ISOMORPHISM TESTING OF GROUPS OF CUBE-FREE ORDER

HEIKO DIETRICH AND JAMES B. WILSON

Abstract. A group $G$ has cube-free order if no prime to the third power divides $|G|$. We describe an algorithm that given two cube-free groups $G$ and $H$ of known order, decides whether $G \cong H$, and, if so, constructs an isomorphism $G \rightarrow H$. If the groups are input as permutation groups, then our algorithm runs in time polynomial in the input size, improving on the previous super-polynomial bound. An implementation of our algorithm is provided for the computer algebra system GAP.

In memory of C.C. Sims.

1. Introduction

Capturing the natural concept of symmetry, groups are one the most prominent algebraic structures in science. Yet, it is still a challenge to decide whether two finite groups are isomorphic. Despite abundant knowledge about groups, presently no one has provided an isomorphism test for all finite groups whose complexity improves substantively over brute-force. In the most general form there is no known polynomial-time isomorphism test even for non-deterministic Turing machines, that is, the problem may lie outside the complexity classes NP and co-NP (see [2, Corollary 4.9]). At the time of this writing, the available implementations of algorithms that test isomorphism on broad classes of groups can run out of memory or run for days on examples of orders only a few thousand, see [6, Section 1.1] and Table 1. To isolate the critical difficulties in group isomorphism, it helps to consider special classes of groups as has been done recently in [1,3,6,9,11,31].

This paper is a part of a larger project intended to describe for which orders of groups is group isomorphism tractable: details of this project are given in [11]. In particular, in [11] we have described polynomial-time algorithms for isomorphism testing of abelian and meta-cyclic groups of most orders; the computational framework for these algorithms is built upon type theory and groups of so-called black-box type. By a theorem of Hôlder ([26, 10.1.10]), all groups of square-free order are coprime meta-cyclic, that is, they can be decomposed as $G = A \ltimes B$ where $A,B \leq G$ are cyclic subgroups of coprime orders; unfortunately, [11, Theorem 1.2] is not guaranteed for all square-free orders. In this paper, we switch to a more restrictive computational model, allowing us to make progress for isomorphism testing of square-free and cube-free groups. Specifically, here we consider groups generated by a set $S$ of permutations on a finite set $\Omega$. This gives us access to a robust family of algorithms by Sims and many others (see [16,28]) that run in time polynomial in $|\Omega| \cdot |S|$. Note that the order of such a group $G$ can be exponential in $|\Omega| \cdot |S|$, even when restricted to groups of square-free order, see Proposition 2.1. The main result of this paper is the following theorem.

Theorem 1.1. There is an algorithm that given groups $G$ and $H$ of permutations on finitely many points, decides whether they are of cube-free order, and if so, decides that $G \not\cong H$ or constructs an isomorphism $G \rightarrow H$. The algorithm runs in time polynomial in the input size.

Theorem 1.1 is based on the structure analysis of cube-free groups by Eick & Dietrich [9] and Qiao & Li [25]. A top-level description of our algorithm is given in Section 3.2. Importantly, our algorithm translates to a functioning implementation for the system GAP [14], in the package “Cubefree” [10]. As a side-product, we also discuss algorithms related to the construction of complements of $\Omega$-groups, Sylow towers, socles, and constructive presentations, see Section 4. These algorithms have applications beyond cube-free groups and might be of general interest in computational group theory.

Date: October 9, 2018.

Key words and phrases. finite groups, cube-free groups, group isomorphisms.

This research was partially supported by the Simons Foundation, the Mathematisches Forschungsinstitut Oberwolfach, and NSF grant DMS-1620454. The first author thanks the Lehrstuhl D of the RWTH Aachen for the great hospitality during his Simon Visiting Professorship in Summer 2016. Both authors thank Alexander Hulpke for advice on conjugacy problems.
1.1. Limitations. In contrast to our work in [11], Theorem 1.1 no longer applies to a dense set of orders: the density of positive integers $n$ which are square-free and cube-free tends to $1/\zeta(2) \approx 0.61$ and $1/\zeta(3) \approx 0.83$, respectively, where $\zeta(x)$ is the Riemann $\zeta$-function, see [12, (2)]. It is known that most isomorphism types of groups accumulate at orders with large prime-power divisors. Indeed, Higman, Sims, and Pyber [4] proved that the number of groups of order $n$, up to isomorphism, tends to $n^{n^{2\mu(n)^2}/27+O(\log n)}$ where $\mu(n) = \max\{k : n \text{ is not } k\text{-free}\}$. Specifically, the number of pairwise non-isomorphic groups of a cube-free order $n$ is not more than $O(n^8)$, with speculation that the tight bound is $o(n^2)$, see [4, p. 236]. The prevailing belief in works like [1, 31] is that the difficult instances of group isomorphism are when $\mu(n)$ is unbounded, especially when $n$ is a prime power. Isomorphism testing of finite $p$-groups is indeed a research area that has attracted a lot of attention. However, Theorem 1.1 completely handles an easily described family of group orders which may make it easier to use in applications. A further point is that groups of cube-free order exhibit many of the fundamental components of finite groups. For instance, groups of cube-free order need not be solvable, to wit the simple alternating group $A_5$ has cube-free order 60. When decomposed into canonical series, such as the Fitting series, the associated extensions have nontrivial first and second cohomology groups – a measure of how difficult it is to compare different extensions.

1.2. Structure of the paper. In Section 2 we introduce some notation and comment on the computational model for our algorithm. In Section 3 we recall the structure of cube-free groups and give a top-level description of our isomorphism test. Various preliminary algorithms (for example, related to the construction of Sylow bases and towers, $\Omega$-complements, socles, and constructive presentations) are described in Section 4. The proof of the main theorem is broken up into three progressively more general families: the solvable Frattini-free case (Section 5), the general solvable case (Section 6), and finally the general case (Section 7). We have implemented many aspects of this algorithm in the computer algebra system GAP and comment on some examples in Section 8.

2. Notation and computational model

2.1. Notation. We reserve $p$ for prime numbers and $n$ for group orders. For a positive integer $n$ we denote by $C_n$ a cyclic group of order $n$, and $\mathbb{Z}/n$ for the explicit encoding as integers, in which we are further permitted to treat the structure as a ring. Let $(\mathbb{Z}/n)^{\times}$ denote the units of this ring. Direct products of groups are denoted variously by “\times” or exponents. Throughout, $\mathbb{F}_q$ is a field of order $q$ and $\text{GL}_d(q)$ is the group of invertible $(d \times d)$- matrices over $\mathbb{F}_q$. The group $\text{PSL}_d(q)$ consists of matrices of determinant 1 modulo scalar matrices.

For a group $G$ and $g, h \in G$, conjugates and commutators are $g^h = h^{-1}gh$ and $[g, h] = g^{-1}gh$, respectively. For subsets $X, Y \subseteq G$ let $[X, Y] = \langle [x, y] : x \in X, y \in Y \rangle$; the centralizer and normalizer of $X$ in $G$ are $C_G(X) = \{g \in G : [X, g] = 1\}$ and $N_G(X) = \{g \in G : [X, g] \subseteq X\}$, respectively. The derived series of $G$ has terms $G^{(n)} = [G^{(n-1)}, G^{(n)}]$ for $n \geq 1$, with $G^{(1)} = G$. We read group extensions from the right and use $A \ltimes B$ for split extensions; we also write $A \ltimes B$ to emphasize the action $\varphi : A \to \text{Aut}(B)$. Hence, $A \ltimes B \ltimes C \ltimes D$ stands for $((A \ltimes B) \ltimes C) \ltimes D$, etc.

We mostly adhere to protocol set out in standard literature on computational group theory, such as the Handbook of Computation Group Theory [16] and the books of Robinson [26] and Seress [28].
One simple but critical implication of our computational model is that if a prime $p$ divides the group order $|G|$, then $p$ divides $d!$, where $d$ is the size of the permutation domain; so $p \leq d$, which is less than the input size for $G$. This shows that all primes dividing the group order are small, allowing for polynomial-time factorization and other relevant number theory. Moreover, many essential group theoretic structures of groups of permutations (and their quotients) can be computed in polynomial time, as outlined in [28, p. 49] and [17] Section 4. For example, it is possible to compute group orders, to produce constructive presentations, and to test membership constructively. For solvable permutation groups one can also efficiently get a constructive polycyclic presentation (see Lemma 4.7).

Before we begin, we demonstrate that the assumption that our groups are input by permutations is not an automatic improvement in the complexity. In particular, we show that large groups of square-free (and so also cube-free) order can arise as permutation groups in small degrees.

**Proposition 2.1.** Let $G$ be a square-free group of order $n = p_1 \cdots p_\ell$, with each $p_i$ prime. The group $G$ can be faithfully represented in a permutation group of degree $p_1 + \cdots + p_\ell$. For infinitely many square-free $m$, there is a faithful permutation representation of the groups of order $m$ on $O(\log^2 m)$ points.

**Proof.** Hölder’s classification [26 (10.1.10)] shows that $G \cong C_a \rtimes C_b$ with $n = ab$. Since $a$ is square-free, all subgroups of $C_a$ are direct factors, thus $C_a = C_d \times C_{C_a} (C_b)$ for a subgroup $C_d$, and $C_a \rtimes C_b = C_d \rtimes C_{C_a}$, where the centralizer in $C_d$ of $C_b$ is trivial. Thus, we can assume that $C_a \rtimes C_b$ with $C_a$ acting faithfully on $C_b$, and $a = p_1 \cdots p_s$ and $b = p_{s+1} \cdots p_\ell$. Using disjoint $p_i$-cycles for each $i > s$, we faithfully represent $C_b$ on $p_{s+1} + \cdots + p_\ell$ points. Since $C_a$ acts faithfully on $C_b$, that representation can be given on the disjoint cycles of $C_b$, that is, $C_a \rtimes C_b$ is faithfully represented on $p_{s+1} + \cdots + p_\ell$ points. The first claim follows. For the last observation, let $m = r_1 \cdots r_\ell$ be the product of the first $\ell$-primes. These primorials have asymptotic growth $m \in \exp((1 + \Theta(1))\ell \log \ell)$, see [27] (3.16). Meanwhile, as just shown, the groups of order $m$ can all be represented faithfully on as few as $r_1 + \cdots + r_\ell$ points, and $r_1 + \cdots + r_\ell \in \Omega((\ell^2 \log \ell)$ by [24] Theorem C.

### 3. Summary of the algorithm

#### 3.1. Structure of cube-free groups.

For a finite group $G$ we denote by $\Phi(G)$ and $\soc(G)$ its Frattini subgroup and its socle, respectively; the first is the intersection of all maximal subgroups of $G$, and the latter is the subgroup generated by all minimal normal subgroups. We write $G_\Phi$ for the Frattini quotient $G/\Phi(G)$. A group is Frattini-free if $\Phi(G) = 1$; in particular, $G_\Phi$ is Frattini-free. By [9], every group $G$ of cube-free order can be decomposed as

$$G = A \times L$$

where $A$ is trivial or $A = \text{PSL}_2(p)$ for a prime $p > 3$ with $p \pm 1$ cube-free, and $L$ is solvable with abelian Frattini subgroup $\Phi(L) = \Phi(G)$ whose order is square-free and divides the order of the Frattini quotient $L_\Phi = L/\Phi(L)$. The latter satisfies

$$L_\Phi = K \ltimes (B \times C)$$

where $\soc(L_\Phi) = B \times C$ is the socle of $L_\Phi$ with

$$B = \prod_{i=1}^s \mathbb{Z}/p_i \text{ and } C = \prod_{j=s+1}^m (\mathbb{Z}/p_j)^2$$

for distinct primes $p_1, \ldots, p_m$. Let $X \ltimes Y$ denote a subdirect product, that is, a subgroup of $X \times Y$ whose projections to $X$ and $Y$ are surjective. With this notation, we have

$$K = K_1 \ltimes \cdots \ltimes K_m \leq \Aut(B \times C) = \prod_{i=1}^s \text{GL}_1(p_i) \times \prod_{j=s+1}^m \text{GL}_2(p_j);$$

It follows from work of Gaschütz (see [9] Lemma 9) that two solvable Frattini-free groups $K \ltimes (B \times C)$ and $\tilde{K} \ltimes (B \times C)$ with $K, \tilde{K} \leq \Aut(B \times C)$ as above are isomorphic if and only if $K$ and $\tilde{K}$ are conjugate in $\Aut(B \times C)$; this is one of the reasons why our proposed isomorphism algorithm works
so efficiently. Lastly, we recall that $L$ is determined by $L_\Phi$: there exists, up to isomorphism, a unique extension $M$ of $L_\Phi$ by $\Phi(L)$ such that $\Phi(M) \cong \Phi(L)$ and $M/\Phi(M) \cong L_\Phi$, see [9] Theorem 11.

**Remark 3.1.** Taunt [29] was probably the first who considered the class of cube-free groups. The focus in the work of Dietrich & Eick [9] is on a construction algorithm for all cube-free groups of a fixed order, up to isomorphism; the approach is based on the so-called Frattini extension method (see [16] §11.4.1). Complimentary to this work, Qiao & Li [25] also analyzed the structure of cube-free groups. They proved in [25] Theorem 1.1 that for every group $G$ of cube-free order there exist integers $a, b, c, d > 0$ such that $G$ is isomorphic to

\[(C_c \times C_d^2) \times (C_a \times C_b^2) \quad \text{or} \quad G_2 \times (C_c \times C_d^2) \times (C_a \times C_b^2) \quad \text{with } G_2 \leq G \text{ a Sylow 2-subgroup}, \]

\[\text{PSL}_2(p) \times (C_c \times C_d^2) \times (C_a \times C_b^2) \quad \text{for some prime } p.\]

Left unclassified in this description are the relevant actions of the semidirect products, and a classification up to isomorphism. As we have shown in [11] Section 4, even for meta-cyclic groups, recovering the appropriate actions and comparing them is in general not easy.

Among the implications of these decomposition results is that a solvable group $G$ of cube-free order has a Sylow tower, that is, a normal series such that each section is isomorphic to a Sylow subgroup of $G$, cf. [25] Corollary 3.4 & Theorem 3.9.

**3.2. The algorithm.** Let $G$ and $\tilde{G}$ be cube-free groups. We now describe the main steps of our algorithm to construct an isomorphism $G \to \tilde{G}$, which fails if and only if $G \not\cong \tilde{G}$. Our approach is to determine, for each group, the Frattini extension structure as described in Section 3. Since our groups are input by permutations, it is possible to decide if $|G| = |\tilde{G}|$ and also to factorize this order. It simplifies our treatment to assume that the groups are of the same order and that the prime factors of this order are known. First, for $G$ (and similarly for $\tilde{G}$) we do the following:

(i) Decompose $G = A \times L$ with $A = 1$ or $A = \text{PSL}_2(p)$ simple, and $L$ solvable.
(ii) Compute the Frattini subgroup $\Phi(L)$ and the Frattini quotient $L_\Phi = L/\Phi(L)$.
(iii) Compute $\text{soc}(L_\Phi) = B \times C$ and $K \leq \text{Aut}(B \times C)$ such that $L_\Phi = K \times (B \times C)$.

Then we proceed as follows; if one of these steps fails, then $G \not\cong \tilde{G}$ is established:

1. Construct an isomorphism $\psi_A : A \to \tilde{A}$.
2. Construct an isomorphism $\psi_\Phi : L_\Phi \to \tilde{L_\Phi}$.
3. Extend $\psi_\Phi$ to an isomorphism $\psi_L : L \to \tilde{L}$.
4. Combine $\psi_A$ and $\psi_L$ to an isomorphism $\psi : G \to \tilde{G}$.

In fact, $G$ and $\tilde{G}$ are isomorphic if and only if we succeed in Steps (1) & (2). Thus, if we just want to decide whether $G \cong \tilde{G}$, then Steps (3) & (4) need not to be carried out; moreover, it is not necessary to construct $\psi_A$: since $A$ and $\tilde{A}$ are groups of type $\text{PSL}_2$, we have $A \cong \tilde{A}$ if and only if $|A| = |\tilde{A}|$, which can be readily determined in our computational framework.

**4. Preliminary algorithms**

We list a few algorithms which are required later. One important result is the description of an algorithm to construct an abelian Sylow tower for a solvable group, if it exists. This is a key ingredient in [3], but in that work groups are input as multiplication tables; in our setting multiplication tables might be exponentially larger than the input, so we cannot use this work.
4.1. Constructive presentations and $\Omega$-complements. Let $\Omega$ be a set. An $\Omega$-group is a group $G$ on which the set $\Omega$ acts via a prescribed map $\theta : \Omega \to \text{Aut}(G)$. We first investigate the problem $\Omega$-ComplementAbelian: given an abelian normal $\Omega$-subgroup $M \trianglelefteq G$, decide whether $G = K \rtimes M$ for some $\Omega$-subgroup $K \leq G$, or certify that no such $K$ exists. Variations on this problem have been discussed in several places; the version we describe is based on a proof in [30, Proposition 4.5] which extends independent proofs by Luks and Wright in lectures at the U. Oregon.

We show in Proposition 4.3 that $\Omega$-ComplementAbelian has a polynomial time solution for solvable groups. The proof involves Luks’ constructive presentations [23, Section 4.2], which will also be useful later to equip solvable permutation groups with polycyclic presentations, see Lemma 4.7.

**Definition 4.1.** Let $G$ be a group and $N \trianglelefteq G$. A constructive presentation of a group $G/N$ is a free group $F_X$ on a set $X$, a homomorphism $\phi : F_X \to G$, a function $\psi : G \to F_X$, and a set $R \subseteq F_X$ such that $g^{-1}(g\psi\phi) \in N$ for every $g \in G$, and $N\psi^{-1} = \langle R^{F_X} \rangle$, the normal closure of $(R)$ in $F_X$.

This can be interpreted as follows: $\langle X \mid R \rangle$ is a generator-relator presentation of the group $G/N$, see [23, Lemma 4.1]: the homomorphism $\phi$ is defined by assigning the generators $X$ of $F_X$ to the generating set $S \subseteq G$. The function $\psi$ is in general not a homomorphism, and serves to writes elements of $G$ as a corresponding word in $X$. The next lemma discusses a constructive presentation for a subgroup of the holomorph $\text{Aut}(G) \times G$ of a group $G$.

**Lemma 4.2.** Let $G$ be an $\Omega$-group via $\theta : \Omega \to \text{Aut}(G)$, and write $g^w = g^{\theta x}$ for $g \in G$ and $w \in \Omega$. Let $\langle X \mid R \rangle$ with $\phi : F_X \to G$ and $\psi : G \to F_X$ be a constructive presentation of $G$. Let $\langle \Omega \mid S \rangle$ be a presentation for $A = \langle \Omega \theta \rangle \leq \text{Aut}(G)$. Then $\langle \Omega \cup X \mid S \times R \rangle$ is a presentation for $A \times G$ where

\[
S \times R = S \cup R \sqcup \{(x^{w})^{w} \psi \cdot (x^{-1})^{w} : x \in X, w \in \Omega \} \subset F_{\Omega \cup X}
\]

with embedding $\theta \sqcup \phi : \Omega \cup X \to A \times G, z \mapsto \begin{cases}
  z\theta & (z \in \Omega) \\
  z\phi & (z \in X)
\end{cases}$.

**Proof.** Without loss of generality, we can assume that $F_X = \langle X \rangle$, $F_\Omega = \langle \Omega \rangle$, and $F_\Omega, F_X \leq F_{\Omega \cup X}$. Let $K$ be the normal closure of $S \times R$ in $F_{\Omega \cup X}$. Recall that, by definition, if $x \in X$, then $x\phi\psi$ and $x$ define the same element in $G$ via $\phi$. It follows that if $w \in \Omega$ and $x \in X$, then $x^{w}, (x^{w})^{w} \psi \in F_{\Omega \cup X}$ define the same element in $A \times G$ via $\theta \sqcup \phi$: if $\alpha : F_{\Omega \cup X} \to A \times G$ is the homomorphism defined by $\theta \sqcup \phi$, then

\[
(x^{w})^{w} = ((w\theta)^{-1}, 1)(x\phi)(w\theta, 1) = (1, (x\phi)^{w}) = (1, (x\phi)^{w} \psi \phi) = ((x\phi)^{w} \psi)\alpha
\]

shows that $(x\phi)^{w} \psi (x^{-1})^{w} \in \ker \alpha$, so $K \leq \ker \alpha$. Now consider $N = KF_X$. From what is said above, if $w \in \Omega$ and $x \in X$, then $x^{w} = K(x\phi)^{w} \psi \leq N$, so $N^{w} = K^{w}F^{w}x \leq \langle K^{x} : x \in X \rangle = N$. This shows that $N \leq F_{\Omega \cup X}$; note that $K^{w} = K$ since $K$ is the normal closure in $F_{\Omega \cup X}$. Now set $C = KF_{\Omega}$. It follows that $F_{\Omega \cup X} = CN$, thus $H = F_{\Omega \cup X}/K = CN/K = (C/K)(N/K)$ and $N/K$ is normal in $H$. Since $C/K$ and $N/K$ satisfy the presentations for $A$ and $G$ respectively, von Dyck’s Theorem [26 (2.2.1)] implies that $H$ is a quotient of $A \times G$. To show that $H$ is isomorphic to $A \times G$ it suffices to notice that $A \times G$ satisfies the relations in $S \times R$ with respect to $\Omega \cup X$ and $\theta \sqcup \phi$.

As shown above, $K \leq \ker \alpha$. Since $H = F_{\Omega \cup X}/K$ is a quotient of the group $A \times G = F_{\Omega \cup X}\alpha$, it follows that $K = \ker \alpha$, and therefore $\langle \Omega \cup X \mid S \times R \rangle$ is a presentation for $A \times G$. \hfill \Box

We now show that $\Omega$-ComplementAbelian has a polynomial-time solution for solvable groups.

**Proposition 4.3.** Let $G$ be a solvable $\Omega$-group with abelian normal $\Omega$-subgroup $M \trianglelefteq G$. There is a polynomial time algorithm which decides whether $G = K \rtimes M$ for some $\Omega$-subgroup $K$, or certifies that no such $K$ exists.

**Proof.** Let $G$ be a quotient of a permutation group on $n$ letters, let $\theta : \Omega \to \text{Aut}(G)$ be a function, and let $M$ be an abelian $(\Omega \cup G)$-subgroup of $G$. We first describe the algorithm, then prove correctness.
We use the algorithm of [11] Lemma 4.11 to produce a constructive presentation for the solvable quotient \( G/M \) with data \( \langle X \mid R \rangle \) and maps \( \phi: X \to G \) and \( \psi: G \to F_X \). For each \( s \in \Omega \) and \( x \in X \), define
\[
w_{s,x} = ((x\phi)^s)\psi \cdot (x^s)^{-1} \in F_{\Omega \cup X}.
\]
Let \( \nu: X \to M \leq G \) be a function. Considering each \( w \in F_{\Omega \cup X} \) as a word in \( \Omega \sqcup X \), we denote by \( w(\phi\nu) \) the element in \( G \) where each symbol \( x \in \Omega \sqcup X \) in \( w \) has been replaced by \( (x\phi)(x\nu) \). Use Solve [18] Section 3.2 to decide if there is a function \( \nu: X \to M \), where
\[
(\forall w \in R : w(\phi\nu) = 1, \text{ and})
\]
\[
(\forall s \in \Omega, \forall x \in X : w_{s,x}(\phi\nu) = 1).
\]
If no such \( \nu \) exists, then report that \( M \) has no \( \Omega \)-complement; otherwise, return the group
\[
K = \langle (x\phi)(x\nu) : x \in X \rangle.
\]
We show that this is correct. Let \( A = \langle \Omega \theta \rangle \leq \text{Aut}(G) \) and let \( \langle \Omega \mid R' \rangle \) be a presentation of \( A \) with respect to \( \theta \), Lemma [22] shows that \( \langle \Omega \cup X \mid R' \rangle \) is a presentation for \( A \ltimes (G/M) \) with respect to \( \theta \sqcup \phi \); note that we need not to compute \( R' \).

First suppose that the algorithm returns \( K = \langle (x\phi)(x\nu) : x \in X \rangle \). As \( \langle x\phi : x \in X \rangle \subseteq KM \) we get that \( G = \langle x\phi : x \in X \rangle \leq KM \leq G \). Since \( w(\phi\nu) = 1 \) for all \( w \in R \) by (4.1), the group \( K \) satisfies the defining relations of \( G/M \cong K/(K \cap M) \), which forces \( K \cap M = 1 \), and so \( G = K \ltimes M \). By (4.1) and (4.2), the generator set \( \emptyset \emptyset \sqcup \{ (x\phi)(x\nu) : x \in X \} \) of \( \langle A, K \rangle \) satisfies the defining relations \( R' \ltimes R \) of \( (A \times G)/M \), and so \( \langle A, K \rangle \) is isomorphic to a quotient of \( (A \times G)/M \) where \( K \) is the image of \( G/M \). This shows that \( K \) is normal in \( \langle A, K \rangle \), in particular, \( (K^\Omega) \leq K \). This proves that if the algorithm returns a subgroup, then the output is correct.

Conversely, suppose \( G = K \ltimes M \) such that \( K^\Omega \subseteq K \) and there is an idempotent endomorphism \( \tau: G \to G \) with kernel \( M \) and image \( K \). We must show that in this case equations (4.1) and (4.2) have a solution, so that the algorithm returns a complementary \( \Omega \)-subgroup to \( M \). Define the map \( \nu: X \to M \) by \( x\nu = (x\phi)^{-1}(x\phi\tau) \). Now \( K = G\tau = \langle (x\phi)(x\nu) : x \in X \rangle \) is isomorphic to \( G/M \) via \( (x\phi)(x\nu) \mapsto x\phi M \), hence \( \langle (x\phi)(x\nu) : x \in X \rangle \) satisfies the relations \( R \). Moreover, we have \( K^\Omega \leq K \), so the isomorphism \( K \cong G/M \) defined by \( (x\phi)(x\nu) \mapsto x\phi M \) extends to \( A \ltimes K \to A \ltimes (G/M) \); thus, for all \( s \in \Omega \) and \( x \in X \) we have \( w_{s,x}(\phi\nu) = 1 \). The claim on the complexity follows since we only applied polynomial-time algorithms.

We will also need to find direct complements; we follow the algorithm in [30] Theorem 4.8. The analysis has not appeared in print so we include its proof.

**Proposition 4.4.** Let \( G \) be an \( \Omega \)-group and let \( U, V \leq G \) be normal \( \Omega \)-subgroups with \( U \leq V \). There is a polynomial time algorithm which decides whether \( V/U \) is a direct \( \Omega \)-factor of \( G/U \) and if so, returns a direct complement.

**Proof.** First compute \( C/U = C_{G/U}(V/U) \) via [17] P6, and test whether \( G = \langle C, V \rangle \), for example, by computing group orders. If \( G \not\cong \langle C, V \rangle \), then report \( V/U \) is not a direct \( \Omega \)-factor of \( G/U \). Otherwise, compute the center \( Z(V/U) \) via [17] P6 and use \( \Omega \)-ComplementAbelian to compute a \( \Omega \)-complement \( K/U \) to \( Z(V/U) \) in \( C/U \), or, if none exists, report that \( V/U \) is not a direct \( \Omega \)-factor of \( G/U \). We prove this this is correct. If \( G/U = K/U \times V/U \) is a direct product of \( \Omega \)-subgroups with \( U \leq K \leq G \), then \( K/U \leq C_{G/U}(V/U) = C/U \) and \( K/U \) complements \( V/U \cap C/U = Z(V/U) \); the algorithm constructs such an \( \Omega \)-complement. Conversely, if we find a \( \Omega \)-complement \( K/U \) to \( Z(V/U) \) in \( C/U \), then we have \( \langle K/U \rangle \cap \langle V/U \rangle = U/U \), and \( K/U \) and \( V/U \) centralizes each other; therefore so long as \( G/U = \langle K/U, V/U \rangle \), the \( \Omega \)-subgroup \( K/U \) is a direct complement to \( V/U \) in \( G/U \). We only applied polynomial-time algorithms.
4.2. Sylow towers and socles. Following [26, Section 9.1], a set of Sylow subgroups, one for each prime dividing the group order, is a Sylow basis if any two such subgroups $U$ and $V$ are permutable, that is, if $UV = VU$; every solvable group admits a Sylow basis. A group $L$ has an abelian Sylow tower if there exists a Sylow basis $\{Y_1, \ldots, Y_\ell\}$ of abelian groups such that $L = Y_1 \times \cdots \times Y_\ell$.

**Proposition 4.5.** Let $L$ be a solvable group which has an abelian Sylow tower. There is a polynomial-time algorithm that computes a Sylow tower $L = Y_1 \times \cdots \times Y_\ell$.

**Proof.** Compute and factorize $|L| = p_1^{e_1} \cdots p_\ell^{e_\ell}$. By assumption, $L$ has a normal Sylow subgroup; we run over the prime factors $p_i$ and compute a Sylow $p_i$-subgroup $P_i$ until $[P_i, L]$ is contained in $P_{i-1}$; if so, set $Y_i = P_i$. Since all Sylow subgroups are abelian, we use $\Omega$-ComplementAbelian to compute a complement $K \leq L$ to $Y_\ell$. By construction, $L = K \times Y_\ell$, and $|K|$ and $|Y_\ell|$ are coprime. Since $K \cong L/Y_\ell$ has an abelian Sylow tower, we can recurse with $K$ and compute a Sylow basis for $K$. We only apply polynomial-time algorithms at most $\sum_{i=1}^\ell i \in O((\log |G|)^2)$ times. \hfill \qed

We also need the ability to compute the socle of a solvable group. Algorithms for that have been given for permutation groups by Luks [17, P15] and for black-box solvable groups by Höfling [15]. Höfling’s algorithm reuses the ingredients given above for computing complements, which we will later use to construct Frattini subgroups. So we pause to note the complexity of Höfling’s algorithm.

**Proposition 4.6.** Generators for the socle of a solvable group can be computed in polynomial-time.

**Proof.** Let $L$ be a solvable group, treated as an $L$-group under conjugation action. Use [17, P11] to compute a chief series $1 = N_0 \rhd N_1 \lhd \cdots \lhd N_r = L$; in particular, $N_1$ is a minimal normal subgroup of $L$. We set $S_1 = N_1$, and for each $i > 1$ compute a direct $L$-complement $S_i$ to $N_{i-1}$ in $N_i$ (so $S_i \leq L$); set $S_1 = 1$ if this does not exist. To this end, we proceed as follows: we use the algorithm of Proposition 4.4 to find an $L$-subgroup $T \leq N_i$ such that $N_i = T \times N_{i-1}$; if no such $T$ exists, then we set $S_i = 1$. As $T$ is normal in $L$, set $S_i = T$. Once this is done for $i = 1, \ldots, r$, return $S_1 \times \cdots \times S_r$. The correctness of this algorithm follows from [15, Proposition 5] where it is shown that $soc(L) = S_1 \times \cdots \times S_r$. We only apply algorithms assumed or shown to be polynomial-time. \hfill \qed

4.3. Computing polycyclic constructive presentations. Constructions of polycyclic presentations from solvable permutation groups are done by various means, sometimes invoking steps (such as collection) whose complexities are difficult to analyze; see for instance [28, p. 166]. In that approach, one first chooses a polycyclic generating sequence $x_1, \ldots, x_s$ and then uses the constructive membership testing mechanics of permutation groups to sift the relations $x_i^{p_i}$ and $x_i^{f_i}$ into words in the $x_k$. That process leaves the resulting words in arbitrary order, rather than in collected order, that is, we need $x_i^{p_i} = x_{i+1}^{f_{i+1}} \cdots x_s^{f_s}$, but all we can know is that $x_i^{p_i}$ is a word in $x_{i+1}, \ldots, x_s$ in no particular order. Hence, in that approach, a final step of rewriting must be applied to get the words in normalised (collected) form; this comes at a cost, see the discussion in [21]. We present an alternative.

**Lemma 4.7.** A polycyclic constructive presentation for a solvable group can be computed in polynomial-time.

**Proof.** Let $L$ be a solvable group. Use [17, P11] to construct a chief series $L = L_0 > \cdots > L_s = 1$. Since $L$ is solvable, each section $L_i/L_{i+1}$ is isomorphic to $C_p^{f_i}$ for some prime $p_i$ and $f_i \geq 1$. In the following, set $d(i) = f_0 + \cdots + f_{i-1}$ for $i > 0$, and denote by $F_m$ with $m \in \mathbb{N}$ the free group on $x_1, \ldots, x_m$. We work with a double recursion through $L/L_i$ and within each factor $L_i/L_{i+1}$.

For the inner recursion we assume $L_i/L_{i+1} \cong C_{p_i}^{f_i}$ and want to create a constructive presentation for this group. Note that every chief series of $L_i/L_{i+1}$ is a composition series, so we use [17, P11] to find generators $g_1, \ldots, g_{f_i}$ of a composition series $L_{i0} > L_{i1} > \cdots > L_{if_i} = L_{i+1}$ such that each $L_{ij} = \langle g_{j+1}, L_{j+1} \rangle$ and $\langle x_j \mid x_j^{p_i} \rangle$ is a presentation for $L_{ij}/L_{ij+1} \cong C_{p_i}$. To make this
constructive, use $\psi_j : L_{ij} \to F_1$, defined by sending $gL_{ij+1} \in L_{ij}/L_{ij+1}$ to $x_1^i$ where $g^{-1}g_{ij+1}^e \in L_{ij+1}$. Since $e \leq p_1$ is less than the size of the input, $\psi_j$ can be evaluated in polynomial time. This yields a constructive polycyclic presentation of $L_{ij}/L_{ij+1}$. Now suppose by induction we have a constructive polycyclic presentation $F_j \to L_{i0}/L_{ij}$. Since we also have a constructive polycyclic presentation of $F_1 \to L_{i0}/L_{i(j+1)}$, we obtain a constructive presentation $F_{j+1} \to L_{i0}/L_{i(j+1)}$ by Luks’ constructive presentation extension lemma [23, Lemma 4.3]. In that new presentation, every polycyclic relation (for example $x_k^e = x_{k+1} \cdots x_k^e$ or $x_k^e = x_{k+1}^e \cdots x_k^e$) is appended with an element of $(x_{j+1})$, and so the resulting relations are in collected form. Thus, at the end of this inner recursion we have a polycyclic constructive presentation for the elementary abelian quotients $L_i/L_{i+1}$.

Now consider the outer recursion. In the base case $i = 0$ we apply the above method to create a constructive polycyclic presentation of $L_0/L_1$. Now suppose by induction we have a polycyclic constructive presentation of $L_i/L_i$ with maps $\varphi : F_{d(i)} \to L_i/L_i$ and $\psi : L_i/L_i \to F_{d(i)}$ which can be applied in polynomial time. As in the base case, we construct a polycyclic constructive presentation with maps $\varphi' : F_{j+1} \to L_i/L_{i+1}$ and $\psi' : L_i/L_{i+1} \to F_{j+1}$. Luks’ extension lemma now makes a constructive presentation for $L_i/L_{i+1}$ with maps $\psi' : F_{d(i+1)} \to L_i/L_{i+1}$ and $\psi : L_i/L_{i+1} \to F_{d(i+1)}$. In this process, relations of $L_i/L_i$ of the form $x_j^i = x_1^i \cdots x_{d(i)}^i$ and $x_j^i = x_{k+1}^i \cdots x_{d(i)}^i$ are appended with normalised words in $L_i/L_{i+1}$, so these continue to be in collected form. We also add the polycyclic relations for $L_i/L_{i+1}$, so the extended constructive presentation is polycyclic.

\section{5. Isomorphism testing of cube-free groups: solvable Frattini-free groups}

We now deal with Step (2) of our algorithm as described in Section 3.2. Using the notation of Section 4 throughout the following $L$ and $\bar{L}$ are finite solvable groups of cube-free order, and we consider their Frattini-free quotients $L_\Phi = L/\Phi(L)$ and $\bar{L} = \bar{L}/\Phi(\bar{L})$. Recall that $L_\Phi = K \times \soc(L_\Phi)$ with $\soc(L_\Phi) = B \times C$ where $|B| = b$ and $|C| = c^2$ with $b$ and $c$ square-free; analogously for $\bar{L}_\Phi$. In the remainder of this section we describe how to construct an isomorphism $L_\Phi \to \bar{L}_\Phi$; our construction fails if and only if the two groups are not isomorphic.

\textbf{Proposition 5.1.} There is a polynomial-time algorithm given a solvable Frattini-free group $L_\Phi$ of cube-free order, returns generators for the decomposition into subgroups $(K, B, C)$ described above, along with isomorphisms $B \to \prod_{i=1}^s \mathbb{Z}/p_i$ and $C \to \prod_{j=s+1}^m (\mathbb{Z}/p_j)^2$, and a representation $K \to \Aut(B \times C) \to \prod_{i=1}^s \GL_1(p_i) \times \prod_{j=s+1}^m \GL_2(p_j)$ induced by conjugation of $K$ on $B \times C$.

\textbf{Proof.} Use the algorithms of Propositions 4.6 & 4.3 to compute generators for $\soc(L_\Phi)$ and for a complement $K$ to $\soc(L_\Phi)$ in $L_\Phi$. Then use the algorithm of Proposition 4.5 to decompose $\soc(L)$ as a direct product of its Sylow subgroups. Using the decomposition series of each Sylow subgroup, we obtain the decomposition $\soc(L_\Phi) = B \times C$ along with primary decompositions of $B = \prod_{i=1}^s Y_i$ and $C = \prod_{j=s+1}^m Y_j$. We can further produce isomorphisms $\beta_i : Y_i \to \mathbb{Z}/p_i$ and $\kappa_j : Y_j \to (\mathbb{Z}/p_j)^2$, for example, by using our results from [11, Section 3], based on Karagiorgos \& Poulakis [19]. Given standard representations for $\Aut(\mathbb{Z}/p_i) \cong (\mathbb{Z}/p_i)^\times$ and $\Aut((\mathbb{Z}/p_j)^2) = \GL_2(p_j)$, compose with $\beta_i$ and $\kappa_j$ respectively to produce an isomorphism $\tau : \Aut(B \times C) \to \prod_{i=1}^s \GL_1(p_i) \times \prod_{j=s+1}^m \GL_2(p_j)$.

Finally, define $\pi : K \to \Aut(B \times C)$ by $(bc)(k)\pi = b^c \cdot \pi$, so $\pi \tau$ is the required map from $K$.

The correctness of this algorithm is apparent. The claim on the timing of the first portion follows since we only invoked $O(\log |L_\Phi|)$ many polynomial-time algorithms. We can apply the algorithms of [11, Section 3] to construct an isomorphism in polynomial time since $|Y_i| = p_i$ and $|Y_j| = p_j^2$,
and both $p_j$ and $p_{\bar{j}}$ are bounded by the size of the permutation domain $\Omega$ of $L$. So the complexity of the results used from [11] is sufficient. Our assumption is that all groups here are permutation groups: in the case of the groups $\prod_{j=s+1}^m \GL_2(p_j)$, we can treat the matrices as permutations of pairs $\bigcup_{j=s+1}^m \{(a,b) | a, b \in \mathbb{Z}/p_j\}$; this domain has size $O(p_{s+1} + \cdots + p_m) \subset O(|\Omega| \log |L|)$, so is polynomial in the input size.

To simplify the exposition, we make the following convention and identify

$$B = \tilde{B} = \prod_{i=1}^s \mathbb{Z}/p_i \quad \text{and} \quad C = \tilde{C} = \prod_{i=s+1}^m (\mathbb{Z}/p_i)^2.$$ 

Recall from Section 3 that the conjugation action of $K$ on $B \times C$ is faithful. Hence, we also treat $K$ and $\tilde{K}$ as subgroups of

$$\Aut(B \times C) = \prod_{i=1}^s \GL_1(p_i) \times \prod_{i=s+1}^m \GL_2(p_i).$$

For $j = 1, \ldots, m$ denote by $K_j$ and $\tilde{K}_j$ the projections of $K$ and $\tilde{K}$, respectively, into the $j$-th factor of $\Aut(B \times C)$; thus $K_j$ and $\tilde{K}_j$ describe the conjugation action of $K$ and $\tilde{K}$, respectively, on the Sylow $p_j$-subgroup $Y_j \leq B \times C$.

Gashütz has shown that $L_\Phi \cong L_\Phi$ if and only if $K$ and $\tilde{K}$ are conjugate in $\Aut(B \times C)$, see [9, Lemma 9]; hence, the isomorphism problem reduces to finding an element $\alpha \in \Aut(B \times C)$ with $\alpha^{-1} K \alpha = \tilde{K}$. Once such an $\alpha$ is found, the isomorphism $\psi_\Phi$ can be defined as follows: writing the elements of $L_\Phi = K \times (B \times C)$ and $\tilde{L}_\Phi = \tilde{K} \times (B \times C)$ as $(k, b, c)$ and $(\tilde{k}, b, c)$, respectively, we set

$$\psi_\Phi : L_\Phi \to \tilde{L}_\Phi, \quad (k, b, c) \mapsto (\alpha^{-1} k \alpha, b^\alpha, c^\alpha).$$

Our construction of $\alpha$ depends very much on the dimension 2 case; in particular, we use a classification of J. Gierster (1881) of the subgroups of $\GL_2(p)$, extracted from [13] Theorems 5.1–5.3.

**Lemma 5.2.** Let $p$ be an odd prime and let $K \leq \GL_2(p)$ be a solvable cube-free $p'$-subgroup.

a) If $K$ is reducible, then $K$ is conjugate to a subgroup of diagonal matrices.

b) If $K$ is irreducible and abelian, then $K$ is conjugate to $\langle s^{(p^2-1)/r} \rangle$ for some $r \mid p^2 - 1$, where $s$ is a generator of a Singer cycle in $\GL_2(p)$, that is, $(s) \cong C_{p^2-1}$.

c) If $K$ is irreducible and non-abelian, then there are three possibilities. First, $K$ might be conjugate to $G_2 \times G_2'$ where $G_2'$ is an odd order diagonal (but non-scalar) subgroup and $G_2$ is one of

$$\langle (1, 0), \, (0, 1) \rangle, \quad \langle (0, 1), \, (1, 0) \rangle, \quad \langle (0, -1), \, (1, 0) \rangle, \quad \langle (0, 1), \, (1, 1) \rangle, \quad \langle (0, -1), \, (1, 0) \rangle,$$

with $z \in \mathbb{Z}/p$ of order 4 (if it exists). Second, $K$ might be conjugate to $\langle S, t \rangle$ where $S$ is a subgroup of a Singer cycle $\langle s \rangle$ and $t$ is an involution such that $N_{\GL_2(p)}(\langle s \rangle) = \langle s, t \rangle$. Third, $K$ might be conjugate to $\langle S, ts^{2a} \rangle$ where $S \leq \langle s \rangle$ has even order and $p - 1 = 4l$ with $l$ odd.

In particular, $N_{\GL_2(p)}(K)/C_{\GL_2(p)}(K)$ is solvable.

We further need an algorithm of Luks & Miyazaki’s [20] that demonstrates how to decide conjugacy of subgroups in solvable permutation groups in time polynomial in the input size.

**Theorem 5.3.** Let $G$, $K$, and $\tilde{K}$ be groups with

$$K, \tilde{K} \leq G = \langle S \rangle = \prod_{i=1}^n \GL_2(p_i),$$

where $K$ and $\tilde{K}$ are solvable groups of equal cube-free order coprime to $p_1 \cdots p_n$. One can decide in polynomial time whether $K$ is conjugate to $\tilde{K}$ and produce a conjugating element, if it exists.

**Proof.** As above, let $K_i$ and $\tilde{K}_i$ be the projections of $K$ and $\tilde{K}$, respectively, to the factor $\GL_2(p_i)$. For each $i$, based on the classification given in Lemma 5.2 we apply basic linear algebra methods...
to solve for $\alpha_i \in \text{GL}_2(p_i)$ such that $K_i^{\alpha_i} = \tilde{K}_i$; we also construct $N_i = N_{\text{GL}_2(p_i)}(\tilde{K}_i)$ based on Lemma 5.2. If we cannot find a particular $\alpha_i$, then $K$ and $\tilde{K}$ are not conjugate and we return that. Once all the $\alpha_i$ have been computed, we replace $K$ by $K = K^{\alpha_1 \cdots \alpha_n}$, so that we can assume that $K_i = \tilde{K}_i$ for all $i$. Note that $K$ and $\tilde{K}$ are conjugate if and only if they are conjugate in $N = \prod_{i=1}^n N_i$, which is solvable by Lemma 5.2. Now we apply the algorithm of [20, Theorem 1.3(ii)] to solve for $\beta \in N$ such that $K^{\beta} = \tilde{K}$, and return $\alpha_1 \cdots \alpha_n \beta$. If we cannot find such a $\beta$, then $K$ and $\tilde{K}$ are not conjugate, and we return false. Lastly, we comment on the timing. Note that we can also locate appropriate $\alpha_i$ by a polynomial-time brute-force search in $\text{GL}_2(p_i)$: the latter has order at most $p_i^4 \leq d^4$, where $d$ is the size of the permutation domain of $G$. We make a total of $n \leq \log |G|$ such searches, followed by the polynomial-time algorithm of [20]. The claim follows.

6. ISOMORPHISM TESTING OF CUBE-FREE GROUPS: SOLVABLE GROUPS

Throughout this section $L$ and $\tilde{L}$ are finite solvable groups of cube-free order, given as permutation groups. To decide isomorphism, we first want to use the algorithm of Section 5 to determine whether the Frattini quotients $L_\Phi$ and $\tilde{L}_\Phi$ are isomorphic. For this we need the Frattini subgroups.

6.1. Frattini subgroups. Since we assume permutation groups as input, we need a polynomial-time algorithm to compute Frattini subgroups of solvable permutation groups of cube-free order. A candidate algorithm has been provided by Eick [8, Section 2.4] for groups given by a polycyclic (pc) presentation. To adapt to a permutation setting we have two choices: replace every step of that algorithm with polynomial-time variants for permutation groups, or apply the algorithm in-situ by appealing to a two-way isomorphism between our original permutation group and a constructive pc-presentation as afforded to us by Lemma 4.7. Note that for the efficiency of the inverse isomorphism, elements in a pc-group are straight-line programs (SLPs) in the generators, so evaluation is determined on the generators and computed in polynomial time. Thus, whenever we take products in the pc-group, we actually carry out permutation multiplications and sift these into the polycyclic generators by applying the isomorphism back to the pc-group. This avoids the potential exponential complexity of collection in pc-groups, see the discussion in [21]. That the algorithm in [8, Section 4.2] uses a polynomial number of pc-group operations follows by considering its major steps. It relies on constructing complements of abelian subgroups (shown in Proposition 4.3 to be in polynomial time), and it applies also module decompositions (which can be done in polynomial time see [20, Theorem 3.7 & Section 3.5]), and finally computing cores [17, P5]. Therefore Eick’s algorithm is in fact a polynomial-time algorithm for groups of permutations, and we cite it as such in what follows.

Once $\Phi(L)$ and $\Phi(\tilde{L})$ have been constructed, we can compute the quotients $L_\Phi$ and $\tilde{L}_\Phi$, see [17], and use the algorithms of Section 5 to test isomorphism. If we have determined that $L_\Phi \not\cong \tilde{L}_\Phi$, then we can report that $L \not\cong \tilde{L}$. Thus, in the following we assume we found an isomorphism $\varphi: L_\Phi \rightarrow \tilde{L}_\Phi$, so we also know that $L \cong \tilde{L}$ by Section 3. In the next sections we describe how to construct an isomorphism $\hat{\varphi}: L \rightarrow \tilde{L}$ such that $\hat{\varphi}$ factors through $\varphi$ in the sense that $\Phi(\hat{\varphi}(g \varphi)) = (\Phi(L)g)\varphi$ for all $g \in L$. This condition is what allows us to not only solve for some isomorphism between $L$ and $\tilde{L}$, but to also lift generators for the automorphism group of $L$ and thus prescribe (generators for) the entire coset of isomorphisms $L \rightarrow \tilde{L}$. Our approach to computing $\hat{\varphi}$ is to work with each prime divisor of $|\Phi(L)|$. We begin with a key observation about these primes and recall the Frattini extension structure of groups of cube-free order.

6.2. Frattini extension structure. As above, write $A_1 \times \cdots \times A_s$ for any subdirect product of groups $A_1, \ldots, A_s$. For a group $Y$ and prime $p$ dividing $|Y|$ let $Y_p$ be a Sylow $p$-subgroup of $Y$. It follows from [26, 9.2] that every finite solvable group has a Sylow basis, and it follows from [7/25] that every
solvable cube-free group $Y$ has one of the following abelian Sylow towers:

$$Y = \begin{cases} Y_{r_1} \times Y_{r_2} \times \ldots \times Y_{r_t} & \text{if } |Y| \text{ odd} \\ Y_2 \times Y_{r_1} \times Y_{r_2} \times \ldots \times Y_{r_t} & \text{if } |Y| \text{ even, } Y_2 \not\sim Y \\ Y_{r_1} \times Y_{r_2} \times \ldots \times Y_{r_t} \times Y_2 \ (\text{with } Y_2 = C_2^2) & \text{if } |Y| \text{ even, } Y_2 \leq Y \end{cases}$$

where $r_1 < \ldots < r_t$ are the odd prime divisors of $|Y|$ and $\{(Y_2), Y_{r_1}, \ldots, Y_{r_t}\}$ forms a Sylow basis of $Y$. Proposition 4.5 provides an algorithm to construct such a Sylow tower.

**Lemma 6.1.** Let $L$ be Frattini-free and solvable, and let $Y^*$ be a cube-free Frattini extension of $Y$, that is, $Y^*/\Phi(Y^*) \cong Y$. If $p \nmid |\Phi(Y^*)|$, then $Y_p^* \cong \alpha_p$, otherwise $Y_p^* \cong C_p$ and $Y_p^* \cong C_{p^2}$.

**Proof.** Recall that every prime dividing $|\Phi(Y^*)|$ must divide $|Y|$, thus $\Phi(Y^*)$ is square-free and the Sylow tower of $Y^*$ looks similar to that of $Y$, where $Y_p^* \cong \alpha_p$ if $p \nmid |\Phi(Y^*)|$, and $Y_p^* \cong C_p$ and $Y_p^* \text{ abelian of order } p^2$ otherwise. We prove that $Y_p^* \cong C_{p^2}$. We use the previous notation and consider $M = \Phi(Y^*) = C_{p_1} \times \ldots \times C_{p_n}$ as a $Y$-module. It is shown in [9, Theorem 12] that $Y^*$ is a subdirect product of Frattini extensions of $Y$ by $C_{p_i}$. Thus, to prove the lemma, it suffices to consider $M = C_p$ for some prime $p$. First, suppose that $p = r_1$ is odd. In this case, $Y_p^* \cong C_p$ and $Y_p^*$ is abelian of order $p^2$. Suppose, for a contradiction, that $Y_p^* \cong C_{p^2}$. It follows from [9, Lemma 5 & Theorem 14] that $Y^*$ is a non-split extension of $Y$ by $M$ such that $N_Y(Y_p)$ acts on $M$ as on $Y_p$. This implies the following: considering $Y_p^* = Y_{r_1}^* = (\mathbb{Z}/p)^2$ as an $\mathbb{Z}/p$-space, there is a basis $\{m, y\}$ such that $M = \langle m \rangle$ and every $g \in (Y_{r_1}^* \times \ldots \times Y_{r_t}^*)$ acts on that space as a matrix $\tilde{g} = \begin{pmatrix} \alpha & \beta \\ 0 & \alpha \end{pmatrix}$ for some $\alpha \in (\mathbb{Z}/p)^\times$ and $\beta \in \mathbb{Z}/p$. Since $|Y^*|$ is cube-free, $g$ has order coprime to $p$, and hence $\beta = 0$, that is, $g$ acts diagonally on $Y_p^*$. Moreover, $W = Y_{r_{i+1}}^* \times \ldots \times Y_{r_t}^*$ centralizes $Y_p^*$ modulo $W$. In conclusion, no nontrivial element in $Y_p^*$ is a non-generator of $Y^*$, contradicting $\Phi(Y^*) \leq Y_p^*$, see [10 Proposition 2.44]. This contradiction proves $Y^* \cong C_{p^2}$. Lastly, suppose $M = C_2$; in this case $Y^* \cong C_2$ and $Y^* = Y_2^* \times Y_{r_1}^* \times \ldots \times Y_{r_t}^*$. If $Y_2^* \cong C_2^2$, then the same argument shows that no nontrivial element in $Y_2^*$ is a non-generator of $Y^*$, contradicting $\Phi(Y^*) \leq Y_2^*$, Thus, $Y_2^* \cong C_4$. \(\square\)

6.3. **Constructing the isomorphism.** Recall that $L \cong \tilde{L}$ if and only if the isomorphism $\psi_{\Phi}$ in Step (2) exists. Suppose $\psi_{\Phi}$ has been constructed as described in Section 5 that is, we know that $L \cong \tilde{L}$. As explained in the proof of Lemma 6.1 the groups $L$ and $\tilde{L}$ are iterated Frattini extensions of $L_0$ and $L_0$, respectively, by cyclic groups of prime order; cf. [9, Definition 4]. Starting with $\psi_{\Phi}$, we iteratively construct isomorphisms of these Frattini extensions until eventually we obtain an isomorphism $L \to \tilde{L}$. Thus, we consider the following situations: let $Y$ and $\tilde{Y}$ be two solvable cube-free groups and let $Y^*$ and $\tilde{Y}^*$ be cube-free Frattini extensions of $Y$ and $\tilde{Y}$, respectively, by $M = C_p$. We assume that we have an isomorphism $\varphi: Y \to \tilde{Y}$; we know that $Y^* \cong \tilde{Y}^*$, and we aim to construct an isomorphism $Y^* \to \tilde{Y}^*$. The following preliminary lemma will be handy.

**Lemma 6.2.** Let $G$ be a group and $P, Q \leq G$ such that $P$ is a cube-free $p$-group and $Q = \langle w \rangle$ is cyclic of order $q^2$, for distinct primes $p$ and $q$. Suppose $PQ = QP$ and $A = \langle w^q \rangle$ is normal in $PQ$.

a) We have $PQ = P \times Q$ or $PQ = Q \times P$.

b) If $PQ = Q \times P$, then $A$ acts trivially on $P$.

c) If $PQ = P \times Q$, then the action of $P$ on $Q$ is uniquely determined by its action on $Q/A$.

**Proof.** Since $PQ$ is cube-free, part a) follows from the structure results mentioned in Section 6.2. For part b), note that $Q$ and $Q/A$ both act on $P$; this forces that $A$ acts trivially on $P$. Now consider part c). Recall that $\text{Aut}(Q)$ is cyclic of order $(q - 1)$, generated by $\beta: Q \to Q$, $w \mapsto w^k$, where $k$ is some primitive root modulo $q^2$. Since $PQ$ is cube-free, the element $g \in P$ acts on $Q$ via an automorphism $\alpha \in \text{Aut}(Q)$ of order coprime $q$. Thus, $\alpha$ lies in the subgroup $T \leq \text{Aut}(Q)$ of order
$q - 1$, and there is a unique $e \in \{1, \ldots, q - 1\}$ such that $\alpha = (\beta^q)^e$. Now $(wA)^{\alpha} = (wA)^i$ with $i \in \{0, \ldots, q - 1\}$ yields $i = k^q \mod q$. Since $k^q$ is a primitive root modulo $q$, it follows that for any given $i \in \{1, \ldots, q - 1\}$ there is a unique $e \in \{1, \ldots, q - 1\}$ such that $i \equiv k^q \mod p$, hence for a given $i \in \{0, \ldots, q - 1\}$ there is a unique automorphism $\alpha \in \text{Aut}(Q)$ with $(wA)^{\alpha} = (wA)^i$.

**Proposition 6.3.** Let $Y$ and $\tilde{Y}$ be two solvable cube-free groups and let $Y^*$ and $\tilde{Y}^*$ be cube-free Frattini extensions of $Y$ and $\tilde{Y}$, respectively, by a group isomorphic to $C_p$. Algorithm 1 is a polynomial-time algorithm which, given an isomorphism $\varphi : Y \to \tilde{Y}$, returns an isomorphism $\tilde{\varphi} : Y^* \to \tilde{Y}^*$.

**Proof.** We compute the Frattini subgroups of $Y^*$ and $\tilde{Y}^*$, and the Sylow $p'$-subgroups $A \leq \Phi(Y^*)$ and $\tilde{A} \leq \Phi(\tilde{Y}^*)$, respectively, see Section 6.1. By assumption, $A \cong \tilde{A} \cong C_p$, and we can assume that $Y = Y^*/A$ and $\tilde{Y} = \tilde{Y}^*/\tilde{A}$. As explained above, the existence of $\varphi : Y \to \tilde{Y}$ implies that $Y^*$ and $\tilde{Y}^*$ are isomorphic. Use the algorithm of Proposition 4.5 to construct a Sylow tower $Y^* = Y_1^* \times \cdots \times Y_n^*$; for each $j$ let $p_j$ be a prime such that $Y_j^*$ is a Sylow $p_j$-subgroup. Let $p = p_i$, and recall from Lemma 6.1 that $Y_i^*$ is cyclic; find a generator $Y_i^* = \langle a \rangle$ and note that $A = \langle a^p \rangle \leq Y_i^*$. For every $j$ define $Q_j = \prod_{k \neq j} Y_k^*$, this is a Hall $p_j'$-subgroup of $Y^*$. Such a set of Hall $r'$-subgroups (one for each prime divisor $r$ of the group order) is called a Sylow system in [26 Section 9.2]); in particular, we can recover each $Y_j^*$ as $Y_j^* = \cap_{k \neq i} Q_k$.

Since $Y_1^*, \ldots, Y_n^*$ form a Sylow tower of $Y^*$, every $x \in Y^*$ has a unique factorization $x = ha^e$ where $h \in H = Q_i$ and $a^e \in Y_i^*$ with $0 \leq e \leq p^2 - 1$; we will use this decomposition later when we define an isomorphism $\tilde{\varphi} : Y^* \to \tilde{Y}^*$. We will construct $\tilde{\varphi}$ via a Sylow basis of $\tilde{Y}^*$ which is compatible with the above Sylow basis of $Y^*$; we explain below what this means.

Let $\Gamma : Y^* \to Y^*/A = Y$ be the natural projection, so that $\{Q_1 \Gamma \varphi, \ldots, Q_n \Gamma \varphi\}$ forms a Sylow system of $Y^*/\tilde{A}$. For each $j$ we define $\tilde{Q}_j \leq \tilde{Y}^*$ to be the full preimage of $Q_j \Gamma \varphi$ under the natural projection $\tilde{\Gamma} : \tilde{Y}^* \to \tilde{Y}^*/\tilde{A} = \tilde{Y}$. Clearly, if $j \neq i$, then $\tilde{Q}_j$ is a Hall $p_j'$-subgroup of $\tilde{Y}^*$. Moreover, $\tilde{Q}_i = \tilde{H} \times \tilde{A}$ where $\tilde{H}$ is some Hall $p'$-subgroup of $\tilde{Q}_i$ and of $\tilde{Y}^*$; we compute $\tilde{H}$ in $\tilde{Q}_i$ by first computing $\tilde{A} \leq \tilde{Q}_i$ as a Sylow $p'$-subgroup and then $\tilde{H}$ as a complement to $\tilde{A}$ in $\tilde{Q}_i$. We define $\tilde{Y}_i^* = \cap_{k \neq i} \tilde{Q}_k$ and

$$\tilde{Y}_j^* = \tilde{H} \cap \cap_{k \neq j} \tilde{Q}_k \quad \text{for each } j \neq i.$$  

It follows from [26 9.2.9] that $\{\tilde{Y}_1^*, \ldots, \tilde{Y}_n^*\}$ is a set of pairwise permutable Sylow subgroups with $Y_j^* \tilde{\Gamma} = Y_j^* \Gamma \varphi$ for all $j$. In particular, we can apply Lemma 6.1 and it follows from our construction that for all $u \neq v$ we have $\tilde{Y}_u Y_v^* = \tilde{Y}_u^* \times \tilde{Y}_v^*$ if and only if $Y_u^* Y_v^* = Y_u^* \times Y_v^*$, and $\tilde{Y}_u^* \tilde{Y}_v^* = \tilde{Y}_v^* \times \tilde{Y}_u^*$ if and only if $Y_u^* Y_v^* = Y_v^* \times Y_u^*$. We say that these two Sylow bases are compatible.

Let $\pi$ and $\tilde{\pi}$ be the restriction of $\Gamma$ and $\tilde{\Gamma}$ to $H$ and $\tilde{H}$, respectively; note that $\pi : H \to HA/A$ and $\tilde{\pi} : \tilde{H} \to \tilde{H} A/\tilde{A}$ are isomorphisms, and we define an isomorphism $\pi \tilde{\pi}^{-1} : H \to \tilde{H}$ via

$$H = H/(H \cap A) \xrightarrow{\pi} HA/A \xrightarrow{\varphi} \tilde{H} A/\tilde{A} \xrightarrow{\tilde{\pi}^{-1}} \tilde{H}/\tilde{H} \cap \tilde{A} = \tilde{H}.$$  

Note that in defining $\tilde{\pi} : h \mapsto \tilde{A} h$, we identify generators of $\tilde{H}$ with generators of $\tilde{H} A/\tilde{A}$; as elements of $\tilde{H} A/\tilde{A}$ are presumed throughout to be words (or SLPs) in the generators, we can compute preimages of $\tilde{\pi}$. This affords us an implementation of $\tilde{\pi}^{-1}$.

Recall that $Y_i^* = \langle a \rangle$, and choose a generator $\tilde{a} \in \tilde{Y}_i^*$ such that

$$a \Gamma \varphi = \tilde{a} \tilde{\Gamma}.$$  

We can now construct an isomorphism $\tilde{\varphi} : Y^* \to \tilde{Y}^*$. As mentioned above, every $x \in Y^*$ has a unique factorization $x = ha^e$ where $h \in H$ and $0 \leq e \leq p^2 - 1$. This shows that

$$\tilde{\varphi} : Y^* \to \tilde{Y}^*, \quad ha^e \mapsto h \pi \varphi \tilde{\pi}^{-1} \cdot \tilde{a}^e,$$
is well-defined; clearly, $\phi$ is a bijection, so it remains to show that it is a homomorphism. We use below the important property of $\phi$ that it maps $Y^*_{j}$ to $\hat{Y}^*_{j}$ for each $j$: this follows from the fact that the Hall subgroups $\hat{Q}_{1}, \ldots, \hat{Q}_{n}$ defining the Sylow basis $Y^*_{1}, \ldots, Y^*_{n}$ are mapped under $\phi$ to the Hall subgroups $\hat{Q}_{1}, \ldots, \hat{Q}_{i-1}, \hat{H}, \hat{Q}_{i+1}, \ldots, \hat{Q}_{n}$ defining the Sylow basis $\hat{Y}^*_{1}, \ldots, \hat{Y}^*_{n}$.

Let $x, y \in Y^*$ and write $x = ha^e$ and $y = ka^f$ with $h, k \in H$ and $e, f \in \{0, \ldots, p^2 - 1\}$. Write $(a^e)^k = ma^u$ with $m \in H$ and $u \in \{0, \ldots, p^2 - 1\}$, so that $xy = hk(a^e)^k a^f = (hk m)a^{u+f}$. This shows that

$$(xy)\phi = x\phi \cdot y\phi \iff (\tilde{a}^e)^{k\pi\varphi^{-1}} = m\pi\varphi^{-1} \cdot \tilde{a}^u,$$

and it remains to prove the following: for all $k \in H$ and $e \in \{0, \ldots, p^2 - 1\}$, if $(a^e)^k = ma^u$ with $m \in H$, then $(\tilde{a}^e)^{k\pi\varphi^{-1}} = m\pi\varphi^{-1} \cdot \tilde{a}^u$. Recall that every $k \in H$ can be written as a product of elements in the chosen Sylow tower of $Y^*$, say $k = h_1 \ldots h_l$ where $h_u$ and $h_v$ lie in different Sylow subgroups for $u \neq v$. We prove the claim by induction on $l$.

First, suppose $l = 1$, that is, $k$ lies in a Sylow $p_j$-subgroup $Y^*_j \leq H$ for some $j \neq i$. It follows from Lemma 6.2 that $Y^*_i Y^*_j = Y^*_i Y^*_j$ is a $\{p, p_j\}$-group, and there are two cases to consider.

(i) If $Y^*_j$ normalizes $Y^*_i$, then $\hat{Y}^*_j$ normalizes $\hat{Y}^*_i$. We can write $(a^e)^k = a^t$ for a uniquely determined $t \in \{0, \ldots, p^2 - 1\}$, which yields

$$\langle Aa \rangle^{k\pi} = \langle Aa \rangle^{i \mod p} \quad \text{and} \quad \langle \tilde{A}a \rangle^{k\pi\varphi} = \langle \tilde{A}a \rangle^{i \mod p}.$$

Since $k$ acts on $\langle Aa \rangle$ the same way as $k\pi\varphi$ acts on $\langle \tilde{A}a \rangle$, it follows from Lemma 6.2 that $k$ acts on $A$ the same way as $k\pi\varphi^{-1}$ acts on $\hat{A}$. Thus, if $(a^e)^k = a^t$, then $(\tilde{a}^e)^{k\pi\varphi^{-1}} = \tilde{a}^t$, as claimed.

(ii) If $Y^*_i$ normalizes $Y^*_j$, then $\hat{Y}^*_i$ normalizes $\hat{Y}^*_j$. Moreover, $A = \langle a^p \rangle \leq Y^*_j$ and $\hat{A} = \langle \tilde{a}^p \rangle \leq \hat{Y}^*_j$ act trivially on $Y^*_j$ and on $\hat{Y}^*_j$, respectively, and

$$(a^e)^k = [k, a^{-e}]a^e = [k, a^{-e \mod p}]a^e \quad \text{with} \quad [k, a^{-e \mod p}] \in Y^*_i \leq H$$

$$(\tilde{a}^e)^{k\pi\varphi^{-1}} = [k\pi\varphi^{-1}, \tilde{a}^{-e \mod p}]\tilde{a}^e \quad \text{with} \quad [k\pi\varphi^{-1}, \tilde{a}^{-e \mod p}] \in \hat{Y}^*_j \leq \hat{H}.$$

Thus, it remains to show that $[k, a^{-e \mod p}]\pi\varphi^{-1} = [k\pi\varphi^{-1}, \tilde{a}^{-e \mod p}]$. Note that

$$[k, a^{-e \mod p}]\pi\varphi^{-1} = [k\pi\varphi, \tilde{A}a^{-e \mod p}]\pi^{-1},$$

and $[k\pi\varphi^{-1}, \tilde{a}^{-e \mod p}] \in \hat{H}$ is a preimage of $[k\pi\varphi, \tilde{A}a^{-e \mod p}] \in \hat{H}\hat{A}/\hat{A}$ under the isomorphism $\tilde{\pi} : \hat{H} \to \hat{H}\hat{A}/\hat{A}$; recall that $\tilde{\pi}$ is the restriction of $\tilde{\Gamma} : \hat{Y}^* \to \hat{Y}$, and $\tilde{\Gamma}$ maps $k\pi\varphi^{-1}$ and $\tilde{a}$ to $k\pi\varphi$ and $\tilde{A}a$, respectively. Thus, $(\tilde{a}^e)^{k\pi\varphi^{-1}} = [k, a^{-e \mod p}]\pi\varphi^{-1} \cdot \tilde{a}^e$, as claimed.

Second, consider the induction step $l \geq 2$ and write $k = st$ such that the induction hypothesis holds for $s$ and $t$, that is, if $(a^e)^s = m_s a^u$ with $m_s \in H$, then $(\tilde{a}^e)^{s\pi\varphi^{-1}} = m_s\pi\varphi^{-1} \cdot \tilde{a}^u$, and that if $(a^u)^t = m_t a^u$ with $m_t \in H$, then $(\tilde{a}^u)^{t\pi\varphi^{-1}} = m_t\pi\varphi^{-1} \cdot \tilde{a}^u$. This yields $(a^e)^k = m_s m_t a^u$ with $m_s m_t \in H$, and therefore

$$(\tilde{a}^e)^{k\pi\varphi^{-1}} = (m_s\pi\varphi^{-1})^{\pi\varphi^{-1}} \cdot m_t\pi\varphi^{-1} \cdot \tilde{a}^u = (m_s m_t)\pi\varphi^{-1} \cdot \tilde{a}^u,$$

as claimed. This completes the proof that $\phi$ is an isomorphism between $Y^*$ and $\hat{Y}^*$. The construction of $\phi$ only employs a finite list of polynomial-time algorithms.

As explained in the beginning of this section, if the order of the cube-free group $L$ has $k$ distinct prime divisors, then the algorithm in Proposition 6.3 has to be iterated at most $k$ times to establish an isomorphism from $L$; note that $k \leq \log |L|$. This proves the following theorem.

**Theorem 6.4.** Let $L$ and $\hat{L}$ be two solvable cube-free groups. Algorithm 6 is a polynomial-time algorithm that constructs an isomorphism $L \to \hat{L}$, and reports false if and only if $L \not\cong \hat{L}$. 
7. Proof of Theorem 1.1 (Isomorphism testing of cube-free groups)

We now prove our main result, Theorem 1.1 by describing Algorithm 3. Recall from Section 3 that every cube-free group has the form \( G = A \times L \), with \( L \) solvable and \( A = \text{PSL}_2(p) \). If \( A \neq 1 \), then \( A = G^{(3)} \), the third term of the derived series of \( G \), see Remark 5.1. We compute \( G^{(3)} \) using the normal closure of commutators [28, p. 23]; since membership testing in permutation groups is in deterministic polynomial time, this can be done efficiently. Furthermore, as \( G^{(3)} \) is normal, the algorithm of [17, P6] applies to compute \( L = C_G(A) \) in polynomial time. Thus, we may decompose \( G = A \times L \), and likewise \( \tilde{G} \), in polynomial time. For Step (1) of the general algorithm, the construction of an isomorphism \( \psi : A \rightarrow \tilde{A} \), we use the next proposition. The correctness of Algorithm 3 now follows from Theorem 6.4 together with Proposition 7.1; the runtime is polynomial in the input size.

Proposition 7.1. Let \( A \) be isomorphic to a non-abelian simple group of cube-free order. There is a polynomial-time algorithm that returns an isomorphism \( A \rightarrow \text{PSL}_2(p) \).

Proof. By assumption, \( A \cong \text{PSL}_2(p) \). We can determine \( p \) by computing \( |A| \), and then find \( x, y \in A \) of order \( p \) and \( (p + 1)/2 \), respectively; note that \( \langle x, y \rangle \cong \text{PSL}_2(p) \) since \( x \) generates a Sylow \( p \)-subgroup, and \( y \) generates the image in \( \text{PSL}_2(p) \) of the \( (p - 1) \)-th power of a Singer cycle in \( \text{GL}_2(p) \). Now construct a presentation \( \langle x, y | R \rangle \) for \( A \) from these elements. In \( \text{PSL}_2(p) \), list all element pairs \( (x', y') \) of order \( p \) and \( (p + 1)/2 \), respectively, and search for an identification \( x \mapsto x' \) and \( y \mapsto y' \) that satisfies the relations \( R \). Once found, return the result as the isomorphism. If \( \text{PSL}_2(p) \) is represented on \( n \) points, then \( p \leq n \) and hence \( |\text{PSL}_2(p)| \leq n^3 \). The algorithm searches \( |\text{PSL}_2(p)|^2 \leq n^6 \) pairs, so this brute-force test ends in time polynomial in the input.

Proposition 7.1 is a shortcut, available because of our focus on a polynomial-time algorithm for permutation groups. Recognizing \( A \cong \text{PSL}_2(p) \) and constructing an isomorphism has been a subject of intense research; a polynomial time solution for groups of black-box type is discussed in [5].

8. Examples

We have implemented the critical features of our algorithm in [10], and we give a few demonstrations of its efficiency in Table 1. For each test, we constructed two (non-)isomorphic groups: we usually started with direct products of groups provided by GAP’s SmallGroup Library, and then created isomorphic random copies \( G \) and \( H \) of these groups (by using random polycyclic generating set). For some of the groups we have used, Table 1 gives their size and code; this data can be used to reconstruct the groups via the GAP function \( \text{PcGroupCode} \). We applied our function \text{IsomorphismCubefreeGroups} to find an isomorphism \( G \rightarrow H \). When comparing the efficiency of our implementation with the GAP function \text{IsomorphismGroups}, we have started both calculations with freshly constructed groups \( G \) and \( H \), to make sure that previously computed data is not stored. We note that GAP also provides a randomized function (\text{RandomIsomorphismTest}) that attempts to decide isomorphism between finite solvable groups (given via their size and code); the current implementation does not return isomorphisms. That algorithm runs exceedingly fast on many examples, see Table 1, but its randomized approach means it cannot be guaranteed to detect all isomorphisms. There are some practical bottlenecks in our implementation which currently applies available libraries for pc-groups (cf. Section 6.1) and matrix groups (cf. Section 5). The efficiency problems for \text{collection} (cf. [21]) become visible when larger primes are involved. (This is one reason why it takes several minutes to reconstruct some of the groups in Table 1 via \text{PcGroupCode}.) Moreover, GAP’s functionality for matrix groups is not yet making full use of the promising advances of the \text{matrix group recognition project}. These bottlenecks are responsible for the long runtime of the examples involving the prime 12198421, which is large from the perspective of GAP. Nevertheless, as a proof of concept, these examples demonstrate well the efficiency of our algorithm compared to existing methods.
**Algorithm 1 CyclicLift**

**Input:** cube-free solvable groups $Y^*, \tilde{Y}^*$ with $|Y^*| = |\tilde{Y}^*|$, subgroups $A \subseteq \Phi(Y^*)$ and $\tilde{A} \subseteq \Phi(\tilde{Y}^*)$ isomorphic to $C_p$, natural projections $\Gamma: Y^* \to Y^*/A$ and $\tilde{\Gamma}: \tilde{Y}^* \to \tilde{Y}^*/\tilde{A}$ with images $Y = Y^*\Gamma$ and $\tilde{Y} = \tilde{Y}^*\tilde{\Gamma}$, and an isomorphism $\varphi: Y \to \tilde{Y}$

**Output:** an isomorphism $\hat{\varphi}: Y^* \to \tilde{Y}^*$

```python
def CyclicLift($Y^*$, $A$, $\Gamma$, $\tilde{Y}^*$, $\tilde{A}$, $\tilde{\Gamma}$, $\varphi$)
    use Proposition 4.6 to get a Sylow basis $Y_1^*, \ldots, Y_n^*$ of $Y^*$
    define $H = \prod_{i \neq j} Y_i^*$, where $A \subseteq Y_i^*$; this is a Hall $p'$-subgroup of $Y^*$
    construct $\hat{H}$ as a Hall $p'$-subgroup in the preimage of $H\Gamma\varphi$ under $\tilde{\Gamma}$
    construct induced isomorphisms $\pi: H \to H\tilde{\Gamma}$
    fix a generator $a$ of $Y_i^*$ (Lemma 6.1) and let $\tilde{a} \in \tilde{Y}^*$ be a preimage of $a\Gamma\varphi$ under $\tilde{\Gamma}$
    let $M$ be a generating set of $H$
    define $\hat{\varphi}: Y^* \to \tilde{Y}^*$ by mapping each $m \in M$ to $m\pi(\tilde{a})^{-1}$, and $a$ to $\tilde{a}$
    return $\hat{\varphi}$
```

**Algorithm 2 Lift**

**Input:** cube-free solvable groups $L$ and $\tilde{L}$ of the same order

**Output:** an isomorphism $\hat{\varphi}: L \to \tilde{L}$, or false if $L \not\cong \tilde{L}$

```python
def Lift($L$, $\tilde{L}$)
    compute $\Phi(L)$ and $\Phi(\tilde{L})$, see Section 6.1
    if $|\Phi(L)| = |\Phi(\tilde{L})| = 1$ then
        use the algorithm of Section 6.2 to get an isomorphism $\hat{\varphi}: L \to \tilde{L}$, or return false if that fails
    else
        decompose $\Phi(L) = \prod_{i=1}^n Y_i$ and $\Phi(\tilde{L}) = \prod_{i=1}^n \tilde{Y}_i$ into Sylow subgroups
        for each $i$ define $M_i = Y_i / Y_i$ and $\tilde{M}_i = \tilde{Y}_i / \tilde{Y}_i$, with $M_{\geq 2} = 1 = \tilde{M}_{\geq 2}$
        for each $i \geq 2$ define natural projections $\pi_i: L_i \to L_{i-1}$ and $\tilde{\pi}_i: \tilde{L}_i \to \tilde{L}_{i-1}$
        use the algorithm of Section 6.3 to get an isomorphism $\hat{\varphi}: L_1 \to \tilde{L}_1$, or return false if that fails
        for $i = 2, \ldots, n + 1$
            set $\hat{\varphi} = \text{CyclicLift}(L_i, M_i, \pi_i, \tilde{L}_i, \tilde{M}_i, \tilde{\pi}_i, \hat{\varphi})$, which is an isomorphism $L_i \to \tilde{L}_i$
        return $\hat{\varphi}$
```

**Algorithm 3 IsomorphismCubeFreeGroups**

**Input:** Cube-free groups $G$ and $\tilde{G}$ of the same order

**Output:** an isomorphism $\varphi: G \to \tilde{G}$, or false if $G \not\cong \tilde{G}$

```python
def IsomorphismCubeFreeGroups($G$, $\tilde{G}$)
    compute $A = G(3)$ and $L = C_G(A)$, as well as $\hat{A} = \tilde{G}(3)$ and $\hat{L} = C_{\tilde{G}}(\hat{A})$
    construct an isomorphism $\psi_A: A \to \hat{A}$, or return false if $A \not\cong \hat{A}$, see Proposition 7.1
    construct $\psi_L = \text{Lift}(L, \hat{L})$, which is an isomorphism $\psi_L: L \to \hat{L}$, or return false if $L \not\cong \hat{L}$
    combine $\psi_A$ and $\psi_L$ to an isomorphism $\varphi: G \to \tilde{G}$
    return $\varphi$.
```
size: \(213444 = 2^2 \cdot 3^2 \cdot 7^2 \cdot 11^2\) (two isomorphic groups)
Runtime IsomorphismCubefreeGroups: 0.12 seconds; GAP runtimes: 110 seconds and 0.30 seconds

size: \(485100 = 2^2 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11\) (two isomorphic groups)
Runtime IsomorphismCubefreeGroups: 0.14 seconds; GAP runtimes: 9.25 hours and 0.10 seconds

size: \(2455229080695145234788 = 2^2 \cdot 3^2 \cdot 7 \cdot 11 \cdot 17 \cdot 23 \cdot 29 \cdot 59 \cdot 709 \cdot 2837 \cdot 22697\) (two isomorphic groups)
Runtime IsomorphismCubefreeGroups: 46 seconds; GAP runtimes: (aborted) and 12.85 hours

code: 2577188729005826832444422524842761846662253556118841815720622231553081763698516063983276468223398558454926208711434863233254329561285
7310614599377329545076424741385533019060045922880910282042489387835906289279581907750184052068613887290089849139978833781413618189

code: 42935964225237064245986914596365100273683956747676598979814176980191348433158368824756791059830426394631361311711333822038779784490919
1398533193638418225692067093120389360092220226273076405684036236511208423558471856377830123474389120161517062590458151937327292273539

size: \(148801462694820 = 2^2 \cdot 3^2 \cdot 5^2 \cdot 13^2 \cdot 101^2 \cdot 12198421\) (two isomorphic groups)
Runtime IsomorphismCubefreeGroups: 1.34 hours; GAP runtimes: (aborted) and 43.02 hours

code: 334854701398962523593284308049022665678988488902162933506287743035973030608696148032177992911339898019268212938339562678223839825646765
361434767015687884441409067142348506676356984698439325927137381308225235803152167560684518515666603208366321490271081072566186700588774
041361401470419

code: 30847018874524841211989997769621350148838465133706306053219004256466469753555564996929067797274360859515323683234426380109067389439911324
745735385820870698510892979250514584900163986344080431816565839

size: \(11793441660 = 2^2 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11^2 \cdot 13 \cdot 17^2 \) (two non-isomorphic groups)
Runtime IsomorphismCubefreeGroups: 1.00 seconds; GAP runtimes: (aborted) and (not applicable)

code: 1307598631642127859218289820459634912906711569345827873045599909594056157779256403437080197441699

code: 1408183141768445386850846022592180843601522695751988374145441188175177279691404248957075447896750

**Table 1.** Comparison of runtimes of isomorphism tests for some cube-free groups; GAP runtimes are given for the GAP functions \texttt{IsomorphismGroups} and \texttt{RandomIsomorphismTest} (in that order); we aborted computations which used more that 20GB of memory
Isomorphism testing of groups of cube-free order

References

[1] L. Babai, P. Codenotti, J. A. Grochow, Y. Qiao. Code equivalence and group isomorphism. Proc. of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, 1395–1408, SIAM, Philadelphia, 2011.

[2] L. Babai, E. Szemerédi. On the complexity of matrix group problems I. In Proc. 25th IEEE Symp. Foundations Comp. Sci., 229–240, 1984.

[3] L. Babai, Y. Qiao. Polynomial-time isomorphism test for groups with abelian Sylow towers. 29th International Symposium on Theoretical Aspects of Computer Science, 453–464, LIPIcs. Leibniz Int. Proc. Inform., 14, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2012.

[4] S. R. Blackburn, P. M. Neumann, G. Venkataraman. Enumeration of finite groups. Cambridge Press 2007.

[5] L. Babai, E. Szemerédi. On the complexity of matrix group problems I. In Proc. 25th IEEE Symposium on Foundations of Computer Science, 229–240, 1984.

[6] L. Babai, Y. Qiao. Polynomial-time isomorphism test for groups with abelian Sylow towers. 29th International Symposium on Theoretical Aspects of Computer Science, 453–464, LIPIcs. Leibniz Int. Proc. Inform., 14, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2012.

[7] L. Babai, Y. Qiao. Polynomial-time isomorphism test for groups whose Lie algebra has genus 2. J. Algebra 473 (2017), 545–590.

[8] L. Babai, Y. Qiao. Polynomial-time isomorphism test for groups with abelian Sylow towers. 29th International Symposium on Theoretical Aspects of Computer Science, 453–464, LIPIcs. Leibniz Int. Proc. Inform., 14, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2012.

[9] H. Dietrich, B. Eick. On the Groups of Cube-Free Order. J. Algebra 292 (2005) 122–137, with addendum in J. Algebra 367 (2012) 247–248.

[10] H. Dietrich. The GAP4 package Cubefree. Available at [users.monash.edu/~heikod/cubefree.html](http://users.monash.edu/~heikod/cubefree.html).

[11] H. Dietrich. The GAP4 package Cubefree. Available at [users.monash.edu/~heikod/cubefree.html](http://users.monash.edu/~heikod/cubefree.html).

[12] R. L. Duncan. On the density of the $k$-free integers. Fibonacci Quart. 7 (1969) 140–142.

[13] D. L. Flannery, E. A. O'Brien. The linear groups of small degree over finite fields. Intern. J. Alg. and Comput. 15 (2005) 467–502.

[14] GAP – Groups, Algorithms and Programming. Available at [gap-system.org](http://gap-system.org).

[15] B. Höfling. Computing projectors, injectors, residuals and radicals of finite soluble groups. J. Symb. Comp. 32 (2001) 499–511.

[16] D. F. Holt, B. Eick, E. A. O'Brien. Handbook of computational group theory. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2005.

[17] W. M. Kantor, E. M. Luks. Computing in quotient groups. Proceedings 22nd ACM Symposium on Theory of Computing (1990) 524–534.

[18] W. M. Kantor, E. M. Luks, P. D. Mark. Sylow subgroups in parallel. J. Algorithms 31 (1999) 132–195.

[19] G. Karagiorgos, D. Poulakis. An algorithm for computing a basis of a finite abelian group. Algebraic informatics, 174–184, Lecture Notes in Comput. Sci., 6742, Springer, Heidelberg, 2011.

[20] E. M. Luks, T. Miyazaki. Polynomial-time normalizers. Discrete Math. Theor. Comput. Sci. 13 (2011) 61–96.

[21] M. F. Newman, A. Niemeyer. On complexity of multiplication in finite soluble groups. J. Algebra 421 (2015) 425–430.

[22] E. M. Luks. Computing the composition factors of a permutation group in polynomial time. Combinatorica 7 (1987) 87–99.

[23] E. M. Luks. Computing in Solvable Matrix Groups. In Proceedings 33rd Annual Symposium on Foundations of Computer Science (1992) 111–120.

[24] J-P. Massias, G. Robin. Bories effectives pour certaines fonctions concernant les nombres premiers. J. Théor. Nombres Bordeaux 8 (1996) 215–242.

[25] S. Qiao, C. H. Li. The finite groups of cube-free order. J. Algebra 334 (2011) 101–108.

[26] D. J. S. Robinson. A Course in the Theory of Groups. Springer-Verlag, 1982.

[27] J. B. Rosser, L. Schoenfeld. Approximate formulas for some functions of prime numbers. Illinois J. Math. 6 (1962) 64–94.

[28] Á. Seress. Permutation group algorithms. Cambridge University Press 152, Cambridge, 2003.

[29] D. Taunt. Remarks on the isomorphism problem in theories of construction of finite groups. Proc. Cambridge Philos. Soc. 51 (1955) 16–24.

[30] J. B. Wilson. Finding direct product decompositions in polynomial time. [arXiv:1005.0548](http://arxiv.org/abs/1005.0548).

[31] J. B. Wilson. The threshold for subgroup profiles to agree is log $n$ − 2. [arXiv:1612.01444](http://arxiv.org/abs/1612.01444).

School of Mathematics, Monash University, Clayton VIC 3800, Australia

E-mail address: Heiko.Dietrich@Monash.Edu

Department of Mathematics, Colorado State University, Fort Collins Colorado, 80523, USA

E-mail address: James.Wilson@ColoState.Edu