the Smith Normal Form of $Z(G)/Z(C_1)$ and $Z(H)/Z(C_2)$ in NC^2 [56] and test for isomorphism of $Z(G)/Z(C_1) = G/C_1$ and $Z(H)/Z(C_2) = H/C_2$. We note as $Z(G)$ and $Z(H)$ are direct products of elementary Abelian groups, we may deduce that C_1 and C_2 are direct factors of G/C_1 and H/C_2 respectively.

▶ **Theorem 21.** *Suppose that $G = A_1 \times B_1$ and $H = A_2 \times B_2$, where A_1 and A_2 are Abelian, and B_1 and B_2 are non-Abelian groups. Furthermore, suppose that $Z(G)$ and $Z(H)$ are direct products of elementary Abelian groups.*

(a) Fix $d \in \mathbb{N}$. If B_1 and B_2 have order at most d, then we may decide in $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ whether $G \cong H$.
(b) Suppose that B_1 and B_2 have generating sets of size $O(1)$. Then we may decide in NC^2 whether $G \cong H$.

Proof. Full details are in Appendix C. ◀

5 Latin Square Isotopy

In this section, we show that the Latin Square Isotopy problem is in $\beta_2\mathsf{L} \cap \beta_2\mathsf{FOLL}$. Recall that a *cube generating sequence* $(s_1, \ldots, s_k)$ for a Latin square (quasigroup) is a generating sequence where each element g in the Latin square can be written as $g = s_1^{e_1} \cdots s_k^{e_k}$, for $e_1, \ldots, e_k \in \{0, 1\}$. The key technique is to guess cube generating sequences A and B for the Latin square L_1, and cube generating sequences A', B' for the Latin square L_2. We then (attempt to) construct appropriate bijections $\alpha, \beta : L_1, L_2$, where α extends the map from $A \to A'$ and β extends the map from $B \to B'$. We can construct such bijections in $\mathsf{FOLL} \cap \mathsf{L}$ using the techniques from Chattopadhyay, Torán, and Wagner.

The key step remains in checking whether the map sending the product of each pair $(x, y) \in L_1 \times L_1$, $xy \mapsto \alpha(x)\beta(y)$ a bijection. Wolf approaches this in the following manner. First, construct sets $C = \{a_ib_i : a_i \in A, b_i \in B\}$ and $C' = \{a'_ib'_i : a'_i \in A', b'_i \in B'\}$. Then check whether the map $a_ib_i \mapsto a'_ib'_i$ extends to a bijection $\gamma : L_1 \to L_2$. Finally, check whether $\alpha(x)\beta(y) = \gamma(xy)$ for all $x, y \in L_1$. We note that Wolf is able to do this in NC^2. If C and C' are cube generating sequences, then we would be able to apply the technique of Chattophadyay, Torán, and Wagner to compute the bijection γ in $\mathsf{FOLL} \cap \mathsf{L}$. However, C and C' need not be cube generating sequences. As an example, suppose that $B = A^{-1}$. Then B is a cube generating sequence. Furthermore, $a_ib_i = 1$ for each i, and so C has only the identity element. In general, we do not expect C and C' to be cube generating sequences, even if they do generate L_1 and L_2 respectively.

Instead of extending Wolf's technique, we extend Miller's technique in his second proof that Latin Square Isotopy is decidable in time $O(n^{\log(n)+O(1)})$. Miller constructs the relation $R = \{(xy, \alpha(x)\beta(y)) : x, y \in L_1\}$ and checks whether R is a bijection. If R is a bijection; then by construction, the pair (α, β, R) is an isotopy [45, Theorem 2, Proof 2]. Using the fact that A and B are cube generating sequences, we can compute x and y in $\mathsf{L} \cap \mathsf{FOLL}$. In the process of computing α and β, we obtain a data structure which allows us to compute $\alpha(x)$ and $\beta(y)$ in $\mathsf{L} \cap \mathsf{FOLL}$.

▶ **Theorem 22.** *Deciding whether two Latin squares are isotopic is in $\beta_2\mathsf{L} \cap \beta_2\mathsf{FOLL}$.*

Proof. Full details are in Appendix D. ◀

Miller [45] showed that isomorphism testing of Latin square graphs is polynomial-time reducible to the Latin square isotopy problem. Wolf [59] improved this bound, showing that isomorphism testing of Latin square graphs is NC^1 reducible to testing for Latin square isotopy. We recall Wolf's result below.

▶ **Lemma 23** (Wolf [59, Lemma 4.11]). *Let G be a Latin square graph derived from a Latin square of size n. We can retrieve a Latin square from G with a polynomial-sized NC circuit with $O(\log(n))$ depth.*

▶ Remark 24. The statement of Lemma 4.11 in Wolf actually claims a depth of $O(\log^2(n))$. However, in the proof of Lemma 4.11, Wolf shows that only $O(\log(n))$ depth is needed [59].

In light of Wolf's result and Theorem 22, we obtain the following corollary.

▶ **Corollary 25.** *Let G_1 and G_2 be Latin square graphs. Deciding whether $G_1 \cong G_2$ is in β_2L.*

6 Conclusion

In this section, we discuss several directions for further research.

Weisfeiler–Leman for Groups. The literature on the Weisfeiler–Leman algorithm for graphs is quite rich and continues to be a vibrant area of active research. Despite the fact that WL is insufficient to resolve Graph Isomorphism in polynomial time (see CFI [16]). We note as well that low-dimensional WL is a key subroutine in the state of the art algorithms for GI, both on the theoretical [4, 34, 26] and practical [43, 40] sides. In the setting of groups, little is known about the abilities of Weisfeiler–Leman to distinguish groups based on their underlying properties. Abelian groups, simple groups, and groups with a bounded number of generators are all easily distinguished by Weisfeiler–Leman. It is also the case that k-WL computes the k-subgroup profiles. It is not clear as to the power and limits of Wesifeiler–Leman for groups. To this end, we propose the following research directions.

- Determine families of groups for which Weisfeiler–Leman yields a polynomial-time isomorphism test, or even as an effective sub-routine in an isomorphism test. In light of Theorem 15, it would also be of interest to determine additional families of groups for which Weisfeiler–Leman can be effectively parallelized.
- In light of Theorem 21, we ask whether Weisfeiler–Leman would be sufficient to yield an L isomorphism test for groups with non-Abelian factors of bounded order or with bounded size generating sets. Even determining the Weisfeiler–Leman dimension would be of interest.
- Let G and H be finite groups. Relate the Weisfeiler–Leman dimension of group products, such as the direct product $G \times H$, semi-direct products $G \rtimes H$, or central product to the Weisfeiler–Leman dimension of G and H.
- Determine group properties that Weisfeiler–Leman can(not) easily distinguish.

One natural approach for the first and last questions is to examine the descriptive complexity of groups. That is, determine group properties that can be expressed using first-order logic with counting quantifiers. The number of variables required dictates the Weisfeiler–Leman dimension, while the quantifier depth determines the number of rounds for the algorithm. Controlling either would lead to insights, and hopefully improvements, on the runtime complexity of the algorithm.

The central open problem in the theory of Weisfeiler–Leman for groups remains whether there exists an absolute constant k such that k-WL correctly determines isomorphism for all pairs of groups. Note that this would provide a polynomial-time algorithm for GpI in the Cayley model. We conjecture that there exists an infinite family of groups requiring $\Omega(\log(n))$-dimensional WL to distinguish them. For such a family, WL would have runtime $n^{\Theta(\log(n))}$, which does not be the trivial generator-enumeration bound [45].

Parallel Algorithms for Groups. In the last ten years, significant progress has been made regarding serial algorithms for GpI with worst-case runtime guarantees. The strategy has been to exhibit restricted families of groups, and then to use the properties of the underlying groups to design isomorphism tests. We propose this same strategy for constructing efficient parallel algorithms for GpI. It may be worthwhile to attempt to parallelize existing efficient serial isomorphism tests for restricted families of groups. In particular, any parallelized sub-routines in these isomorphism tests may be of independent interest to the Computational Group Theory community. On the other hand, this approach may also highlight new barriers in parallelizing GpI, which may yield new insights in the P vs. NC question. This may also highlight underlying group structure that inhibits parallelization; studying such structure would be of independent interest.

It would also be of interest to improve upon Theorem 21. One direction would be to show how to remove the assumption that $Z(G)$ is the direct product of elementary Abelian groups. Even doing so in the case where the non-Abelian factor B is of bounded order would be of interest, as it would fully improve upon the result of Das & Sharma [20]. In the case when B is only assumed to have $O(1)$ generators, the key bottleneck lies in computing the Smith Normal Form. Improving upon the NC^2 barrier for computing the Smith Normal Form is a major open problem. On the other hand, it would be of interest construct an isomorphism test that does not explicitly rely on computing the Smith Normal Form.

It would also be of interest to determine lower bounds for GpI. Chattopadhyay, Torán, and Wagner [19] showed that Parity is not AC^0-reducible to GpI. So GpI is not hard for any class containing Parity, including $\mathsf{ACC}^0, \mathsf{NC}^1, \mathsf{L}$, or NL. On the other hand, computing a single group operation in the Cayley model is in AC^0 and appears not to be in NC^0. Most isomorphism tests require $\omega(1)$ iterations, where each iteration uses group products computed from the previous rounds. This seems to suggest that $\mathsf{GpI} \not\in \mathsf{AC}^0$. On the other hand, it does not formally rule out clever bit manipulations of the Cayley table.

A Equivalence of Weisfeiler–Leman, Logics, and Pebbling Games

For groups, we introduce two new pebbling games in the same flavor as those defined by Cai, Fürer, and Immerman [16], including the count-free $\mathcal{L}_k$ and counting $\mathcal{C}_k$ pebbling games. Both games are parameterized by k, the number of pebbles. Let G and H be finite groups of order n. Suppose we have k pairs of pebbles $(p_1, p_1'), \ldots, (p_k, p_k')$. Each game contains two players, Spoiler and Duplicator. Informally, Spoiler wins if they can establish by using at most k pebbles, that $G \not\cong H$. Duplicator wins otherwise. Each turn of the game proceeds as follows. Spoiler picks up a pebble pair (p_i, p_i').

- In the $\mathcal{L}_k$ game, Spoiler places the pebble p_i on an element $g \in G$. Duplicator responds by placing p_i' on an element $h \in H$.
- In the $\mathcal{C}_k$ game, Spoiler selects a set $A \subseteq G$. Duplicator responds by selecting a set $B \subseteq H$ such that $|B| = |A|$. Spoiler then selects an element $b \in B$ to pebble with p_i. Duplicator selects an element $a \in A$ to pebble with p_i'.

Let $g_{i_1}, \ldots, g_{i_\ell} \in G$ and $h_{i_1}, \ldots, h_{i_\ell}$ be the elements that are pebbled at the end of stage r, using the pebble pairs $(p_{i_1}, p_{i_1}'), \ldots, (p_{i_\ell}, p_{i_\ell}')$ (where $\ell \leq k$). Spoiler wins if the map sending $g_i \mapsto h_i$ for each $i \in \{i_1, \ldots, i_\ell\}$ does not induce an isomorphism of $\langle g_{i_1}, \ldots, g_{i_\ell} \rangle \cong \langle h_{i_1}, \ldots, h_{i_\ell} \rangle$, or if there exist $j, k \in [\ell]$ such that $g_{i_j} = g_{i_k} \iff h_{i_j} = h_{i_k}$ does not hold.

The analogues of the CFI-style $\mathcal{L}_k$ pebbling game and the $\mathcal{L}_k$ logics for graphs are known to be equivalent. Similarly, the CFI-style $\mathcal{C}_k$ pebbling game, the $\mathcal{C}_k$ logics for graphs, and Weisfeiler–Leman are also known to be equivalent [16].

A key tool in establishing these equivalences is the notion of a k-configuration, which allows us to relate the logics to pebbling games. We recall the notion of a k-configuration here.

▶ **Definition 26.** *Fix $k \in \mathbb{Z}^+$, and let G and H be groups of order n. A k-configuration is a pair of partial functions (u, v), where $u : \{x_1, \ldots, x_k\} \to G$, $v : \{x_1, \ldots, x_k\} \to H$, and the domains of u and v are the same. Denote the domain of u to be $Dom(u)$.*

Let G and H be groups of order n, and let (u, v) be a k-configuration. For a (partial) function f, we denote updating f to map $f(x_i) = g$ by $f(x_i/g)$. Let $\varphi \in \mathcal{C}_k$. We evaluate $\varphi(u)$ by substituting $\varphi(x_i/u(x_i))$ whenever $u(x_i)$ is defined. The remaining variables of φ are treated as free variables. We say that $(G, u) \vDash \varphi$ if there exists a satisfying configuration of the remaining free variables from $\varphi(u)$. If there exists $\overline{g} \in G^k$ such that $\varphi(\overline{g})$ holds, we denote that $(G, \overline{g}) \vDash \varphi$.

▶ **Definition 27.** *Let G and H be groups of order n, and let (u, v) be a k-configuration. Fix $m \in \mathbb{N}$. We say that (G, u) and (H, v) are equivalent in $\mathcal{L}_{k,m}$, denoted $(G, u) \equiv_{\mathcal{L}_{k,m}} (H, v)$, if for every $\varphi \in \mathcal{L}_{k,m}$, $(G, u) \vDash \varphi \iff (H, v) \vDash \varphi$. We say that (G, u) and (H, v) are $\mathcal{L}_k$ equivalent, denoted $(G, u) \equiv_{\mathcal{L}_k} (H, v)$, if $(G, u) \equiv_{\mathcal{L}_{k,m}} (H, v)$ for all $m \in \mathbb{N}$.*

Similarly, we say that (G, u) and (H, v) are equivalent in $\mathcal{C}_{k,m}$, denoted $(G, u) \equiv_{\mathcal{C}_{k,m}} (H, v)$, if for every $\varphi \in \mathcal{C}_{k,m}$, $(G, u) \vDash \varphi \iff (H, v) \vDash \varphi$. We say that (G, u) and (H, v) are $\mathcal{C}_k$ equivalent, denoted $(G, u) \equiv_{\mathcal{C}_k} (H, v)$, if $(G, u) \equiv_{\mathcal{C}_{k,m}} (H, v)$ for all $m \in \mathbb{N}$.

▶ **Theorem 28.** *Let G and H be groups of order n, and let $k \geq 1, r \geq 0$. Then for all k-tuples $\overline{g} \in G^k, \overline{h} \in H^k$, the following are equivalent:*

1. *$\chi_r(\overline{g}) \neq \chi_r(\overline{h})$ in the k-dimensional count-free Weisfeiler–Leman*
2. *$(G, \overline{g}) \not\equiv_{\mathcal{L}_{k+1,r}} (H, h)$*
3. *Spoiler has a winning strategy for the CFI-style $\mathcal{L}_{k+1}$ pebble game with $(k + 1)$ pebbles in r moves, starting from the k-configuration $(\overline{g}, \overline{h})$.*

Proof. The proof is similar to that for graphs [16]. Full details are in Appendix A.1. ◀

Similarly, the analogues of the counting Weisfeiler–Leman algorithm, the $\mathcal{C}$ logics, and the pebbling game for graphs are known to be equivalent [16]. We show the corresponding result for groups.

▶ **Theorem 29.** *Let G and H be groups of order n, and let $k \geq 1, r \geq 0$. Then for all k-tuples $\overline{g} \in G^k, \overline{h} \in H^k$, the following are equivalent:*

1. *$\chi_r(\overline{g}) \neq \chi_r(\overline{h})$ in k-dimensional Weisfeiler–Leman*
2. *$(G, \overline{g}) \not\equiv_{\mathcal{C}_{k+1,r}} (H, \overline{h})$*
3. *Spoiler has a winning strategy for the $(k + 1)$-pebble game in r moves starting from the k-configuration $(\overline{g}, \overline{h})$.*

Proof. The proof is similar to that for graphs in [16]; full details are in Appendix A.2. ◀

▶ Remark 30. Brachter & Schweitzer [12] established that the Hella-style $(k + 1)$-pebble game [29, 30] they used is equivalent to k–WL. This yields the following immediate corollary.

▶ **Corollary 31.** *Let G and H be groups of order n, and let $k \in \mathbb{N}$. We have that G and H are distinguishable by the CFI-style k-pebble game for groups if and only if G and H are distinguishable by the Hella-style k-pebble game for groups.*

A.1 Proofs Establishing Equivalence of Count-Free Weisfeiler–Leman, Logics, and Pebbling Game

In this section, we prove Theorem 28, establishing the equivalence of the k-dimensional count-free Weisfeiler–Leman algorithm, the $\mathcal{L}_{k+1}$ logics, and the CFI-style $\mathcal{L}_k$ pebble game for groups with $(k+1)$ pebbles. We break the proof of Theorem 28 up into a series of propositions.

We begin by establishing that if the count-free k-WL fails to distinguish the groups G and H in r iterations, then Spoiler is unable to win the $(k+1)$-pebble game in at most r moves.

▶ **Proposition 32.** *Let G and H be groups of order n, and fix k to be the dimension of the Weisfeiler–Leman algorithm. Suppose that $\overline{g} = (g_1, \ldots, g_k) \in G^k$ and $\overline{h} = (h_1, \ldots, h_k) \in H^k$ satisfy $\chi_r(\overline{g}) = \chi_r(\overline{h})$. Suppose that we have a $(k+1)$-pebbling configuration (u, v) where $u(x_i) = g_i$ and $v(x_i) = h_i$ for all $i \in [k]$. Then Spoiler cannot improve their strategy by updating at most r coordinates of u. Note that Duplicator can respond by updating their strategy.*

Proof. The proof is by induction on the number of rounds, $r \in \mathbb{N}$. Suppose that $\chi_0(\overline{g}) = \chi_0(\overline{h})$. So $\overline{g}$ has the same marked isomorphism type as $\overline{h}$. By the definition of the pebbling game, Spoiler is unable to win the $(k+1)$-pebble game without making a move. Now fix $r \geq 0$; and suppose that if $\chi_r(\overline{g}) = \chi_r(\overline{h})$, then Spoiler is unable to improve their strategy by updating at most r coordinates of u.

Now suppose that $\chi_{r+1}(\overline{g}) = \chi_{r+1}(\overline{h})$. So $\chi_r(\overline{g}) = \chi_r(\overline{h})$. By the Inductive Hypothesis, we have that for any $g^* \in G$ that differs from $\overline{g}$ in coordinates $i_1, \ldots, i_r$, there exists an $h^* \in H$ that differs from $\overline{h}$ in coordinates $i_1, \ldots, i_r$ such that the mapping $g_i^* \mapsto h_i^*$ for all $i \in [k]$ is an isomorphism of $\langle g_1^*, \ldots, g_k^* \rangle \cong \langle h_1^*, \ldots, h_k^* \rangle$ and also satisfies that for all $i, j \in [k]$, $g_i^* = g_j^* \iff h_i^* = h_j^*$. As $\overline{g}$ and $\overline{h}$ have the same marked isomorphism type, it is a weakly improving strategy for Spoiler to have updated their strategy to use $u(x_{i_j}/g_{i_j}^*)$ for all $j \in [r]$. Duplicator's strategy is appropriately updated so that $v(x_{i_j}/h_{i_j}^*)$ is used instead of h_{i_j} for all $j \in [r]$. Now as $\chi_{r+1}(\overline{g}) = \chi_{r+1}(\overline{h})$, it follows that for all $g \in G$, there exists $h \in H$ such that:

$$(\chi_0(g^*(g_1^*/g)), \ldots, \chi_0(g^*(g_k^*/g))) = (\chi_0(h^*(h_1^*/h)), \ldots, \chi_0(h^*(h_k^*/h))).$$

So $\chi_1(g^*) = \chi_1(h^*)$. Now if we have the pebbling $g_\ell^* \mapsto h_\ell^*$ for all $\ell \in [k]$, then Spoiler is unable to improve their strategy by exchanging one element of g^* for some other element of G. It follows that for the pebbling mapping $g_i \mapsto h_i$ for all $i \in [k]$, Spoiler is unable to improve their strategy by exchanging $r+1$ elements of $\overline{g}$ for other elements of G. ◀

We next show that if k-WL can distinguish G and H, then G and H are distinguishable according to $\mathcal{L}_{k+1}$. More precisely, we show that if k-WL distinguishes two k-tuples $\overline{g} \in G^k$ and $\overline{h} \in H^k$ in at most r iterations, then some sentence $\varphi \in \mathcal{L}_{k+1,r}$ where $(G, g) \vDash \varphi$ but $(H, h) \nvDash \varphi$. Before proving this result, we need the following lemmas.

▶ **Lemma 33** (Lemma 4.4, [16]). *For any relational language with finitely many relation symbols, and any k and m, there are only finitely many formulas up to equivalence in $\mathcal{L}_{k,m}$.*

▶ **Lemma 34** (Lemma 3.6, [12]). *There is a quantifier free k-variable formula $\varphi(x_1, \ldots, x_k) \in \mathcal{L}_{II}$ distinguishing the k-tuples $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$ if and only if these tuples differ in their initial coloring of k–WL.*

▶ Remark 35. Recall that we refer to the language $\mathcal{L}_{II}$ from Brachter & Schweitzer [12] as $\mathcal{C}$, in keeping with the conventions from CFI [16]. We also note that the quantifier-free formulas in $\mathcal{C}$ are precisely the quantifier-free formulas in $\mathcal{L}$.

▶ **Proposition 36.** *Let G and H be groups of order n, and let $r \in \mathbb{N}$. Suppose that $\overline{g} = (g_1, \ldots, g_k) \in G^k$ and $\overline{h} = (h_1, \ldots, h_k) \in H^k$ satisfy $\chi_r(\overline{g}) \neq \chi_r(\overline{h})$. Then $(G, g) \not\equiv_{\mathcal{L}_{k+1,r}} (H, h)$.*

Proof. The proof is by induction on r. Let $\overline{g}$ and $\overline{h}$ such that the map $g_i \mapsto h_i$ induces an isomorphism of $\langle g_1, \ldots, g_k\rangle \cong \langle h_1, \ldots, h_k\rangle$. Then by Lemma 40, $\chi_0(\overline{g}) = \chi_0(\overline{h})$ if and only if $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$ satisfy the same quantifier-free formulas in $\mathcal{L}_{k+1}$. Now fix $r \geq 0$, and suppose that $\overline{g} \in G^k$ and $\overline{h} \in H^k$ satisfy that if $\chi_r(\overline{g}) \neq \chi_r(\overline{h})$, then $(G, \overline{g}) \not\equiv_{\mathcal{L}_{k+1,r}} (H, \overline{h})$.

Suppose that $\overline{g} \in G^k$ and $\overline{h} \in H^k$ satisfy that $\chi_{r+1}(\overline{g}) \neq \chi_{r+1}(\overline{h})$. We have two cases. Suppose first that $\chi_r(\overline{g}) \neq \chi_r(\overline{h})$. Then by the inductive hypothesis, $(G, g) \not\equiv_{\mathcal{L}_{k+1,r}} (H, h)$. So $(G, g) \not\equiv_{\mathcal{L}_{k+1,r+1}} (H, h)$.

Suppose instead that $\chi_r(\overline{g}) = \chi_r(\overline{h})$. As $\chi_r(\overline{g}) = \chi_r(\overline{h})$, but $\chi_{r+1}(\overline{g}) \neq \chi_{r+1}(\overline{h})$, there exists a color t such that without loss of generality, $\overline{g}$ has a neighbor of color t, while $\overline{h}$ has no neighbors of color t. Fix a color t neighbor of $\overline{g}$, which we denote g^*. Let i denote the index in which g^* differs from $\overline{g}$. By Lemma 39, the color class t is characterized by finitely many formulas. Denote ψ_t to be conjunction of finitely many $\mathcal{L}_{k+1,r}$ formulas characterizing the color class t at iteration r of the k-dimensional Weisfeiler–Leman algorithm. So for $F \in \{G, H\}$ and $\overline{w} \in F^k$, we have that: $\chi_r(\overline{w}) = t \iff (F, \overline{w}) \vDash \psi_t$.

Denote: $\varphi := (\exists N x_{k+1})\psi_t(x_i/x_{k+1})$. We note that $(G, \overline{g}) \vDash \varphi$, but $(H, \overline{h}) \nvDash \varphi$. The result follows. ◀

We next relate the distinguishing power of $\mathcal{L}_k$ to the k-pebbling game.

▶ **Proposition 37.** *Fix $k \in \mathbb{Z}^+$. Suppose that G and H are groups of order n. Let (u, v) be a k-configuration. If $(G, u) \not\equiv_{\mathcal{L}_{k,r}} (H, v)$, then Spoiler has a winning strategy for the k-pebble game in at most r moves.*

Proof. The proof is by induction on r. Suppose that (u, v) is a k-configuration such that the map $u(x_i) \mapsto v(x_i)$ for all $x_i \in \text{Dom}(u)$, induces an isomorphism of

$$\langle\{u(x_i) : x_i \in \text{Dom}(u)\}\rangle \cong \langle\{v(x_i) : x_i \in \text{Dom}(u)\}\rangle.$$

So u and v satisfy the same quantifier-free formulas in $\mathcal{L}_k$. By the definition of the k-pebble game, Spoiler is unable to win without moving. Now fix $r \geq 0$, and suppose that for any k-configuration (u, v) such that $(G, u) \equiv_{\mathcal{L}_{k,r}} (H, v)$, Spoiler is unable to win the k-pebble game in at most r moves. Now suppose that (u, v) is a k-configuration such that: $(G, u) \not\equiv_{\mathcal{L}_{k,r+1}} (H, v)$.

If $(G, u) \not\equiv_{\mathcal{L}_{k,r}} (H, v)$, then Spoiler has a winning strategy by the inductive hypothesis. So suppose that: $(G, u) \equiv_{\mathcal{L}_{k,r}} (H, v)$. Thus, there exists $\varphi \in \mathcal{L}_{k,r+1} \setminus \mathcal{L}_{k,r}$ such that, without loss of generality, $(G, u) \vDash \varphi$ but $(H, v) \nvDash \varphi$. We note that if φ is a conjunction, disjunction, or negation smaller formulas, then (H, h) does not satisfy one of the smaller formulas. However, such a smaller formula belongs to $\mathcal{L}_{k,r}$, a contradiction. Without loss of generality, we may assume that φ is of the form $(\exists x_{k+1})\psi$, where $\psi \in \mathcal{L}_{k,r}$. We construct a winning strategy in the k-pebble game for Spoiler as follows. Spoiler begins by selecting an unpebbled element $g \in G$ that satisfies ψ. As $(H, v) \nvDash \varphi$, we have that for every $h^* \in H$, $(H, v(x_{k+1}/h^*)) \nvDash \psi$. Fix such an h^*. Denote $u_1 := u(x_{k+1}/g^*)$ and $v_1 := v(x_{k+1}/h^*)$. So (u_1, v_1) is a k-configuration such that $(G, u_1) \vDash \psi$ and $(H, v_1) \nvDash \psi$. Thus, $(G, u_1) \not\equiv_{\mathcal{L}_{k,r}} (H, v_1)$.

By the inductive hypothesis, Spoiler has a winning strategy in the remaining r moves of the k-pebbling game. The result follows. ◀

A.2 Proofs Establishing Equivalence of the Counting Weisfeiler–Leman, Logics, and Pebbling Game

In this section, we prove Theorem 29, establishing the equivalence of the k-dimensional Weisfeiler–Leman algorithm, the $\mathcal{C}_{k+1}$ logics, and the CFI style pebble game for groups with $(k+1)$ pebbles. We break the proof of Theorem 29 up into a series of propositions.

We begin by establishing that if k-WL fails to distinguish the groups G and H in r iterations, then Spoiler is unable to win the $(k+1)$-pebble game in at most r moves.

▶ **Proposition 38.** *Let G and H be groups of order n, and fix k to be the dimension of the Weisfeiler–Leman algorithm. Suppose that if $\overline{g} = (g_1, \ldots, g_k) \in G^k$ and $\overline{h} = (h_1, \ldots, h_k) \in H^k$ satisfy $\chi_r(\overline{g}) = \chi_r(\overline{h})$. Suppose that we have a $(k+1)$-pebbling configuration (u, v) where $u(x_i) = g_i$ and $v(x_i) = h_i$ for all $i \in [k]$. Then Spoiler cannot improve their strategy by updating at most r coordinates of u. Note that Duplicator can respond by updating their strategy.*

Proof. The proof is by induction on $r \in \mathbb{N}$. Suppose that $\chi_0(\overline{g}) = \chi_0(\overline{h})$. So $\overline{g}$ has the same marked isomorphism type as $\overline{h}$. By definition of the pebbling game, Spoiler is unable to win the $(k+1)$-pebble game without making a move. Now fix $r \geq 0$, and suppose that if $\chi_r(\overline{g}) = \chi_r(\overline{h})$, then Spoiler is unable to improve their strategy by updating at most r-coordinates of u.

Now suppose that $\chi_{r+1}(\overline{g}) = \chi_{r+1}(\overline{h})$. So $\chi_r(\overline{g}) = \chi_r(\overline{h})$. By the Inductive Hypothesis, we have that for any $g^* \in G^k$ that differs from $\overline{g}$ in coordinates $i_1, \ldots, i_r$, there exists $h^* \in H^k$ that differs from $\overline{h}$ in coordinates $i_1, \ldots, i_r$ such that the mapping $g_i^* \mapsto h_i^*$ is an isomorphism of $\langle g_1^*, \ldots, g_k^* \rangle \cong \langle h_1^*, \ldots, h_k^* \rangle$. As $g_i \mapsto h_i$ for all $i \in [k]$ is an isomorphism of $\langle g_1, \ldots, g_k \rangle \cong \langle h_1, \ldots, h_k \rangle$, it is a weakly improving strategy for Spoiler to have updated their strategy to use $u(x_{i_j}/g_{i_j}^*)$ instead of g_{i_j} for all $j \in [r]$. Duplicator's strategy is appropriately updated, so that $v(x_{i_j}/h_{i_j}^*)$ instead of h_{i_j} for all $j \in [r]$. Now as $\chi_{r+1}(\overline{g}) = \chi_{r+1}(\overline{h})$, it follows that there is a bijection $\varphi : G \to H$ such that:

$$(\chi_0(g^*(g_1^*/g)), \ldots, \chi_0(g^*(g_k^*/g))) = (\chi_0(h^*(h_1^*/\varphi(g))), \ldots, \chi_0(h^*(h_k^*/\varphi(g)))).$$

So $\chi_1(g^*) = \chi_1(h^*)$. Now if we have the pebbling $g_\ell^* \mapsto h_\ell^*$ for all $\ell \in [k]$, then Spoiler is unable to improve their strategy by exchanging one element of g^* for some other element of G. It follows that for the pebbling mapping $g_i \mapsto h_i$ for all $i \in [k]$, Spoiler is unable to improve their strategy by exchanging $r+1$ elements of $\overline{g}$ for other elements of G. ◀

We next show that if k-WL can distinguish G and H, then G and H are distinguishable according to $\mathcal{C}_{k+1}$. More precisely, we show that if k-WL distinguishes two k-tuples $\overline{g} \in G^k$ and $\overline{h} \in H^k$ in at most r iterations, then some sentence $\varphi \in \mathcal{C}_{k+1,r}$ where $(G, g) \vDash \varphi$ but $(H, h) \nvDash \varphi$. Before proving this result, we need the following lemmas.

▶ **Lemma 39** ([16], Lemma 4.4). *For any relational language with finitely many relation symbols, and any k and m, there are only finitely many formulas up to equivalence in $\mathcal{C}_{k,m}$.*

▶ **Lemma 40** ([12], Lemma 3.6). *There is a quantifier free k-variable formula $\varphi(x_1, \ldots, x_k) \in \mathcal{L}_{II}$ distinguishing the k-tuples $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$ if and only if these tuples differ in their initial coloring of k–WL.*

▶ Remark 41. Recall that we refer to the language $\mathcal{L}_{II}$ from Brachter & Schweitzer [12] as $\mathcal{C}$, in keeping with the conventions from CFI [16].

▶ **Proposition 42.** *Let G and H be groups of order n, and let $r \in \mathbb{N}$. Suppose that $\overline{g} = (g_1, \ldots, g_k) \in G^k$ and $\overline{h} = (h_1, \ldots, h_k) \in H^k$ satisfy $\chi_r(\overline{g}) \neq \chi_r(\overline{h})$. Then $(G, g) \not\equiv_{\mathcal{C}_{k+1,r}} (H, h)$.*

Proof. The proof is by induction on r. Let $\overline{g}$ and $\overline{h}$ such that the map $g_i \mapsto h_i$ induces an isomorphism of $\langle g_1, \ldots, g_k\rangle \cong \langle h_1, \ldots, h_k\rangle$. Then by Lemma 40, $\chi_0(\overline{g}) = \chi_0(\overline{h})$ if and only if $(g_1, \ldots, g_k)$ and $(h_1, \ldots, h_k)$ satisfy the same quantifier-free formulas in $\mathcal{C}_{k+1}$. Now fix $r \geq 0$, and suppose that $\overline{g} \in G^k$ and $\overline{h} \in H^k$ satisfy that if $\chi_r(\overline{g}) \neq \chi_r(\overline{h})$, then $(G, \overline{g}) \not\equiv_{\mathcal{C}_{k+1,r}} (H, \overline{h})$.

Suppose that $\overline{g} \in G^k$ and $\overline{h} \in H^k$ satisfy that $\chi_{r+1}(\overline{g}) \neq \chi_{r+1}(\overline{h})$. We have two cases. Suppose first that $\chi_r(\overline{g}) \neq \chi_r(\overline{h})$. Then by the inductive hypothesis, $(G, g) \not\equiv_{\mathcal{C}_{k+1,r}} (H, h)$. So $(G, g) \not\equiv_{\mathcal{C}_{k+1,r+1}} (H, h)$.

Suppose instead that $\chi_r(\overline{g}) = \chi_r(\overline{h})$. As $\chi_r(\overline{g}) = \chi_r(\overline{h})$, but $\chi_{r+1}(\overline{g}) \neq \chi_{r+1}(\overline{h})$, there exists a color t such that without loss of generality, $\overline{g}$ has more neighbors of color t than $\overline{h}$. Let N denote the number of $\overline{g}$'s neighbors of color t, and let $i_1, \ldots, i_N$ denote the indices in which the color t neighbors of $\overline{g}$ differ from $\overline{g}$. By Lemma 39, the color class t is characterized by finitely many formulas. Denote ψ_t to be conjunction of finitely many $\mathcal{C}_{k+1,r}$ formulas characterizing the color class t at iteration r of the k-dimensional Weisfeiler–Leman algorithm. So for $F \in \{G, H\}$ and $\overline{w} \in F^k$, we have that: $\chi_r(\overline{w}) = t \iff (F, \overline{w}) \vDash \psi_t$.

Denote:

$$\varphi := (\exists N\, x_{k+1})\Big(\bigwedge_{j=1}^{N} \psi_t(x_{i_j}/x_{k+1})\Big).$$

We note that $(G, \overline{g}) \vDash \varphi$, but $(H, \overline{h}) \not\vDash \varphi$. The result follows. ◀

We next relate the distinguishing power of $\mathcal{C}_k$ to the k-pebbling game.

▶ **Proposition 43.** *Fix $k \in \mathbb{Z}^+$. Suppose that G and H are groups of order n. Let (u, v) be a k-configuration. If $(G, u) \not\equiv_{\mathcal{C}_{k,r}} (H, v)$, then Spoiler has a winning strategy for the k-pebble game in at most r moves.*

Proof. The proof is by induction on r. Suppose that (u, v) is a k-configuration such that the map $u(x_i) \mapsto v(x_i)$ for all $x_i \in \text{Dom}(u)$, induces an isomorphism of

$$\langle\{u(x_i) : x_i \in \text{Dom}(u)\}\rangle \cong \langle\{v(x_i) : x_i \in \text{Dom}(u)\}\rangle.$$

So u and v satisfy the same quantifier-free formulas in $\mathcal{C}_k$. By the definition of the k-pebble game, Spoiler is unable to win without moving. Now fix $r \geq 0$, and suppose that for any k-configuration (u, v) such that $(G, u) \equiv_{\mathcal{C}_{k,r}} (H, v)$, Spoiler is unable to win the k-pebble game in at most r moves. Now suppose that (u, v) is a k-configuration such that: $(G, u) \not\equiv_{\mathcal{C}_{k,r+1}} (H, v)$.

If $(G, u) \not\equiv_{\mathcal{C}_{k,r}} (H, v)$, then Spoiler has a winning strategy by the inductive hypothesis. So suppose that: $(G, u) \equiv_{\mathcal{C}_{k,r}} (H, v)$. Thus, there exists $\varphi \in \mathcal{C}_{k,r+1} \setminus \mathcal{C}_{k,r}$ such that, without loss of generality, $(G, u) \vDash \varphi$ but $(H, v) \not\vDash \varphi$. We note that if φ is a conjunction, then (H, h) does not satisfy one of the conjuncts. However, such a conjunct belongs to $\mathcal{C}_{k,r}$, a contradiction. It follows that φ is of the form $(\exists N\, x_{k+1})\psi$, where $\psi \in \mathcal{C}_{k,r}$. We construct a winning strategy in the k-pebble game for Spoiler as follows. Spoiler begins by selecting the set $A \subset G$ of such variables x_{k+1} that satisfy ψ. Duplicator then selects a set B of size $|A|$. As $(H, v) \not\vDash \varphi$, there exists some $h^* \in B$ such that $(H, v(x_{k+1}/h^*)) \not\vDash \psi$. Spoiler pebbles h^*. Now regardless of the element $g^* \in A$ that Duplicator pebbles, $(G, u(x_{k+1}/g^*)) \vDash \psi$. Denote $u_1 := u(x_{k+1}/g^*)$

and $v_1 := v(x_{k+1}/h^*)$. So (u_1, v_1) is a k-configuration such that $(G, u_1) \models \psi$ and $(H, v_1) \not\models \psi$. Thus, $(G, u_1) \not\equiv_{\mathcal{C}_{k,r}} (H, v_1)$.

By the inductive hypothesis, Spoiler has a winning strategy in the remaining r moves of the k-pebbling game. The result follows. ◀

B Parallel Weisfeiler–Leman Implementation

We recall the statement of Theorem 15.

▶ **Theorem 44** (Theorem 15)**.** *Let $k \geq 2$ be constant, and let $r := r(n)$ be a function where n is the order of the input groups. We have the following.*

(a) The r-round, k-WL algorithm can be implemented by a logspace uniform family of TC *circuits of depth $O(r + \log(n))$ and size $O(r \cdot n^{3k})$.*
(b) The r-round, count-free k-WL algorithm can be implemented by a logspace uniform family of AC *circuits of depth $O(r + \log(n))$ and size $O(r \cdot n^{3k})$.*

The proof of Theorem 15 largely resembles that of Grohe & Verbitsky in the case of graphs [27]. The key difference lies in computing the initial coloring. In the setting of graphs, marked isomorphism can be tested in AC^0. So the initial coloring is AC^0 computable. However, for groups, the size of the subgroup may be exponential in the number of generators. To this end, we apply Theorem 11 to compute the initial coloring in L (see for instance, the proof of Corollary 12). The color refinement components of the Grohe–Verbitsky circuit then apply verbatim [27]. We note that the L implementaion for the initial coloring adds depth $O(\log(n))$ to the circuit in the setting of groups, while the AC^0 circuit to compute the initial coloring of graphs only requires constant depth.

Proof of Theorem 15. Let $N := 2n^k$. We fix a bijection between G^k nad $[n^k]$, and a bijection between H^k and $\{n^k + 1, \ldots, 2n^k\}$. For each $a \in [N]$, let $\overline{u}(a) \in G^k \cup H^k$ denote the k-tuple corresponding to a. We use $\text{tp}(\overline{u}(a))$ to denote the marked isomorphism type of $\overline{u}(a)$.

We construct a circuit with $(r + 2)$ layers, where layer 0 is an $O(\log(n))$-depth circuit and each layer $\ell \geq 1$ is a constant-depth circuit. Layer 0 computes the initial coloring. For $\ell \in [r]$, layer ℓ is used to refine the coloring obtained in the previous layer as per the (count-free) k-WL algorithm. Finally, the $(r + 1)$st layer computes the output of the circuit. For the count-free k-WL, we implement an AC circuit, while we implement a TC circuit for the counting version of k-WL where we track both the colors and their multiplicities.

For layer ℓ and $a, c \in [N]$, denote $X_\ell(a, c)$ to be the indicator function such that $X_\ell(a, c) = 1$ if and only if $\chi_\ell(\overline{u}(a)) = c$. For each $(a, c) \in [N] \times [N]$, level ℓ will have the output $X_\ell(a, c)$.

At layer 0, we have an L circuit to compute marked isomorphism. For each $a, c \in N$, we use the L circuit from Theorem 11 to check whether $\text{tp}(\overline{u}(a)) = \text{tp}(\overline{u}(c))$. Denote $Y_0(a, c) = 1 \iff \text{tp}(\overline{u}(a)) = \text{tp}(\overline{u}(c))$. Now define:

$$X_0(a, c) = Y_0(a, c) \wedge \bigwedge_{d=1}^{c-1} \neg Y_0(a, d).$$

So the initial color $\chi_0(\overline{u}(a))$ is the index c, of the numerically first tuple that has the same marked isomorphism type as $\overline{u}(a)$. Furthermore, computing $X_0(a, c)$ adds a constant layer AC^0 circuit on top of the L circuits that compute that $Y_0(a, d)$ functions. Thus, layer 0 is an AC^1 circuit.

Now fix $\ell \in [r]$, and suppose that layers $0, \ldots, \ell - 1$ have been defined. We now define layer ℓ as follows. For each $(a, c) \in [N] \times [N]$, we have an output $X_\ell(a, c)$, where $X_\ell(a, c) = 1 \iff \chi_\ell(\overline{u}(a)) = \chi_\ell(\overline{u}(c))$. For $\overline{u}(a) = (a_1, \ldots, a_k)$ and $x \in G$, denote a^{ix} to be:

$$(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_k).$$

Fix $a, c \in N$. We establish conditions for $\chi_\ell(\overline{u}(a)) = \chi_\ell(\overline{u}(c))$. In the counting version of WL, $\chi_\ell(\overline{u}(a)) = \chi_\ell(\overline{u}(c))$ if and only if the following two conditions are satisfied.

(i) $\chi_{\ell-1}(\overline{u}(a)) = \chi_{\ell-1}(\overline{u}(c))$.
(i) We have that:

$$\{\{(\chi_{\ell-1}(\overline{u}(a^{1x})), \ldots, \chi_{\ell-1}(\overline{u}(a^{kx}))) : x \in G\}\} = \{\{(\chi_{\ell-1}(\overline{u}(c^{1y})), \ldots, \chi_{\ell-1}(\overline{u}(c^{ky}))) : y \in H\}\}.$$

The following formula defines condition (i).

$$\bigwedge_{b=1}^{N} (X_{\ell-1}(a, b) \leftrightarrow X_{\ell-1}(c, b)). \tag{1}$$

So $X_{\ell-1}(a, b) = 1 \iff X_{\ell-1}(c, b) = 1$. We note that condition (1) can be computed using an AC^0 circuit.

To define condition (ii), we introduce a formula $\varphi(u, v)$ for each $u, v \in G$, where $\varphi(u, v)$ encodes the relationship that:

$$(\chi_{\ell-1}(\overline{u}(a^{1u})), \ldots, \chi_{\ell-1}(\overline{u}(a^{ku}))) = (\chi_{\ell-1}(\overline{u}(a^{1v})), \ldots, \chi_{\ell-1}(\overline{u}(a^{kv}))).$$

Precisely, we define:

$$\varphi(u, v) = \bigwedge_{j=1}^{k} \bigwedge_{c=1}^{N} (X_{\ell-1}(a^{ju}, c) \leftrightarrow X_{\ell-1}(a^{jv}, c)).$$

Similaly, we define a formula $\psi(u, v)$ for each $u \in G$ and each $v \in H$ to express whether:

$$(\chi_{\ell-1}(\overline{u}(a^{1u})), \ldots, \chi_{\ell-1}(\overline{u}(a^{ku}))) = (\chi_{\ell-1}(\overline{u}(c^{1u})), \ldots, \chi_{\ell-1}(\overline{u}(c^{ku}))).$$

Now condition (ii) can be expressed via the formula:

$$\bigwedge_{u \in G} \left(\left(\sum_{v \in G} \varphi(u, v) \right) = \left(\sum_{v \in H} \psi(u, h) \right) \right), \tag{2}$$

which can be implemented using a TC^0 circuit. Define $Y_\ell(a, c)$ to be the conjunction of the formulas in (1) and (2). So $Y_\ell(a, c)$ can be computed using a TC^0 circuit.

In the count-free version of WL, condition (ii) above is replaced by:

$$\{(\chi_{\ell-1}(\overline{u}(a^{1x})), \ldots, \chi_{\ell-1}(\overline{u}(a^{kx}))) : x \in G\} = \{(\chi_{\ell-1}(\overline{u}(c^{1y})), \ldots, \chi_{\ell-1}(\overline{u}(c^{ky}))) : y \in H\},$$

which can be expressed via the formulas:

$$\left(\bigwedge_{u \in G} \bigvee_{v \in H} \psi(u, v) \right) \wedge \left(\bigwedge_{v \in H} \bigvee_{u \in G} \psi(u, v) \right), \tag{3}$$

which can be implemented with an AC^0 circuit. Define $Y_\ell(a, c)$ to be the conjunction of the formulas in (1) and (3). As both conjuncts of $Y_\ell(a, c)$ can be implemented using AC^0 circuits, we have that $Y_\ell(a, c)$ itself can be implemented with an AC^0 circuit.

Now as in layer 0, we define:

$$X_\ell(a,c) = Y_\ell(a,c) \wedge \bigwedge_{d=1}^{c-1} \neg Y_\ell(a,d),$$

which completes the definition of layer ℓ. Note that this last step serves to re-index the colors after each refinement has been made.

We now define the $(r+1)$st layer of the circuit, which is the output layer. In the counting version of WL, the output of the circuit is defined by:

$$\bigwedge_{c=1}^{N} \left(\sum_{a=1}^{n^k} X_r(a,c) = \sum_{b=n^k+1}^{N} X_r(b,c) \right),$$

which can be computed with a TC^0 circuit. In the count-free version, the output of the circuit is defined by:

$$\bigwedge_{c=1}^{N} \left(\left(\bigvee_{a=1}^{n^k} X_r(a,c) \right) \leftrightarrow \left(\bigvee_{b=n^k+1}^{N} X_r(b,c) \right) \right),$$

which can be computed using a AC^0 circuit. The result follows. ◀

▶ Remark 45. In the above proof, we use an L circuit to compute the marked isomorphism types, followed by an AC^0 circuit to label each k-tuple with the corresponding color. We may replace this AC^0 circuit with an L circuit, which provides that the initial coloring may be computed in L.

C Proof of Theorem 21

In this section, we prove Theorem 21.

▶ **Theorem 46** (Theorem 21). *Suppose that $G = A_1 \times B_1$ and $H = A_2 \times B_2$, where A_1 and A_2 are Abelian, and B_1 and B_2 are non-Abelian groups. Furthermore, suppose that $Z(G)$ and $Z(H)$ are direct products of elementary Abelian groups.*

(a) Fix $d \in \mathbb{N}$. If B_1 and B_2 have order at most d, then we may decide in $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ whether $G \cong H$.

(b) Suppose that B_1 and B_2 have generating sets of size $O(1)$. Then we may decide in NC^2 whether $G \cong H$.

Proof of Theorem 21. Without loss of generality, suppose that $|A_1| = |A_2|$ and $|B_1| = |B_2|$. Note that $Z(G) = A_1 \times Z(B_1)$ and $Z(H) = A_2 \times Z(B_2)$. We note that $G \cong H$ if and only if $A_1 \cong A_2$ and $B_1 \cong B_2$. Our strategy is to non-deterministically guess candidates C_1 and C_2. Intuitively, we think of guessing C_1 and C_2 as trying to guess B_1 and B_2. As there is no easy way to verify whether $C_i \cong B_i$, we instead try to find an Abelian direct complement A_i' for C_i such that $G = A_1' \times C_1$ and $H = A_2' \times C_2$. Note that the A_i' need not be isomorphic to the A_i.

1. We first compute $Z(G)$ and $Z(H)$. We identify whether an element $g \in G$ belongs to $Z(G)$ by testing whether $gs = sg$ for all $s \in G$. We similarly identify the elements of $Z(H)$. This may be done in AC^0.

2. Once $Z(G)$ and $Z(H)$ have been computed, we test whether $Z(G) \cong Z(H)$ are isomorphic using the $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$ test of Chattopadhyay, Torán, and Wagner [19]. If $Z(G) \not\cong Z(H)$, we return that $G \not\cong H$.

3. We next identify candidates C_1 and C_2 for B_1 and B_2 respectively, and test whether $C_1 \cong C_2$. We have the following cases.
   - **Case 1:** Suppose that B_1 and B_2 have order at most d. Without loss of generality, suppose that B_1 and B_2 have order exactly d. We guess elements $x_1, \ldots, x_d$ for our candidate subgroup C_1, and we similarly guess $y_1, \ldots, y_d$ for our candidate subgroup C_2. We may, in time $O(1)$, check whether $C_1 \cong C_2$. Now as $Z(G)$ and $Z(H)$ are direct products of elementary Abelian groups, we have that $Z(G) = A_1' \times Z(C_1)$ and $Z(H) = A_2' \times Z(C_2)$, where A_1' and A_2' are Abelian. Note that for each $i \in [2]$, A_i' need not be (isomorphic to) A_i.

     We note that $G = A_1' \times C_1$ and $H = A_2' \times C_2$ if and only if $G = \langle C_1, Z(G) \rangle$ and $H = \langle C_2, Z(H) \rangle$. We may test whether $G = \langle C_1, Z(G) \rangle$ by checking whether each $g \in G$ can be written as a product of $b_i z_i$ for $b_i \in C_1$ and $z_i \in Z(G)$. This may be done in AC^0. We similarly test whether $H = \langle B_2, Z(H) \rangle$. In this case, our algorithm is $\mathsf{TC}^0(\mathsf{FOLL}) \cap \mathsf{L}$-computable.

   - **Case 2:** Suppose instead that B_1 and B_2 are only assumed to have generating sets of size $O(1)$. We begin by guessing generating sequences $\overline{x} := (x_1, \ldots, x_\ell)$ for our candidate subgroup C_1 and $\overline{y} := (y_1, \ldots, y_\ell)$ for our candidate subgroup C_2. Let $C_1 := \langle x_1, \ldots, x_\ell \rangle$ and $C_2 := \langle y_1, \ldots, y_\ell \rangle$. Using the L marked isomorphism test of Theorem 11, we check whether the map $x_i \mapsto y_i$ for all $i \in [\ell]$ extends to an isomorphism of C_1 and C_2. If $C_1 \not\cong C_2$, we reject.

     Next, we test whether $C_1 \trianglelefteq G$ and $C_2 \trianglelefteq H$. This can be done in L; see for instance, [54, Theorem 7.3.8]. We next enumerate $Z(C_1)$ by examining each $g \in G$ and checking (i) whether $g \in C_1$, and (ii) whether $gx_i = x_i g$ for all $i \in [\ell]$. Condition (i) may be checked in L [8], and condition (ii) may be checked in AC^0. We similarly enumerate $Z(C_2)$.

     Note that as $Z(G)$ and $Z(H)$ are direct products of elementary Abelian groups, we have that $Z(G) = A_1' \times Z(C_1)$ and $Z(H) = A_2' \times Z(C_2)$. Note that for each $i \in [2]$, A_i' need not be (isomorphic to) A_i. Once we have $Z(C_1)$ and $Z(C_2)$, we may in NC^2 compute the Smith Normal Forms for $Z(G)/Z(C_1)$ and $Z(H)/Z(C_2)$ respectively [56]. Using these Smith Normal Forms, we may now easily check whether $A_1' \cong A_2'$, as well as pick bases for A_1' and A_2'. After selecting bases for A_1' and A_2', we check in L whether $A_1' \cong G/C_1$ by checking whether each commutator $[g, h] \in G$ belongs to C_1 [8]. We similarly check whether $A_2' \cong H/C_2$. As $Z(G)$ and $Z(H)$ are direct products of elementary Abelian groups, we have that C_1 and C_2 are direct complements for A_1 and A_2.

     In this case, our algorithm is NC^2-computable.

The result follows. ◀

▶ Remark 47. The key bottleneck in Case 2 lies in computing the Smith Normal Form. Improving upon the NC^2 barrier for computing the Smith Normal Form is a major open problem. It would also be of interest construct an isomorphism test that does not explicitly rely on computing the Smith Normal Form.

D Proof that Latin Square Isotopy is in β_2L $\cap$ β_2FOLL

In this section, we prove Theorem 22.

D.1 Preliminaries- Groups and Quasigroups

A *quasigroup* consists of a set G and a binary operation $\star : G \times G \to G$ satisfying the following.

- For every $a, b \in G$, there exists a unique x such that $ax = b$.
- For every $a, b \in G$, there exists a unique y such that $ya = b$.

Unless otherwise stated, all groups and quasigroups are assumed to be finite and represented using their Cayley tables. For a (quasi)group of order n, thet Cayley table has n^2 entries, each represented by a string of size $\lceil \log_2(n) \rceil$. For an element g in the group G, we denote the *order* of g as $|g|$. Denote $d(G)$ to be the minimum size of a generating set for the group G.

A *Latin square* of order n is an $n \times n$ matrix L where each cell $L_{ij} \in [n]$, and each element of $[n]$ appears exactly once in a given row or a given column. Latin squares are precisely the Cayley tables corresponding to quasigroups. An *isotopy* of Latin squares L_1 and L_2 is an ordered triple (α, β, γ), where $\alpha, \beta, \gamma : L_1 \to L_2$ are bijections satisfying the following: whenever $ab = c$ in L_1, we have that $\alpha(a)\beta(b) = \gamma(c)$ in L_2.

As quasigroups are non-associative, the parenthesization of a given expression may impact the resulting value. We restrict attention to balanced parenthesizations, which ensure that words of the form $g_0 g_1 \cdots g_k$ are evaluated using a balanced binary tree with $k+1$ leves and therefore depth $O(\log(\log(n)))$.

For a given Latin square L of order n, we associate a *Latin square graph* $G(L)$ that has n^2 vertices; one for each triple (a, b, c) that satisfies $ab = c$. Two vertices (a, b, c) and (x, y, z) are adjacent in $G(L)$ precisely if $a = x$, $b = y$, or $c = z$. Miller showed that two Latin squares L_1 and L_2 are isotopic if and only if $G(L_1) \cong G(L_2)$ [45]. Albert showed that a quasigroup Q is isomotopic to a group G if and only if Q is isomorphic to G. In general, isotopic quasigroups need not be isomorphic [1].

D.2 Proof of Theorem 22

▶ **Theorem 48** (Theorem 22). *Let L_1, L_2 be Latin squares of order n. Deciding whether L_1 and L_2 are isotopic is in β_2L $\cap$ β_2FOLL.*

We begin with the following lemma.

▶ **Lemma 49.** *Let L_1 and L_2 be Latin squares of order n, and let $k = 4\lceil \log(n) \rceil$. Suppose that $(g_0, g_1, \ldots, g_k)$ and $(h_0, \ldots, h_k)$ are cube generating sequences with balanced parenthesization P. Deciding whether the map $g_i \mapsto h_i$ for all $i \in \{0, \ldots, k\}$ extends to a bijection is in L $\cap$ FOLL.*

Proof. For each $(g, h) \in L_1 \times L_2$, define $X(g, h) = 1$ if and only if there exists $(\epsilon_1, \ldots, \epsilon_k) \in \{0, 1\}^k$ such that:

$$g := g_0 g_1^{\epsilon_1} \cdots g_k^{\epsilon_k},$$
$$h := h_0 h_1^{\epsilon_1} \cdots h_k^{\epsilon_k}.$$

Chattophadyay, Torán, and Wagner showed that the cube words for g and h can be computed in $\mathsf{L} \cap \mathsf{FOLL}$ [19], so $X(g, h)$ is computable in $\mathsf{L} \cap \mathsf{FOLL}$. We note that the FOLL bound follows from the fact that P is a balanced parenthesization. It suffices to check whether the induced map is surjective. For each $h \in L_2$, define $Y(h) = 1$ if and only if there exists $g \in L_1$ such that $g \mapsto h$, under the map induced by the mapping $g_i \mapsto h_i$ for all $i \in \{0, \ldots, k\}$. We note that:

$$Y(h) = \bigvee_{g \in L_1} X(g, h),$$

which is computable using an AC^0 circuit. The map is surjective if and only if $Y(h) = 1$ for all $h \in L_2$, which is defined by the following condition:

$$\varphi := \bigwedge_{h \in L_2} Y(h).$$

Observe that φ is AC^0 computable. ◀

Proof of Theorem 22. Let $k = 4\lceil \log_2(n) \rceil$. We use $4k^2$ random bits to guess cube generating sequences $A, B \subset L$ and $A', B' \subset L'$, where $A = \{a_1, \ldots, a_k\}$, $B = \{b_1, \ldots, b_k\}$, $A' = \{a'_1, \ldots, a'_k\}$, and $B' = \{b'_1, \ldots, b'_k\}$. We may then in $\mathsf{L} \cap \mathsf{FOLL}$ check the following:

- The map $a_i \mapsto a'_i$ extends to a bijection of $\langle A \rangle$ and $\langle A' \rangle$. In particular, we may check in $\mathsf{L} \cap \mathsf{FOLL}$ whether $L_1 = \langle A \rangle$ and $L_2 = \langle A' \rangle$ (see [19, Theorem 5]). The procedure in Lemma 49 that decides whether $\langle A \rangle \cong \langle A' \rangle$ also explicitly computes a bijection $\varphi_A : \langle A \rangle \to \langle A' \rangle$ if one exists.
- We proceed analogously for the map $b_i \mapsto b'_i$ for all $i \in \{0, \ldots, k\}$.

Suppose now that the bijections $\varphi_A, \varphi_B : L_1 \to L_2$ have been constructed. We now attempt to construct a relation $\varphi_C : L_1 \to L_2$ in such a way that φ_C is a bijection if and only if $(\varphi_A, \varphi_B, \varphi_C)$ is an isotopism. For each pair $(\ell, m) \in L_1 \times L_2$, define $X(\ell, m) = 1$ if and only if we define φ_C to map $\ell \mapsto m$. For each $(\epsilon_1, \ldots, \epsilon_k), (\nu_1, \ldots, \nu_k) \in \{0, 1\}^k$, we do the following:

(a) Compute:

$$\begin{aligned} g &:= g_0 g_1^{\epsilon_1} \cdots g_k^{\epsilon_k}, \\ h &:= h_0 h_1^{\nu_1} \cdots h_k^{\nu_k}. \end{aligned}$$

We note that computing g and h can be done in $\mathsf{L} \cap \mathsf{FOLL}$ (see Chattopadhyay, Toràn, and Wagner [19]).

(b) Compute $\ell := gh$, $\varphi_A(g)$, $\varphi_B(h)$, and $m := \varphi_A(g)\varphi_B(h)$. We set $\varphi_C(\ell) = m$, which we indicate by defining $X(\ell, m) = 1$. The computations at this stage are computable using an AC^0 circuit.

(c) For each $m \in L_2$, we define $Y(m)$ to be 1 if and only if there exists $\ell \in L_1$ such that $\varphi_C(\ell) = m$. This is realized by the formula:

$$Y(m) = \bigvee_{\ell \in L_1} X(\ell, m),$$

which is computable using an AC^0 circuit.

(d) Finally, we test whether φ_C is a bijection. As L_1 and L_2 are finite structures of the same order n, φ_C is a bijection if and only if φ_C is surjective. We test for surjectivity using the following condition:

$$\psi = \bigwedge_{m \in L_2} Y(m),$$

which is computable using an AC^0 circuit.

By construction, φ_C was constructed so that $(\varphi_A, \varphi_B, \varphi_C)$ satisfies the homotopy condition, so, as they are also bijective, they are an isotopy. Thus, checking whether L_1 and L_2 are isotopic is in $\beta_2\mathsf{L} \cap \beta_2\mathsf{FOLL}$.

◀

References

1 A. A. Albert. Quasigroups. i. *Transactions of the American Mathematical Society*, 54(3):507–519, 1943. URL: `http://www.jstor.org/stable/1990259`.

2 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`.

3 L. Babai and E. Szemeredi. On the complexity of matrix group problems i. In *25th Annual Symposium onFoundations of Computer Science, 1984.*, pages 229–240, 1984.

4 László Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015. URL: `http://arxiv.org/abs/1512.03547`, `arXiv:1512.03547`.

5 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*, SODA '11, page 1395–1408, USA, 2011. Society for Industrial and Applied Mathematics.

6 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`.

7 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`.

8 David A.Mix Barrington and Pierre McKenzie. Oracle branching programs and logspace versus p. *Information and Computation*, 95(1):96 – 115, 1991. URL: `http://www.sciencedirect.com/science/article/pii/089054019190017V`, `doi:https://doi.org/10.1016/0890-5401(91)90017-V`.

9 David Mix Barrington, Peter Kadau, Klaus-Jörn Lange, and Pierre McKenzie. On the complexity of some problems on groups input as multiplication tables. *Journal of Computer and System Sciences*, 63(2):186 – 200, 2001. URL: `http://www.sciencedirect.com/science/article/pii/S0022000001917647`, `doi:https://doi.org/10.1006/jcss.2001.1764`.

10 Hans Ulrich Besche and Bettina Eick. Construction of finite groups. *J. Symb. Comput.*, 27(4):387–404, 1999. `doi:10.1006/jsco.1998.0258`.

11 Hans Ulrich Besche, Bettina Eick, and E.A. O'Brien. A millennium project: Constructing small groups. *Intern. J. Alg. and Comput*, 12:623–644, 2002. `doi:10.1142/S0218196702001115`.

12 Jendrik Brachter and Pascal Schweitzer. On the weisfeiler-leman dimension of finite groups. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, page 287–300, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3373718.3394786`.

13 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.

14 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. URL: `http://www.sciencedirect.com/science/article/pii/S0021869316304628`, `doi:https://doi.org/10.1016/j.jalgebra.2016.12.007`.

15 Harry Buhrman and Steven Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *FSTTCS*, 1992.

16 J.-Y. Cai, M. Furer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, page 612–617, USA, 1989. IEEE Computer Society. `doi:10.1109/SFCS.1989.63543`.

17 John J. Cannon and Derek F. Holt. Automorphism group computation and isomorphism testing in finite groups. *J. Symb. Comput.*, 35:241–267, March 2003. `doi:10.1016/S0747-7171(02)00133-5`.

18 Robert Daniel. Carmichael. *Introduction to the theory of groups of finite order.* Dover, 1956.

19 A. Chattopadhyay, J. Torán, and F. Wagner. Graph isomorphism is not AC^0 reducible to group isomorphism. *Electron. Colloquium Comput. Complex.*, 17:117, 2010.

20 Bireswar Das and Shivdutt Sharma. Nearly linear time isomorphism algorithms for some nonabelian group classes. In René van Bevern and Gregory Kucherov, editors, *Computer Science – Theory and Applications*, pages 80–92, Cham, 2019. Springer International Publishing.

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

22 Bettina Eick, C. R. Leedham-Green, and E. A. O'Brien. Constructing automorphism groups of p-groups. *Comm. Algebra*, 30(5):2271–2295, 2002. `doi:10.1081/AGB-120003468`.

23 Walid Gomaa. Descriptive complexity of finite abelian groups. *IJAC*, 20:1087–1116, 12 2010. `doi:10.1142/S0218196710006047`.

24 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`.

25 M. Grohe. *Descriptive complexity, canonisation, and definable graph structure theory.* 08 2017. `doi:10.1017/9781139028868`.

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

27 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*, pages 3–14, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

28 Hermann Heineken and Hans Liebeck. The occurrence of finite groups in the automorphism group of nilpotent groups of class 2. *Archiv der Mathematik*, 25:8–16, 1974.

29 Lauri Hella. Definability hierarchies of generalized quantifiers. *Annals of Pure and Applied Logic*, 43(3):235 – 271, 1989. URL: `http://www.sciencedirect.com/science/article/pii/0168007289900705`, `doi:https://doi.org/10.1016/0168-0072(89)90070-5`.

30 Lauri Hella. Logical hierarchies in ptime. *Information and Computation*, 129(1):1 – 19, 1996. URL: `http://www.sciencedirect.com/science/article/pii/S089054019690070X`, `doi:https://doi.org/10.1006/inco.1996.0070`.

31 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512 – 530, 2001. URL: `http://www.sciencedirect.com/science/article/pii/S002200000191774X`, `doi:https://doi.org/10.1006/jcss.2001.1774`.

32 T. Kavitha. Linear time algorithms for abelian group isomorphism and related problems. *Journal of Computer and System Sciences*, 73(6):986 – 996, 2007. URL: `http://www.sciencedirect.com/science/article/pii/S0022000007000293`, `doi:https://doi.org/10.1016/j.jcss.2007.03.013`.

33 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`.

34 Sandra Kiefer, Ilia Ponomarenko, and Pascal Schweitzer. The weisfeiler–leman dimension of planar graphs is at most 3. *J. ACM*, 66(6), November 2019. `doi:10.1145/3333003`.

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

36 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`.

37 François Le Gall and David J. Rosenbaum. On the group and color isomorphism problems. arXiv:1609.08253, 2016.

38 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`.

39 R. J. Lipton, L. Snyder, and Y. Zalcstein. The complexity of word and isomorphism problems for finite groups. 1977. URL: `https://apps.dtic.mil/dtic/tr/fulltext/u2/a053246.pdf`.

40 José López-Presa and Antonio Fernández Anta. Fast algorithm for graph isomorphism testing. pages 221–232, 06 2009. `doi:10.1007/978-3-642-02011-7_21`.

41 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`.

42 Yuki Maehara, May 2013. URL: `https://arxiv.org/pdf/1305.0080.pdf`.

43 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014. URL: `https://www.sciencedirect.com/science/article/pii/S0747717113001193`, `doi:https://doi.org/10.1016/j.jsc.2013.09.003`.

44 Alan H. Mekler. Stability of nilpotent groups of class 2 and prime exponent. *The Journal of Symbolic Logic*, 46(4):781–788, 1981. URL: `http://www.jstor.org/stable/2273227`.

45 Gary L. Miller. On the nlog n isomorphism technique (a preliminary report). In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, page 51–58, New York, NY, USA, 1978. Association for Computing Machinery. `doi:10.1145/800133.804331`.

46 André Nies and Katrin Tent. Describing finite groups by short first-order sentences. *Israel Journal of Mathematics*, 221, 09 2014. `doi:10.1007/s11856-017-1563-2`.

47 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`.

48 David J. Rosenbaum. Bidirectional collision detection and faster deterministic isomorphism testing. *ArXiv*, abs/1304.3935, 2013.

49 David J. Rosenbaum and Fabian Wagner. Beating the generator-enumeration bound for p-group isomorphism. *Theor. Comput. Sci.*, 593:16–25, 2015. `doi:10.1016/j.tcs.2015.05.036`.

50 C. Savage. An $o(n^2)$ algorithm for abelian group isomorphism. Technical report, North Carolina State University, 01 1980.

51 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`.

52 Bangsheng Tang. 2013. URL: `http://papakonstantinou.org/periklis/pdfs/bangsheng_thesis.pdf`.

53 Jacobo Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004. `arXiv:https://doi.org/10.1137/S009753970241096X`, `doi:10.1137/S009753970241096X`.

54 T. Vijayaraghavan. Classifying certain algebraic problems using logspace counting classes[hbni th 5]. 2008.

55 Narayan Vikas. Ano(n) algorithm for abelianp-group isomorphism and ano(nlogn) algorithm for abelian group isomorphism. *Journal of Computer and System Sciences*, 53(1):1 – 9, 1996. URL: `http://www.sciencedirect.com/science/article/pii/S0022000096900458`, `doi:https://doi.org/10.1006/jcss.1996.0045`.

56 G. Villard. Fast parallel algorithms for matrix reduction to normal forms. *Applicable Algebra in Engineering, Communication and Computing*, 8:511–537, 1997.

57 B. Yu. Weisfeiler and A. A. Leman. Reduction of a graph to a canonical form and an algebra arising during this reduction. 1968.

58 James B. Wilson. The threshold for subgroup profiles to agree is logarithmic. *Theory of Computing*, 15(19):1–25, 2019. `doi:10.4086/toc.2019.v015a019`.

59 Marty J. Wolf. Nondeterministic circuits, space complexity and quasigroups. *Theoretical Computer Science*, 125(2):295 – 313, 1994. URL: `http://www.sciencedirect.com/science/article/pii/030439759200014I`, `doi:https://doi.org/10.1016/0304-3975(92)00014-I`.