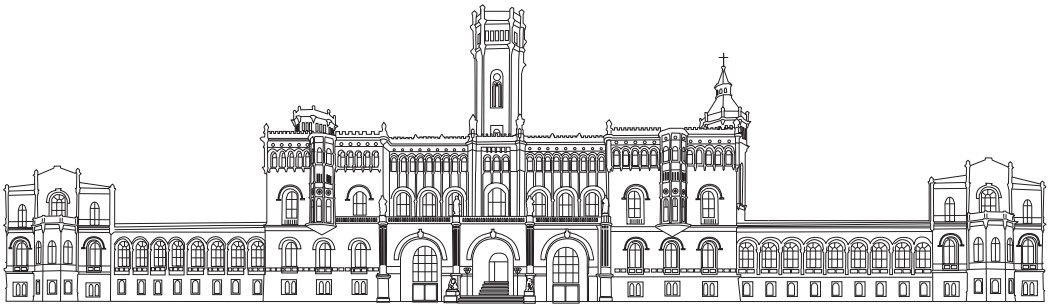

Leibniz Universität Hannover

Institut für theoretische Informatik

Masterarbeit

Group Isomorphism

Daniel Wiebking

Matrikel-Nr.: 2941990

8. November 2016

Erstprüfer: Prof. Dr. Heribert Vollmer
Zweitprüfer: Dr. Arne Meier
Betreuer: M. Sc. Maurice Chandoo

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst habe und keine anderen Hilfsmittel und Quellen als angegeben verwendet habe.

_____

Daniel Wiebking

Contents

1 Introduction

The group isomorphism problem is to decide whether two finite groups, given as multiplication tables, are isomorphic. This problem is clearly in NP since such an isomorphism would have polynomial length. The main reason to analyse group isomorphism is that the problem is neither known to be in P, nor to be NP-complete. However, it is known that the group isomorphism problem is Karp reducible to the isomorphism problem of graphs. This reduction can be done by representing the group elements as vertices and describing the ternary group relation "$g_1 \bullet g_2 = g_3$" with a suitable graph attached to the vertices. On the other side, graph isomorphism may be harder since it is not AC^0-reducible to the isomorphism problem of groups [CTW13]. So, the group isomorphism problem may be strictly easier than the isomorphism problem of graphs, which itself is not NP-hard, unless the polynomial hierarchy collapses to the second level [GMW91]. Additionally, we can bound the time complexity of group isomorphism by a quasi-polynomial term of $n^{\log_2(n)+\mathcal{O}(1)}$, where n is the order of the given groups. This was done by the *generator enumeration algorithm*, which was published first in [FN14] in 1967. Their idea utilizes the fact that every group of order n has a generating set whose size is $\log_2(n)$ at most, and that this generating set can be found in polynomial time iteratively. Moreover, an isomorphism is uniquely determined by assigning the function image of this generating set. This leads to an upper bound of $n^{\log_2(n)}$ on the number of possible isomorphisms, of which we can check each in polynomial time. The quasi-polynomial bound is an additional hint that the group isomorphism problem may not be NP-hard, since otherwise all NP-complete problems would be solvable in sub-exponential time, which contradicts the exponential time hypothesis in [IP99]. For a long time it was not known whether the bound of $n^{\log_2(n)+\mathcal{O}(1)}$ could be tightened until David J. Rosenbaum invented the *bidirectional collision detection* in 2013 to obtain a square-root speedup for the generator enumeration algorithm [Ros13]. This new bound of $n^{\frac{1}{2}\log_2(n)+\mathcal{O}(1)}$ is currently the best for general groups. In addition, bidirectional collision detection was also applicable to many other isomorphism decision algorithms.

One example was Rosenbaum's previously published algorithm for solvable groups [Ros14] whose runtime could be improved to $n^{\frac{1}{4}\log_2(n)+o(\log_2(n))}$. Rosenbaum's key for solvable groups was a reduction to the isomorphism problem of *composition series* and a subsequent reduction to the isomorphism problem of bounded degree graphs. The bounded degree graph isomorphism problem in turn is polynomial-time reducible to the group theoretic computational problem *set-stabilizer* as shown by Luks M. Eugene [Luk82]. To solve the *set-stabilizer* problem for certain instances, Eugene developed a divide-and-conquer mechanism that runs in polynomial time [Luk82].

To extend Rosenbaum's approach to general groups, Eugene developed a more direct way to reduce group isomorphism to the set-stabilizer problem in 2015 [Luk15]. He accomplished this by skipping the reduction to bounded degree graphs. As a result, Eugene achieves a runtime of $n^{\frac{1}{2}\log_2(n)+\mathcal{O}(1)}$ which does not outperform the improved generator enumeration algorithm. It is worth noting that Eugene does not currently make use of Rosenbaum's bidirectional collision detection. He has, however, claimed to apply it in a follow-up paper, which would result in a new time bound for the isomorphism problem of general groups. In this thesis we do some preparatory work for applying bidirectional collision detection to Eugene's framework. Moreover, we give detailed proofs of his results.

2 Preliminaries

If the reader is already familiar with group theory, then this chapter may be skipped.

Definition 2.1 (Group)**.** *A group is a set G together with a binary operation $\bullet : G \times G \to G$ which satisfies the following conditions, known as the group axioms. (We write $g_1 g_2$ instead of $\bullet(g_1, g_2)$.)*

1. *(Associativity) $(g_1 g_2)g_3 = g_1(g_2 g_3)$ for all $g_1, g_2, g_3 \in G$.*
2. *(Identity element) There is an element $e \in G$ such that $eg = ge = g$ for all $g \in G$.*
3. *(Inverse element) For all elements $g \in G$ there is an element in G, denoted by g^{-1}, such that $gg^{-1} = g^{-1}g$ is the identity element.*

Definition 2.2 (Order)**.** *The order of a group G is the cardinality of its underlying set $|G|$.*

In this thesis we shall always use the letter n to denote the cardinality of our considered groups. Next, we consider cyclic groups, which are classic examples of groups.

Definition 2.3 (Cyclic group)**.** *A finite group G is cyclic if it is generated by one single element, in symbols $G = \langle g \rangle = \{g, g^2, \ldots, g^n\}$.*

Example 2.4 The group $\mathbb{Z}_n := \{1, 2, \ldots, n\}$ together with modular addition $a \bullet b := ((a + b) \mod n)$ is cyclic. In fact, every cyclic group of order n is isomorphic to $\mathbb{Z}_n$.

Finite groups can be denoted as a table.

Definition 2.5 (Cayley table)**.** *A Cayley table of a group G with operation $\bullet : G \times G \to G$ is the representation of $\bullet$ by a table.*

Example 2.6 This is a possible Cayley table of $\mathbb{Z}_4$.

$\mathbb{Z}_4$	0	3	2	1
0	0	3	2	1
3	3	2	1	0
2	2	1	0	3
1	1	0	3	2

Note that the rows and columns of each Cayley table may be rearranged. This results in a new Cayley visualization of the same group. Hence, we refer to a Cayley table as "a", and not "the" Cayley table.

Definition 2.7 (Symmetric group)**.** *The symmetric group on a finite set $\mathcal{X}$, in symbols Sym($\mathcal{X}$), is the set of all permutations on $\mathcal{X}$ together with composition as the group operation. The symmetric group Sym($\{1, 2, \ldots, n\}$) is also denoted by Sym(n).*

Definition 2.8 (Subgroup)**.** *A subset $S \subseteq G$ is a subgroup of G, denoted by $S \leq G$, if S together with $\bullet|_S$ is a group as well. This implies that S is closed under its group operation, in symbols $\bullet_S : S \times S \to S$.*

Every group possesses at least one subgroup, as seen in the next definition.

Definition 2.9 (Trivial subgroup)**.** *The trivial subgroup of a group G is the set $\{e\} =: 1 \leq G$.*

Definition 2.10 (Coset)**.** *The left coset of a subgroup $S \leq G$ in G with respect to $g \in G$ is the set $gS = \{gh \mid h \in S\}$. Analogously, the right coset is defined as Sg.*

In this thesis we do not distinguish between right and left cosets since the subgroups, for which we consider cosets, will always meet certain criteria which we shall define next.

Definition 2.11 (Normal subgroup)**.** *A subgroup N of a group G is normal, symbolically $N \trianglelefteq G$, if $gN = Ng$ for all $g \in G$. In such a case, the left and right cosets of N in G coincide and we simply refer them as cosets.*

It is fairly obvious that $\{e\}$ and the group itself are normal in every group. Moreover, there are certain groups in which these are the only normal subgroups.

Definition 2.12 (Simple group)**.** *A group $G \neq 1$ is simple if the only normal subgroups it has are 1 and G.*

Definition 2.13 (Centralizer)**.** *The centralizer of a subset $S \subseteq G$ is*

$$C_G(S) := \{g \in G \mid gs = sg \ \forall s \in S\}.$$

Definition 2.14 (Center)**.** *The center of a group G is $C(G) := C_G(G)$.*

It is easy to show that the centralizer is a subgroup. Moreover, the center is a normal one. To understand the structure of a given group it is useful to consider its normal subgroups. Furthermore, the relationship of these subgroups to the group itself may provide us with more information. Next, we define how this relationship can be expressed.

Definition 2.15 (Factor group)**.** *Let $N \trianglelefteq G$. The factor group of G by N, in symbols G/N, is the set of all cosets of N in G and is equal to $\{gN \mid g \in G\}$. The induced operation of G defines an operation on the cosets in N:*

$$(g_1N)(g_2N) = g_1(Ng_2)N = g_1(g_2N)N = (g_1g_2)N.$$

The concept of factor groups can be extended to chains of normal subgroups.

Definition 2.16 (Subnormal series)**.** *A subnormal series of a group G is a subgroup chain where each subgroup is normal in the next one. It is denoted by $1 = G_0 \triangleleft G_1 \triangleleft \ldots \triangleleft G_m = G$.*

Definition 2.17 (Composition series)**.** *A composition series cs_G of a group G is a subnormal series $1 = G_0 \triangleleft G_1 \triangleleft \ldots \triangleleft G_m = G$ of maximal length. Or equivalently, a subnormal series, in which each composition factor G_{i+1}/G_i is simple.*

The finite cyclic groups are very well understood. Next, we consider groups that can be constructed using them.

Definition 2.18 (Solvable group)**.** *A finite group G is solvable if there is a composition series $1 = G_0 \triangleleft \ldots \triangleleft G_m = G$ of G, in which every composition factor G_{i+1}/G_i is cyclic of prime order.*

Definition 2.19 (Solvable radical)**.** *The solvable radical of a finite group G, denoted by $Rad(G)$, is the union of all solvable normal subgroups of G and therefore the inclusion maximal solvable normal subgroup.*

An established way to compare groups uses structure-preserving maps.

Definition 2.20 (Group homomorphism)**.** *A group homomorphism from G to H is a function $\varphi : G \to H$ such that*

$$\varphi(g_1g_2) = \varphi(g_1)\varphi(g_2) \quad \forall g_1, g_2 \in G.$$

Definition 2.21 (Group isomorphism)**.** *Two groups G and H are isomorphic, in symbols $G \cong H$, if there is a bijective group homomorphism φ from G to H. In this case φ is called an (group) isomorphism.*

Definition 2.22 (Automorphism group)**.** *The automorphism group of a group G, symbolically written as $Aut(G)$, is the set of isomorphisms from G to itself together with composition as the group operation.*

By definition we have $\mathrm{Aut}(G) \leq \mathrm{Sym}(G)$.

Theorem 2.23 (First isomorphism theorem [Rot12])**.** *Let $\varphi : G \to H$ be a group homomorphism. Then $G/\ker\varphi \cong \operatorname{im}\varphi$.*

We define how to join groups in an intuitive way.

Definition 2.24 (Direct product)**.** *The direct product of groups G and H is the Cartesian product $G \times H$ together with the element-wise operation:*

$$(g_1, h_1) \bullet (g_2, h_2) := (g_1 g_2, h_1 h_2).$$

3 Isomorphism of Groups

In this chapter we will consider the following problem.

Problem	GroupIso
Input	Groups G and H given by their Cayley tables.
Question	Is there an isomorphism between G and H?

In order to decide this problem, we will perform multiple Turing reductions until we reach an elementarily solvable problem.

3.1 Reduction from GroupIso to CSeriesIso

In this section we will reduce the given problem to the isomorphism problem of composition series.

Definition 3.1 (Subnormal series isomorphism)**.** *Two subnormal series s_G and s_H are isomorphic, in symbols $s_G \cong s_H$, if there is an isomorphism φ from G to H such that*

$$\varphi(G_i) = H_i \quad \forall\ 0 \leq i \leq m.$$

We define a corresponding isomorphism problem for subnormal series of maximal length.

Problem	CSeriesIso
Input	Composition series of groups G and H.
Question	Is there an isomorphism between the composition series?

For the reduction to this problem we may proceed as follows. At first, we enumerate all composition series for the groups G and H respectively. Then, we check whether these enumerations contain an isomorphic pair. If we find such a pair, then the groups must be patently isomorphic. On the flipside, isomorphic groups have isomorphic sets of composition series. Here, two sets are called isomorphic if there is a bijection between them which maps an element to an isomorphic one. A problem with this approach is that the number of composition series is not bounded by a polynomial. For this reason,

one could attempt to enumerate suitable subsets of composition series. These subsets should have the following property.

Definition 3.2 (Collision sets)**.** *Two sets $\mathcal{A}$ and $\mathcal{B}$ of composition series of G and H respectively are called collision sets if*

$$G \cong H \quad \text{if and only if} \quad \exists cs_G \in \mathcal{A}\ \exists cs_H \in \mathcal{B} : cs_G \cong cs_H.$$

Next, we will define a concept which will prove to be very useful when considering the isomorphism of sets.

Definition 3.3 (Canonical form)**.** *Let $\sim$ be an equivalence relation on a set $\mathcal{X}$. A canonical form for the set $\mathcal{X}$ with $\sim$ is a function $\mathbf{Can} : \mathcal{X} \to \mathcal{X}$ such that $x \sim \mathbf{Can}(x)$ and*

$$x \sim y \quad \text{if and only if} \quad \mathbf{Can}(x) = \mathbf{Can}(y) \quad \forall x, y \in \mathcal{X}.$$

With the help of a canonical form we can decide the isomorphism of sets more effectively, as is illustrated in the next example.

Example 3.4 Group isomorphism is an equivalence relation on the set of all groups represented by their Cayley tables and two groups are equal if their tables coincide. Then, **Can** is a function on the Cayley tables of groups which preserves the isomorphism class, in symbols $G \cong \mathbf{Can}(G)$, and which is invariant under isomorphisms, in symbols

$$G \cong H \quad \text{if and only if} \quad \mathbf{Can}(G) = \mathbf{Can}(H).$$

If we find a canonical form for groups, we can decide the isomorphism of two groups G and H by evaluating **Can** for them and by comparing the results in one step afterwards. So, evaluating a canonical form on groups is at least as hard as deciding isomorphism. In fact, the canonical form computation is more useful if we want to decide isomorphism between two sets of groups $\{G_1, \ldots, G_n\}$ and $\{H_1, \ldots, H_n\}$. Only $2n$ evaluations of the canonical form, instead of n^2 calls of a group isomorphism decision algorithm are needed. The same premise is advantageous in checking if two collision sets share an isomorphic element.

Next, we shall define a canonical version of the isomorphism problem of composition series. Since there can be different canonical forms for composition series, we shall now stipulate one form denoted by **Can**.

Problem	**Can**(CSeries)
Input	Composition series of a group G.
Output	The function **Can** evaluated at the composition series.

The first approach to get smaller collision sets involves enumeration of composition series with a certain property, instead of considering the entire set of composition series. This strategy works provided the property is invariant under isomorphisms such as the following one.

Definition 3.5 (Tree property)**.** *A composition series* $1 = G_0 \lhd \ldots \lhd G_m$ *has the tree property if one of the following holds.*

1. $m = 1$*, so the composition series is equal to* $1 \lhd G_1$*.*
2. *There is one* $i \in \{1, \ldots, m\}$ *such that both* $1 = G_0 \lhd G_1 \lhd \ldots \lhd G_i$ *and* $1 = G_i/G_i \lhd G_{i+1}/G_i \lhd \ldots \lhd G_m/G_i$ *have the tree property, where* G_i *is the smallest subgroup which fulfills* $G_0 \lhd G_i \lhd G_m$*.*

In order to enumerate the composition series with the tree property, it is natural to ask how many such series exist. This question was answered in [RW15] with the help of László Babai. They showed that the number of all composition series for a group of order n is confined by $n^{\frac{1}{2}\log_2(n)+\mathcal{O}(1)}$ and enumerated them within the same time bound. However, we are interested in getting even smaller collision sets. Thus, we abstain from enumerating all composition series with the tree property. To achieve this, we need the next definition.

Definition 3.6 (Normal closure)**.** *The normal closure of a subset* $\mathcal{S} \subseteq G$*, symbolically represented as* $\langle \mathcal{S} \rangle^G$*, is the intersection of all normal subgroups of* G *that contain* $\mathcal{S}$ *and is therefore the inclusion minimal normal subgroup of* G *that contains* $\mathcal{S}$*.*

Lemma 3.7. *The normal closure is polynomial-time computable.*

Proof. This can be shown by iteratively computing the closure under the group operation and the closure under conjugation in G. □

Lemma 3.8. *Let* $n = |G|$ *and* $0 \leq a \leq b \leq \log_2(n)$*. Algorithm 3.1 computes composition series* $1 = G_0 \lhd \ldots \lhd G_m = G$ *in time* $2^{\frac{1}{2}(b-a)(2\log_2(n)-b-a)}n^{\mathcal{O}(1)}$*.*

Proof. The idea behind this proof is quite simple. In order to enumerate the composition series with the tree property, the algorithm recursively computes the result using the property's definition. Whether or not the algorithm traces all series, depends on the values of a and b. The algorithm branches maximally if the current recursion depth lies between a and b. Otherwise, it branches minimally. In each recursive call the recursion depth is incremented by one.

Algorithm 3.1 Enumerate composition series with the tree property.

function EnumCSeries(G_0, G_m, a, b)
  for all $gG_0 \in G_m/G_0$ **do**
    $G_i := \langle g, G_0 \rangle^{G_m}$
    if $G_0 \lhd G_i \lhd G_m$ **then**
      $a_0 := \max(\min(\log_2(|G_i/G_0|), a-1), 0)$
      $b_0 := \max(\min(\log_2(|G_i/G_0|), b-1), 0)$
      $S_0 := \text{EnumCSeries}(G_0, G_i, a_0, b_0)$
      $a_1 := \max(\min(\log_2(|G_m/G_i|), a-1), 0)$
      $b_1 := \max(\min(\log_2(|G_m/G_i|), b-1), 0)$
      $S_1 := \text{EnumCSeries}(G_i, G_m, a_1, b_1)$
      $S_0 \circ S_1 := \left\{ G_0 \lhd \ldots \lhd G_i \lhd \ldots \lhd G_m \;\middle|\; \begin{array}{l} G_0 \lhd \ldots \lhd G_i \in S_0 \\ G_i \lhd \ldots \lhd G_m \in S_1 \end{array} \right\}$
      $S := S \cup (S_0 \circ S_1)$
      if $a = 0 \Leftrightarrow b = 0$ **then**
        return S
      end if
    end if
  end for
  if $S \neq \emptyset$ **then**
    return S
  else
    return $S := \{G_0 \lhd G_m\}$
  end if
end function

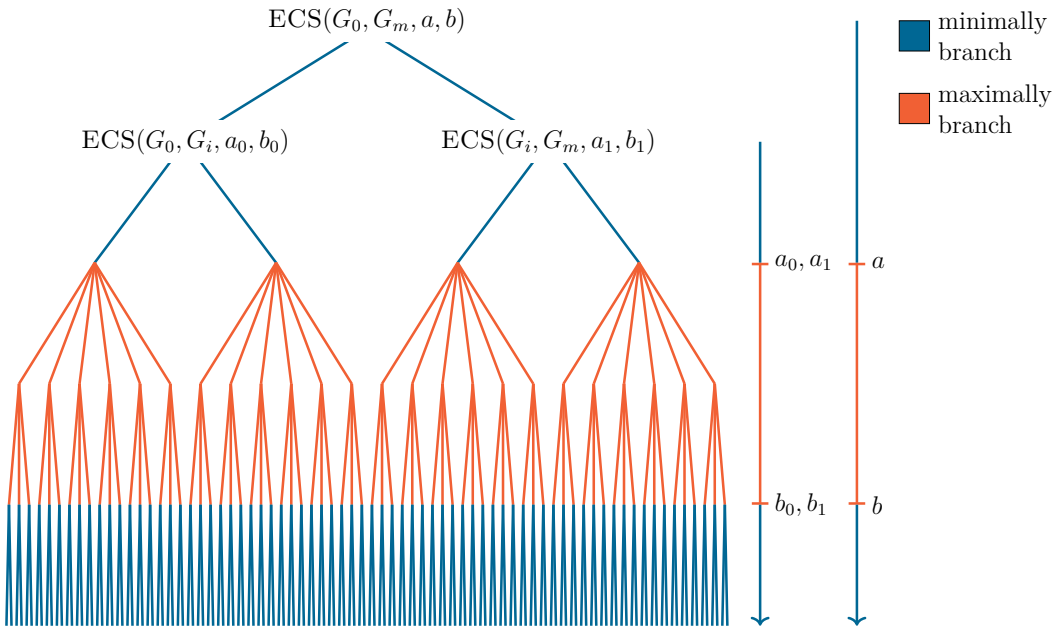

Figure 3.1: Recursion tree of Algorithm 3.1.

Therefore, the algorithm reduces the values of a and b in line 5 - 9 by one as well. To keep the values in the range $0 \leq a \leq b \leq \log_2(n)$, we need to consider their maximum and minimum respectively. For a better understanding, the reader is encouraged to see the example of the algorithm's recursion tree in Figure 3.1 as well.

Correctness. By induction on the recursion depth, we show that Algorithm 3.1 with input $G_0 \lhd G_m$ enumerates subnormal series $G_0 \lhd G_1 \lhd \ldots \lhd G_m$ of maximal length. For the base case, we note that the algorithm does not have a recursive call and therefore returns $G_0 \lhd G_m$. We shall provide a proof using contradiction for the above-mentioned statement. Let us assume that there is a longer subnormal series $G_0 \lhd \ldots \lhd G_{m-1} \lhd G_m$. Now, observe that for $g \in G_{m-1} \setminus G_0$ the group $G_i = \langle g, G_0 \rangle^{G_m}$ is by definition the inclusion minimal normal subgroup of G_m that contains $\{g\} \cup G_0 \subseteq G_{m-1}$. This implies $G_i \trianglelefteq G_m$ and $G_i \leq G_{m-1} \lhd G_m$, which leads to a strict normal subgroup, in symbols $G_i \lhd G_m$. Next, we consider the normal subgroup relation $G_0 \trianglelefteq G_i$ which holds since $G_0 \trianglelefteq G_m$. This is, again, a strict inclusion $G_0 \lhd G_i$ for $g \notin G_0$. In summary, we get $G_0 \lhd G_i \lhd G_m$ and therefore the algorithm reaches line 5, which contradicts the assertion that the algorithm does not have a recursive call$\lightning$.

Thus $G_0 \lhd G_m$ is of maximal length and the base case is proven. For the inductive step we assume that the algorithm reaches line 11. By induction S_0 and S_1 are sets of subnormal series of maximal length and so is $S_0 \circ S_1$. This holds since a subnormal series is of maximal length if and only if each composition factor is simple.

Complexity. Let $n_0 = |G_0|$, $n_m = |G_m|$ and let $T(n_0, n_m)$ denote the total runtime of line 11 and 21 over each recursive call. Let $R(n_0, n_m)$ denote the total number of recursive calls, which is confined by $T(n_0, n_m)$ since each recursive call reaches line 11 or 21. Moreover, let $T_{\text{single}}(n_0, n_m)$ denote the runtime of all lines except line 11 and 21 in one single recursive call, which is bounded by a polynomial p. This bound holds since the algorithm leaves the current recursive call in line 7 and 10 and therefore the runtime of line 7 and 10 has no influence on $T_{\text{single}}(n_0, n_m)$. This leads to a total runtime of

$$\begin{aligned} \underbrace{T(n_0, n_m)}_{\text{line 11 and 21}} + \underbrace{R(n_0, n_m) T_{\text{single}}(n_0, n_m)}_{\text{other lines}} &\leq T(n_0, n_m) + T(n_0, n_m) p(n_0 + n_m) \\ &= T(n_0, n_m)(1 + p(n_0 + n_m)). \end{aligned}$$

Now, it remains to be shown that $T(n_0, n_m)$ is bounded by the desired term in Lemma 3.8. To analyze $T(n_0, n_m)$, we observe that it satisfies the following recursion.

$$\begin{aligned} T(n_0, n_m, a, b) \leq & \underbrace{\sum_{k=1}^{r}}_{\text{line 2}} \Bigg(\underbrace{T(n_0, n_{j_k}, a_0, b_0)}_{\text{line 7}} + \underbrace{T(n_{j_k}, n_m, a_1, b_1)}_{\text{line 10}} \\ & + \underbrace{T(n_0, n_{j_k}, a_0, b_0) T(n_{j_k}, n_m, a_1, b_1)}_{\text{line 11}} \Bigg) \\ & \underbrace{+1}_{\text{line 21}} \\ & \leq r \cdot 4 T(n_0, n_i, a_0, b_0) T(n_i, n_m, a_1, b_1) \end{aligned}$$

Here, the number of iterations r of the for-loop depends on the values of a and b, and n_i is the worst case value for $|G_i|$.

$$\begin{aligned} \delta_{a=0,b>0} &:= \begin{cases} 1, & \text{if } a = 0 \text{ and } b > 0 \\ 0, & \text{else} \end{cases} \\ r &:= (n_m/n_0)^{\delta_{a=0,b>0}} \\ n_i &:= \underset{n \in \{n_{j_1}, \ldots, n_{j_r}\}}{\arg\max} T(n_0, n, a_0, b_0) T(n, n_m, a_1, b_1) \end{aligned}$$

By induction on the recursion depth we will show this upper bound.

$$\begin{aligned} T(n_0, n_m, a, b) &\leq S(n_0, n_m, a, b) \\ &:= 2^{\frac{1}{2}(b-a)(2\log_2(n_m/n_0)-b-a)+4\log_2(n_m/n_0)-4} \\ &\in 2^{\frac{1}{2}(b-a)(2\log_2(n_m/n_0)-b-a)} n^{\mathcal{O}(1)} \end{aligned}$$

For the base case it holds that the algorithm does not have a recursive call. Therefore, it reaches line 21, which can be executed in one step.

$$\begin{aligned} T(n_0, n_m, a, b) &= 1 \\ &\leq 2^{\frac{1}{2}(\underbrace{b-a}_{\geq 0})(\underbrace{2\log_2(n_m/n_0)-b-a}_{\geq 0})+4(\underbrace{\log_2(n_m/n_0)-1}_{\geq 0})} \\ &= S(n_0, n_m, a, b) \end{aligned}$$

For the inductive step, we assume that we have a recursive call and without

loss of generality that $b_0 \leq b_1$. Also, we define $l := \log_2(n_m/n_0)$.

$$
\begin{aligned}
T(n_0, n_m, a, b) \leq & r \cdot 4T(n_0, n_i, a_0, b_0)T(n_i, n_m, a_1, b_1) \\
\leq & 2^{\delta_{a=0,b>0}l+2} && \text{induction hypothesis} \\
& \cdot 2^{\frac{1}{2}(b_0-a_0)(2\log_2(n_i/n_0)-b_0-a_0)+4\log_2(n_i/n_0)-4} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2\log_2(n_m/n_i)-b_1-a_1)+4\log_2(n_m/n_i)-4} \\
\leq & 2^{\delta_{a=0,b>0}l+4l-6} && \log_2(n_i/n_0)+\log_2(n_m/n_i)=l \\
& \cdot 2^{\frac{1}{2}(b_0-a_0)(2\log_2(n_i/n_0)-b_0-a_0)} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2\log_2(n_i/n_0)-b_1-a_1)} \\
\leq & 2^{\delta_{a=0,b>0}l+4l-6} && b_0 \leq \log_2(n_i/n_0) \\
& \cdot 2^{\frac{1}{2}(b_0-a_0)(2b_0-b_0-a_0)} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2b_0-b_1-a_1)} \\
= & 2^{\delta_{a=0,b>0}l+4l-6} \\
& \cdot 2^{\frac{1}{2}b_0(b_0-2a_0-2b_1+2a_1)+\frac{1}{2}a_0^2} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-b_1-a_1)} \\
\leq & 2^{\delta_{a=0,b>0}l+4l-6} && a_0+\delta_{a=0,b>1} \leq b_0 \\
& \cdot 2^{\frac{1}{2}(a_0+\delta_{a=0,b>1})(a_0+\delta_{a=0,b>1}-2a_0-2b_1+2a_1)+\frac{1}{2}a_0^2} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-b_1-a_1)} \\
= & 2^{\delta_{a=0,b>0}l+4l-6} \\
& \cdot 2^{\frac{1}{2}(a_0+\delta_{a=0,b>1})(\delta_{a=0,b>1}-a_0)+\frac{1}{2}a_0^2} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2a_0-2\delta_{a=0,b>1}-b_1-a_1)} \\
= & 2^{\delta_{a=0,b>0}l+4l-6} && a_0\delta_{a=0,b>1}=0 \\
& \cdot 2^{\frac{1}{2}\delta_{a=0,b>1}^2} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2a_0-2\delta_{a=0,b>1}-b_1-a_1)} \\
\leq & 2^{\delta_{a=0,b>0}l+4l-\frac{11}{2}} && \frac{1}{2}\delta_{a=0,b>1} \leq \frac{1}{2} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2a_0-2\delta_{a=0,b>1}-b_1-a_1)} \\
\leq & 2^{\delta_{a=0,b>0}l+4l-\frac{11}{2}} && \delta_{b>1} \leq a_0+\delta_{a=0,b>1} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2\delta_{b>1}-b_1-a_1)} \\
\leq & 2^{\delta_{a=0,b>0}l+4l-\frac{9}{2}} && (b_1-a_1)(\delta_{b>0}-\delta_{b>1}) \leq 1 \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2\delta_{b>0}-b_1-a_1)} \\
= & 2^{\delta_{a=0,b>0}l+4l-\frac{9}{2}} \\
& \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2\delta_{b>0}-(b_1-a_1)-2a_1)}
\end{aligned}
$$

$$
\begin{aligned}
&= 2^{\delta_{a=0,b>0} l + 4l - \frac{9}{2}} && -(b_1-a_1)a_1 = -(b_1-a_1)(a-\delta_{a>0}) \\
&\quad \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2\delta_{b>0}-(b_1-a_1)-2a+2\delta_{a>0})} \\
&= 2^{\delta_{a=0,b>0} l + 4l - \frac{9}{2}} && -\delta_{b>0}+\delta_{a>0} = -\delta_{b>0,a=0} \\
&\quad \cdot 2^{\frac{1}{2}(b_1-a_1)(2l-2\delta_{b>0,a=0}-(b_1-a_1)-2a)} \\
&\leq 2^{\delta_{a=0,b>0} l + 4l - \frac{9}{2}} && b_1-a_1 \leq b-a-\delta_{a=0,b>0} \\
&\quad \cdot 2^{\frac{1}{2}(b-a-\delta_{a=0,b>0})(2l-2\delta_{b>0,a=0}-(b-a-\delta_{a=0,b>0})-2a)} \\
&= 2^{\delta_{a=0,b>0} l + 4l - \frac{9}{2}} \\
&\quad \cdot 2^{\frac{1}{2}(b-a-\delta_{a=0,b>0})(2l-\delta_{b>0,a=0}-b-a)} \\
&= 2^{\delta_{a=0,b>0} l + 4l - \frac{9}{2}} \\
&\quad \cdot 2^{\frac{1}{2}(b-a)(2l-b-a)-\delta_{a=0,b>0} l + \frac{1}{2}\delta^2_{a=0,b>0}+\delta_{a=0,b>0} a} \\
&= 2^{4l-\frac{9}{2}} && \delta_{a=0,b>0} a = 0 \\
&\quad \cdot 2^{\frac{1}{2}(b-a)(2l-b-a)+\frac{1}{2}\delta^2_{a=0,b>0}} \\
&\leq 2^{4l-\frac{8}{2}} && \frac{1}{2}\delta^2_{a=0,b>0} \leq \frac{1}{2} \\
&\quad \cdot 2^{\frac{1}{2}(b-a)(2l-b-a)} \\
&= S(n_0, n_m, a, b)
\end{aligned}
$$

□

Using Algorithm 3.1, we are now able to compute collision sets of composition series for G and H. We follow the idea of [Ros13].

Theorem 3.9 (Bidirectional collision detection)**.** *Collision sets $\mathcal{A}$ and $\mathcal{B}$ are computable in time $\alpha(n)$ and $\beta(n)$ respectively. With either*

1. $\alpha(n) \in n^{\mathcal{O}(1)}$ *and* $\beta(n) \in n^{\frac{1}{2}\log_2(n)+\mathcal{O}(1)}$
   or
2. $\alpha(n) \in n^{\frac{1}{4}\log_2(n)+\mathcal{O}(1)}$ *and* $\beta(n) \in n^{\frac{1}{4}\log_2(n)+\mathcal{O}(1)}$.

Proof. In order to compute the set $\mathcal{A}$, we execute Algorithm 3.1 with input $(1, G, 0, c\log_2(n))$. Similarly, we compute $\mathcal{B}$ by calling Algorithm 3.1 with input $(1, H, c\log_2(n), \log_2(n))$.

Correctness. For a recursion depth lower than $c\log_2(n)$ the algorithm for $\mathcal{A}$ branches maximally and the algorithm for $\mathcal{B}$ branches minimally. This branching manner switches, when the recursion depth exceeds $c\log_2(n)$. The corresponding recursion trees of this proceeding computation are pictured in Figure 3.3. To prove that $\mathcal{A}$ and $\mathcal{B}$ are collision sets, we show that the corresponding recursion trees always share a common path for isomorphic groups

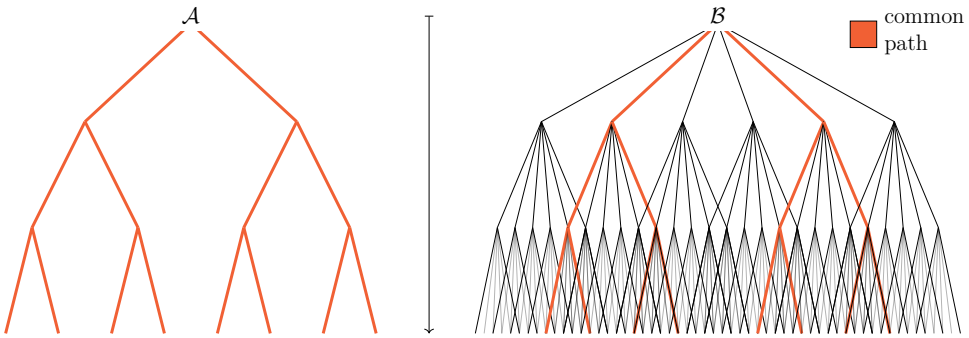

Figure 3.2: The first collision argument.

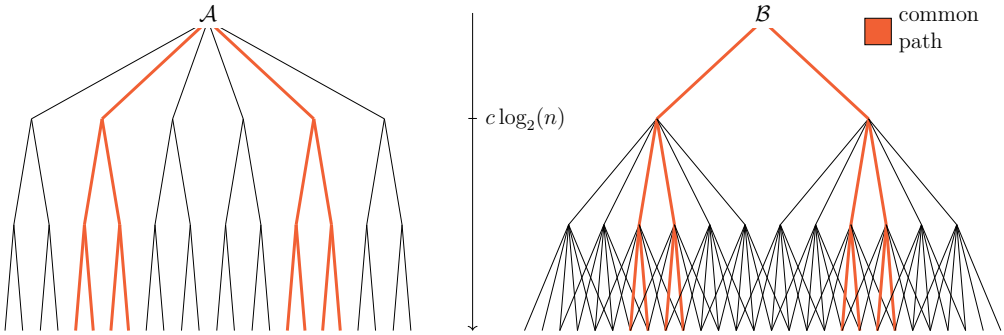

Figure 3.3: The second collision argument.

G and H. This can by illustrated as follows. If the algorithm for $\mathcal{A}$ traces only one minimal normal subgroup G_i, then the algorithm for $\mathcal{B}$ branches maximally to keep track of an isomorphic minimal normal subgroup and vice versa.

Complexity. By Lemma 3.8 the algorithm for $\mathcal{A}$ runs in time

$$\begin{aligned}\alpha(n) &\overset{3.8}{\in} 2^{\frac{1}{2}(b-a)(2\log_2(n)-b-a)} n^{\mathcal{O}(1)} \\ &= 2^{\frac{1}{2}(c\log_2(n))(2\log_2(n)-c\log_2(n))} n^{\mathcal{O}(1)} \\ &= 2^{c(1-\frac{1}{2}c)\log_2(n)^2} n^{\mathcal{O}(1)}.\end{aligned}$$

By the same lemma we get a runtime for $\mathcal{B}$ of

$$\begin{aligned}\beta(n) &\overset{3.8}{\in} 2^{\frac{1}{2}(b-a)(2\log_2(n)-b-a)} n^{\mathcal{O}(1)} \\ &= 2^{\frac{1}{2}(1-c)\log_2(n)(2\log_2(n)-1\log_2(n)-c\log_2(n))} n^{\mathcal{O}(1)} \\ &= 2^{\frac{1}{2}(1-c)^2\log_2(n)^2} n^{\mathcal{O}(1)}.\end{aligned}$$

For case 1, we set $c := 0$ which leads to the recursion trees pictured in Figure 3.2. For 2, we set $c := 1 - \frac{1}{\sqrt{2}}$ instead. □

With the help of collision sets, we are now able to reduce the group isomorphism problem to the isomorphism problem of composition series.

Theorem 3.10. *Let G and H be groups of order n and let $\mathcal{A}$ and $\mathcal{B}$ be collision sets that are computable in time $\alpha(n)$ and $\beta(n)$ respectively. The following holds.*

1. GroupIso *is Turing reducible to* CSeriesIso *in a time complexity of* $(\alpha(n) \cdot \beta(n))n^{\mathcal{O}(1)}$.
2. GroupIso *is Turing reducible to* **Can**(CSeries) *in a time complexity of* $(\alpha(n) + \beta(n))n^{\mathcal{O}(1)}$.

Proof. For case 1, we compute the collision sets $\mathcal{A}$ and $\mathcal{B}$ of the groups G and H respectively. Next, we go through all pairs $(a, b) \in \mathcal{A} \times \mathcal{B}$ and check if a and b are isomorphic. By the definition of collision sets we find an isomorphic pair if and only if $G \cong H$. This leads to an algorithm that runs in time $(\alpha(n) \cdot \beta(n))n^{\mathcal{O}(1)}$. For case 2, we start similarly, by computing the collision sets $\mathcal{A}$ and $\mathcal{B}$. For the next step, we evaluate the canonical form **Can** for each composition series $\mathrm{cs}_G \in \mathcal{A}$ and also for each $\mathrm{cs}_H \in \mathcal{B}$. Subsequently, we sort the evaluated sets $\mathcal{A}$ and $\mathcal{B}$ in lexicographical order in time $|\mathcal{A}| \log_2(|\mathcal{A}|) + |\mathcal{B}| \log_2(|\mathcal{B}|)$. This is still bounded by $(\alpha(n) + \beta(n))n^{\mathcal{O}(1)}$ since we can bound the cardinality by the runtime and get both $|\mathcal{A}| \leq \alpha(n)$ and $|\mathcal{B}| \leq \beta(n)$. Moreover, $\log_2(|\mathcal{A}|)$ and $\log_2(|\mathcal{B}|)$ are bounded by a polynomial due the upper bound on the number of composition series which is $n^{\frac{1}{2}\log_2(n)+\mathcal{O}(1)}$, as shown in [RW15]. For the last step, we must check if the sorted sets $\mathcal{A}$ and $\mathcal{B}$ share a common element, which can be done via merge sort in linear time. □

The rest of this thesis will show how to decide the isomorphism of composition series in polynomial time.

3.2 Reduction from CSeriesIso to QCSeriesAuto

In this section we will reduce the isomorphism problem of composition series to the computation of all isomorphisms between one subnormal series and itself, namely the automorphisms. We shall use the strategy described in [Luk15]. For the reduction, we must introduce another concept, more general than composition series.

Definition 3.11 (Quasi-composition series)**.** *A quasi-composition series qcs_G of a group G is a subnormal series $1 = G_0 \lhd G_1 \lhd \ldots \lhd G_m = G$, in which*

each composition factor G_{i+1}/G_i is simple or a direct product of two simple groups.

The problem we will reduce to is defined as follows.

Problem	QCSeriesAuto
Input	Quasi-composition series of the group G.
Output	Generators for the automorphism group of the quasi-composition series.

To decide isomorphism by finding the automorphism group, we use the same idea as the one used for the well known graph isomorphism problem. Let G and H be graphs. If we know the automorphism group of the disjoint union graph $\mathrm{Aut}(G \dot\cup H)$, we can decide isomorphism of G and H by finding an automorphism in $\mathrm{Aut}(G \dot\cup H)$ which swaps the components G and H. A problem is that this automorphism group may be exponentially large and therefore just be given as a generating set $\mathcal{S}$ with $\langle \mathcal{S} \rangle = \mathrm{Aut}(G \dot\cup H)$. However, one can show that for *connected* graphs G and H such a swapping automorphism exists in $\langle \mathcal{S} \rangle$ if and only if one exists in $\mathcal{S}$ as well. Fortunately, it is easy to show that the graph isomorphism problem for general graphs is polynomial-time equivalent to the isomorphism problem of connected ones. A comparable concept of connectivity in group theory is the direct product decomposition. It is worth acknowledging the following fact about it.

Theorem 3.12 (Krull–Schmidt for finite groups [Rot12])**.** *Every finite group G can uniquely be written as a direct product $G_1 \times G_2 \times \ldots \times G_k$ of directly indecomposable subgroups of G. Here, uniqueness refers to the fact that if there is another such expression $H_1 \times H_2 \times \ldots \times H_l$ of G, then there is a reindexing $\sigma \in Sym(l)$ such that*

1. *$G_i \cong H_{\sigma(i)}$ for all $1 \leq i \leq k = l$.*
2. *$G \cong G_1 \times \ldots \times G_j \times H_{\sigma(j+1)} \times \ldots \times H_{\sigma(k)}$ for all $0 \leq j \leq k = l$.*

Now, we are ready for the proposed reduction.

Theorem 3.13. CSeriesIso *is polynomial-time Turing reducible to* QCSeriesAuto.

Proof. We present Algorithm 3.2.

Correctness. In the case of graphs, we need the graphs to be connected. For the group scenario, the function breaks down the groups to directly indecomposable ones. This procedure is correct since the Krull–Schmidt theorem implies

Algorithm 3.2 Reduction from CSeriesIso to QCSeriesAuto.

function CSeriesIso$\begin{pmatrix} \mathrm{cs}_G := & 1 = G_0 \lhd \ldots \lhd G_m = G, \\ \mathrm{cs}_H := & 1 = H_0 \lhd \ldots \lhd H_m = H \end{pmatrix}$
  if G and H are directly decomposable **then**
    Decompose $G \cong \tilde{G} \times \hat{G}$ and $H \cong \tilde{H} \times \hat{H}$
    Write c_G as $1 \times 1 \lhd \tilde{G}_1 \times 1 \lhd \tilde{G}_2 \times \hat{G}_2 \lhd \tilde{G}_3 \times \hat{G}_3 \lhd \ldots \lhd \tilde{G}_m \times \hat{G}_m$
    Write c_H as $1 \times 1 \lhd \tilde{H}_1 \times 1 \lhd \tilde{H}_2 \times \hat{H}_2 \lhd \tilde{H}_3 \times \hat{H}_3 \lhd \ldots \lhd \tilde{H}_m \times \hat{H}_m$
    $\tilde{c_G} := \quad 1 \lhd \tilde{G}_1 \trianglelefteq \tilde{G}_2 \trianglelefteq \tilde{G}_3 \trianglelefteq \ldots \trianglelefteq \tilde{G}_m$
    $\tilde{c_H} := \quad 1 \lhd \tilde{H}_1 \trianglelefteq \tilde{H}_2 \trianglelefteq \tilde{H}_3 \trianglelefteq \ldots \trianglelefteq \tilde{H}_m$
    $\tilde{\text{c}}\text{heck} := \text{CSeriesIso}(\tilde{c_G},\ \tilde{c_H})$
    $\hat{c_G} := \quad 1 = \hat{G}_1 \trianglelefteq \hat{G}_2 \trianglelefteq \hat{G}_3 \trianglelefteq \ldots \trianglelefteq \hat{G}_m$
    $\hat{c_H} := \quad 1 = \hat{H}_1 \trianglelefteq \hat{H}_2 \trianglelefteq \hat{H}_3 \trianglelefteq \ldots \trianglelefteq \hat{H}_m$
    $\hat{\text{c}}\text{heck} := \text{CSeriesIso}(\hat{c_G},\ \hat{c_H})$
    if $\tilde{\text{c}}\text{heck} == \hat{\text{c}}\text{heck} ==$ **accept then**
      return accept
    else
      return reject
    end if
  else if G and H are directly indecomposable **then**
    $\text{qcs}_{G_m \times H_m} := \quad 1 = G_0 \times H_0 \lhd G_1 \times H_1 \lhd \ldots \lhd G_m \times H_m$
    $\mathcal{S} := \text{QCSeriesAuto}(\text{qcs}_{G_m \times H_m})$
    for all $\varphi \in \mathcal{S}$ **do**
      if $\varphi(G_i \times 1) = 1 \times H_i$ for all $0 \leq i \leq m$ **then**
        return accept
      end if
    end for
    return reject
  end if
end function

that finite groups $G_1 \times G_2$ and $H_1 \times H_2$ are isomorphic if and only if there exists a $\sigma \in \mathrm{Sym}(2)$ such that $G_1 \cong H_{\sigma(1)}$ and $G_2 \cong H_{\sigma(2)}$. To verify if such a $\sigma \in \mathrm{Sym}(2)$ exists, we need to check only one element of $\mathrm{Sym}(2)$ instead of both, as we will see next. The decomposition of the groups in line 3 induces a decomposition of their composition series as written in line 4 and 5. Here, we assume, without loss of generality, that

$$\hat{G}_1 = 1,\ \hat{H}_1 = 1 \text{ and } \tilde{G}_1 \neq 1,\ \hat{H}_1 \neq 1. \tag{3.1}$$

Through the following observation we may spare two isomorphism tests and it suffices to consider the isomorphisms in line 8 and 11.

$$\begin{aligned}
c_G \cong c_H &\Leftrightarrow \exists \varphi : G \overset{\varphi}{\cong} H \text{ and } \varphi(G_i) = H_i \\
&\overset{3,12}{\Leftrightarrow} \exists \varphi, \psi : \tilde{G} \overset{\varphi}{\cong} \tilde{H},\ \hat{G} \overset{\psi}{\cong} \hat{H} \text{ and } \varphi(\tilde{G}_i) = \tilde{H}_i,\ \varphi(\hat{G}_i) = \hat{H}_i \text{ for all } i \\
&\qquad \text{or } \tilde{G} \overset{\varphi}{\cong} \hat{H},\ \hat{G} \overset{\psi}{\cong} \tilde{H} \text{ and } \varphi(\tilde{G}_i) = \hat{H}_i,\ \varphi(\hat{G}_i) = \tilde{H}_i \text{ for all } i \\
&\overset{(3.1)}{\Leftrightarrow} \exists \varphi, \psi : \tilde{G} \overset{\varphi}{\cong} \tilde{H},\ \hat{G} \overset{\psi}{\cong} \hat{H} \text{ and } \varphi(\tilde{G}_i) = \tilde{H}_i,\ \varphi(\hat{G}_i) = \hat{H}_i \text{ for all } i \\
&\Leftrightarrow \tilde{c_G} \cong \tilde{c_H} \text{ and } \hat{c_G} \cong \hat{c_H}
\end{aligned}$$

A technical detail worth mentioning is that the defined composition series in lines 4 - 8 are of the form $1 \trianglelefteq \ldots \trianglelefteq F_i = F_{i+1} \trianglelefteq \ldots \trianglelefteq F_m$. In such a case we merge iteratively and write them as $1 \trianglelefteq \ldots \trianglelefteq F_i \trianglelefteq \ldots \trianglelefteq F_{m-1}$ until we arrive at the notation for composition series.

Next, we discuss the directly indecomposable case. We continue to follow the graph scenario and look for a swapping automorphism, that is an automorphism φ with $\varphi(G_i \times 1) = 1 \times H_i$ for all $0 \leq i \leq m$. It is fairly obvious that the composition series $1 = G_0 \lhd \ldots \lhd G_m$ and $1 = H_0 \lhd \ldots \lhd H_m$ are isomorphic if and only if there is a swapping automorphism $\varphi \in \mathrm{Aut}(\mathrm{qcs}_{G_m \times H_m}) = \langle \mathcal{S} \rangle$. For the correctness of the indecomposable case we claim that $\langle \mathcal{S} \rangle$ contains a swapping automorphism if and only if there is a swapping automorphism in $\mathcal{S}$. For proving the nontrivial direction of the bi-implication, assume that there is a swapping automorphism $\varphi \in \langle \mathcal{S} \rangle$ which therefore maps G to H, in symbols $\varphi(G \times 1) = 1 \times H$. Since $\mathcal{S}$ is a generating set there is a $\psi \in \mathcal{S}$ that does not map G to G, in symbols $\psi(G \times 1) \neq G \times 1$. We claim that $\psi \in \mathcal{S}$ is the swapping automorphism that we are looking for. Notice, that $\psi(G \times 1) \times \psi(1 \times H)$ is a direct product decomposition of $G \times H$ since the subgroups satisfy the

following three conditions.

$$\begin{aligned}
&\psi(G\times 1)\cap\psi(1\times H) &&= \psi(G\times 1\cap 1\times H) &&= \psi(1\times 1) &&= 1\times 1\\
&\psi(G\times 1)\psi(1\times H) &&= \psi((G\times 1)(1\times H)) &&= \psi(G\times H) &&= G\times H\\
&\psi(g,1)\psi(1,h) &&= \psi(g1,1h) &&= \psi(1g,h1) &&= \psi(1,h)\psi(g,1)
\end{aligned}$$

This leads to

$$\begin{aligned}
\psi(G\times 1)\times\psi(1\times H) &\cong G\times H\\
&\cong G\times 1\times 1\times H.
\end{aligned}$$

But we already observed that $\psi(G\times 1)\neq G\times 1$ and both G and H are directly indecomposable, which implies $\psi(G\times 1)=1\times H$. The previous equation then also holds for subgroups.

$$\begin{aligned}
\psi(G_i\times 1) &= \psi((G_i\times H_i)\cap(G\times 1))\\
&= \psi(G_i\times H_i)\cap\psi(G\times 1)\\
&= (G_i\times H_i)\cap(1\times H)\\
&= 1\times H_i && \text{for all } 0\leq i\leq m.
\end{aligned}$$

Therefore, $\psi\in S$ is a swapping automorphism.

Complexity. Every single recursive call of the function CSeriesIso runs in polynomial time since line 3 is executed in polynomial time as showed in [KN09] and in contrast to $\langle\mathcal{S}\rangle$, the set $\mathcal{S}$ is polynomially bounded. The number of recursive calls is bounded by $|G|$ and therefore the total runtime is also polynomially bounded. □

3.3 Reduction from QCSeriesAuto to AutoLift

In this section we describe how to determine the automorphism group of a quasi-composition series by *lifting* the automorphisms of a normal subgroup to the whole group as done in [Luk15].

Definition 3.14 (Almost-solvable)**.** *Let $\mathcal{X}$ be a set. A group $G\leq Sym(\mathcal{X})$ is almost-solvable if $|G/Rad(G)|\leq|\mathcal{X}|^4$.*

Remark 3.15 Solvable groups play a big role in computational group theory because they are a subclass of Γ_d, for which many group theoretic computations can be done in polynomial time [LM11]. Since most of these algorithms

deal with cosets, they can be generalized to almost-solvable groups by using the following divide-and-conquer strategy. At first, we break the group into cosets of the solvable radical, which can be done in polynomial time for permutation groups [Ser03]. By doing this, we reduce the given group theoretic computational problem to $|\mathcal{X}|^4$ sub-problems for solvable groups which can each be solved in polynomial time as assumed. We then combine the interim results to get a conclusive solution for the original computational problem.

Lemma 3.16. *A subgroup of an almost-solvable group is as well almost-solvable.*

Proof. Let S be a subgroup of an almost-solvable group G. Notice that $\mathrm{Rad}(G) \cap S \trianglelefteq S$ since the radical is by definition normal and therefore both $s\mathrm{Rad}(G) = \mathrm{Rad}(G)s$ and $sS = S = Ss$ for all $s \in S$. Recall that $\mathrm{Rad}(S)$ is the maximal normal subgroup of S and so

$$\mathrm{Rad}(G) \cap S \leq \mathrm{Rad}(S). \tag{3.2}$$

Next, we consider the homomorphism $\varphi : S \to G/\mathrm{Rad}(G)$ defined by $s \mapsto s\mathrm{Rad}(G)$. This homomorphism is also know as the natural projection. The kernel of the map is $\mathrm{Rad}(G) \cap S$ and therefore the First isomorphism theorem implies

$$S/(\mathrm{Rad}(G) \cap S) \cong \operatorname{im} \varphi. \tag{3.3}$$

Combining both equations leads to the statement which had to be shown.

$$\begin{aligned} |S/\mathrm{Rad}(S)| &\overset{(3.2)}{\leq} |S/(\mathrm{Rad}(G) \cap S)| \\ &\overset{(3.3)}{=} |\operatorname{im} \varphi| \\ &\leq |G/\mathrm{Rad}(G)| \\ &\leq |\mathcal{X}|^4 \end{aligned}$$

□

Lemma 3.17. *Let S be a simple group or a direct product of simple groups. The group $Aut(S) \leq Sym(S)$ can be computed in polynomial time and the cardinality $|Aut(S)|$ is bounded by $|S|^4$.*

Proof. Every simple group can be generated by two elements as shown in [AG84]. As a consequence S is generated by at most four elements. An automorphism is completely determined by fixing the image for a generating

set, which implies $|\mathrm{Aut}(S)| \leq |S|^4$. This fact also leads to a polynomial-time algorithm which goes through all possible generating sets and all possible corresponding images and then checks if this bijection extends to an automorphism. □

Lemma 3.18. *Let S be a simple group or a direct product of simple groups. The group $Aut(S) \leq Sym(S)$ is almost-solvable.*

Proof. As a consequence of Lemma 3.17 we have

$$|\mathrm{Aut}(S)/\mathrm{Rad}(\mathrm{Aut}(S))| \leq |\mathrm{Aut}(S)| \leq |S|^4.$$

□

Definition 3.19 (Stabilizer subgroup)**.** *The stabilizer subgroup of a group $G \leq Sym(\mathcal{X})$ for $\mathcal{Z} \subseteq \mathcal{X}$ is*

$$G_{\mathcal{Z}} := \{g \in G \mid g(z) \in \mathcal{Z} \quad \forall z \in \mathcal{Z}\}.$$

Example 3.20 For a function $f : \mathcal{X} \to \mathcal{Y}$ and a subset $\mathcal{Z} \subseteq \mathcal{X}$ we define $f(\mathcal{Z}) := \cup_{z \in \mathcal{Z}} f(z)$. This means that a function can be evaluated for the elements of $\mathcal{X}$ and subsets of $\mathcal{X}$, as well. Now, we can view $\mathrm{Sym}(G)$ as a subgroup of $\mathrm{Sym}(G \cup \mathcal{P}(G))$ and we define a stabilizer subgroup for both, a normal subgroup $N \subseteq G \cup \mathcal{P}(G)$ and the set of cosets $G/N \subseteq G \cup \mathcal{P}(G)$.

$$\mathrm{Sym}(G)_{G/N,N} = \left\{ \varphi \in \mathrm{Sym}(G) \;\middle|\; \begin{array}{ll} \varphi(h) \in N & \forall h \in N, \\ \varphi(gN) \in G/N & \forall g \in G \end{array} \right\}.$$

Definition 3.21 (Natural homomorphism)**.** *Let $N \trianglelefteq G$. The natural homomorphism $\Theta : Sym(G)_{G/N,N} \to Sym(G/N) \times Sym(N)$ is defined via $\varphi \mapsto (\varphi|_{G/N}, \varphi|_N)$.*

For a normal subgroup N of G we can now define the following computational problem.

Problem	AutoLift
Input	Generators for $A \leq \mathrm{Aut}(G/N)$ and $B \leq \mathrm{Aut}(N)$, both almost-solvable.
Output	Generators for almost-solvable group $\mathrm{Aut}(G) \cap \Theta^{-1}(A \times B)$.

Now, we are ready to determine the automorphisms of a quasi-composition series by lifting automorphism groups.

Theorem 3.22. QCSeriesAuto *is polynomial-time Turing reducible to* AutoLift.

Algorithm 3.3 Reduction from QCSeriesAuto to AutoLift.

1: **function** QCSeriesAuto($1 = G_0 \lhd \ldots \lhd G_{m-1} \lhd G_m$)
2: **if** $m = 1$ **then**
3: **return** $\mathrm{Aut}(G_m)$
4: **end if**
5: $A := \mathrm{Aut}(G_m/G_{m-1})$
6: $B :=$ QCSeriesAuto($1 = G_0 \lhd \ldots \lhd G_{m-1}$)
7: **return** AutoLift(A, B)
8: **end function**

Proof. We present Algorithm 3.3.

Correctness. By induction on m we show that the algorithm with input $1 = G_0 \lhd \ldots \lhd G_m$ returns $\mathrm{Aut}(G_m)_{G_1,\ldots,G_{m-1}}$ which is almost solvable. For the base case, the algorithm returns $\mathrm{Aut}(G_m)$ which is almost solvable by Lemma 3.18. For the inductive step, the algorithm reaches line 5, in which A is almost-solvable again by Lemma 3.18. The automorphism group B in line 6 equals the almost-solvable group $\mathrm{Aut}(G_{m-1})_{G_1,\ldots,G_{m-2}}$ by the induction hypothesis. Finally, the algorithm computes the lift of the almost-solvable groups A and B and returns the following.

$$
\begin{aligned}
&\mathrm{Aut}(G_m) \cap \Theta^{-1}(\mathrm{Aut}(G_m/G_{m-1}) \times \mathrm{Aut}(G_{m-1})_{G_1,\ldots,G_{m-2}}) \\
=\ &\mathrm{Aut}(G_m) \cap \left\{ \varphi \in \mathrm{Sym}(G_m)_{G_m/G_{m-1},G_{m-1}} \;\middle|\; \begin{array}{l} \varphi|_{G_m/G_{m-1}} \in \mathrm{Aut}(G_m/G_{m-1}), \\ \varphi|_{G_{m-1}} \in \mathrm{Aut}(G_{m-1})_{G_1,\ldots,G_{m-2}} \end{array} \right\} \\
=\ &\left\{ \varphi \in \mathrm{Aut}(G_m)_{G_m/G_{m-1},G_{m-1}} \;\middle|\; \begin{array}{l} \varphi|_{G_m/G_{m-1}} \in \mathrm{Aut}(G_m/G_{m-1}), \\ \varphi|_{G_{m-1}} \in \mathrm{Aut}(G_{m-1})_{G_1,\ldots,G_{m-2}} \end{array} \right\} \\
=\ &\{ \varphi \in \mathrm{Aut}(G_m)_{G_{m-1}} \mid \varphi|_{G_{m-1}} \in \mathrm{Aut}(G_{m-1})_{G_1,\ldots,G_{m-2}} \} \\
=\ &\mathrm{Aut}(G_m)_{G_1,\ldots,G_{m-1}}
\end{aligned}
$$

The almost-solvability of the returned group is guaranteed by the definition of the problem AutoLift and will be proven when we give an algorithm for it in the next section.

Complexity. The automorphism groups in line 3 and 5 can be computed in polynomial time due to Lemma 3.17. Since the recursion depth is m, the algorithm's total runtime is polynomially bounded. □

3.4 Reduction from AutoLift to SetStabilizer

The goal of this section is the reduction of lifting automorphisms to the computational problem set-stabilizer as done in [Luk15]. As in the last section, we use the letter N to denote a normal subgroup of a group G.

Definition 3.23. $L := \left\{ \varphi \in Sym(G)_{G/N,N} \;\middle|\; \begin{array}{c} \varphi(gh) = \varphi(g)\varphi(h), \\ \varphi(hg) = \varphi(h)\varphi(g) \\ \forall g \in G, h \in N \end{array} \right\}.$

Bear in mind that the condition stated in the previous definition of L is slightly weaker than the automorphism condition since h does not need to be in G. By definition, we have the following.

$$\begin{aligned} \mathrm{Aut}(G)_{G/N,N} &\leq L \\ &\leq \mathrm{Sym}(G)_{G/N,N} \\ &= \mathrm{im}\,\Theta^{-1} \end{aligned}$$

Lemma 3.24. *Let $\varphi \in L$. Then $\varphi(ghg^{-1}) = \varphi(g)\varphi(h)\varphi(g)^{-1}$ holds.*

Proof.

$$\begin{aligned} \varphi(ghg^{-1}) &= \underbrace{\varphi(ghg^{-1})}_{\in\varphi(N)}\,\underbrace{\varphi(g)}_{\in\varphi(G)}\,\varphi(g)^{-1} \\ &= \varphi(ghg^{-1}g)\varphi(g)^{-1} \\ &= \varphi(gh)\varphi(g)^{-1} \\ &= \varphi(g)\varphi(h)\varphi(g)^{-1} \end{aligned}$$

□

Lemma 3.25. $\ker\Theta|_L \cong \underbrace{C_N(N) \times \ldots \times C_N(N)}_{|G/N|-1 \text{ times}}$. *Furthermore, generators for $\ker\Theta|_L$ can be computed in polynomial time.*

Proof. Our first claim is that $\varphi \in L$, together with the following two conditions, implies $k \in C_N(N)$.

(*i*) $\varphi(h) = h$ for all $h \in N$.

(*ii*) $\varphi(g) = gk$ and for some $g \in G, k \in N$.

The next equation proves our claim.

$$\begin{aligned}
kh &\overset{(i)}{=} k\varphi(h) \\
&= k\varphi(g)^{-1}\varphi(g)\varphi(h)\varphi(g)^{-1}\varphi(g) \\
&\overset{3.24}{=} k\varphi(g)^{-1}\varphi(\underbrace{ghg^{-1}}_{\in N})\varphi(g) \\
&\overset{(i)}{=} k\varphi(g)^{-1}ghg^{-1}\varphi(g) \\
&\overset{(ii)}{=} k(gk)^{-1}ghg^{-1}gk \\
&= hk && \Leftrightarrow k \in C_N(N) \qquad (3.4)
\end{aligned}$$

Now, we are ready to prove the lemma.

$$\begin{aligned}
\ker\Theta|_L &= \left\{ \varphi \in L \;\middle|\; \begin{array}{l} \forall h \in N : \varphi(h) = h, \\ \forall g \in G : gN = \varphi(gN) = \varphi(g)\varphi(N) = \varphi(g)N \end{array} \right\} \\
&= \left\{ \varphi \in L \;\middle|\; \begin{array}{l} \forall h \in N : \varphi(h) = h, \\ \forall g \in G\ \exists k \in N : \varphi(g) = gk \end{array} \right\} \\
&\overset{(3.4)}{=} \left\{ \varphi \in L \;\middle|\; \begin{array}{l} \forall h \in N : \varphi(h) = h, \\ \forall g \in G\ \exists k \in C_N(N) : \varphi(g) = gk \end{array} \right\} \\
&= \left\langle \varphi_{gN,k} := \left(x \mapsto \begin{cases} xk, & \text{if } x \in gN \\ x, & \text{else} \end{cases} \right) \in L \;\middle|\; k \in C_N(N),\ g \notin N \right\rangle \\
&\cong \underbrace{C_N(N) \times \ldots \times C_N(N)}_{|G/N|-1 \text{ times}}
\end{aligned}$$

It is easily verifiable that elements in the generating set, denoted by $\varphi_{gN,k}$, satisfy the condition stated in the definition of L for any $k \in C_N(N)$. Furthermore, these elements can be computed in polynomial time. □

Definition 3.26 (Inner automorphism)**.** *The inner automorphism $i(g) : N \to N$ with respect to $g \in G$ is defined via $h \mapsto ghg^{-1}$.*

We verify that $i(g)$ is in fact an automorphism of N for any $g \in G$.

$$\begin{aligned}
i(g)(h_1h_2) &= gh_1h_2g^{-1} \\
&= gh_1g^{-1}gh_2g^{-1} \\
&= i(g)(h_1)i(g)(h_2) && \forall h_1, h_2 \in N
\end{aligned}$$

Therefore, the image $i(S) = \cup_{s \in S} i(s)$ is a subgroup of $\mathrm{Aut}(N)$ for any $S \leq G$. Furthermore, $i : G \to \mathrm{Aut}(N)$ is a group homomorphism as seen next.

$$\begin{aligned} i(g_1 g_2)(h) &= g_1 g_2 h (g_1 g_2)^{-1} \\ &= g_1 g_2 h g_2^{-1} g_1^{-1} \\ &= i(g_1)(g_2 h g_2^{-1}) \\ &= i(g_1) i(g_2) h \qquad\qquad \forall g_1, g_2 \in G, h \in N \end{aligned}$$

The next lemma will lend itself to be very useful for proving the correctness of our reduction.

Lemma 3.27. *Let* $C := C_G(N)$ *and*

(⊛) $(\alpha, \beta) \in \mathit{Aut}(G/N) \times \mathit{Aut}(N)$.

The following three assertions are equivalent.

(i) $L \cap \Theta^{-1}(\alpha, \beta)$ *contains at least one element* φ *and can be computed in polynomial time.*

(ii) $\beta i(gN) \beta^{-1} = i(\alpha(gN))\ \forall g \in G$.

(iii) $i^{-1}(\beta i(gN) \beta^{-1}) = \alpha(gN) C\ \forall g \in G$.

Moreover, assertion (i) *implies:*

(iv) $\alpha \in \mathit{Aut}(G/N)_{CN/N}$.

And assertion (iv) *implies:*

(v) α *induces an automorphism* $\tilde{\alpha} \in \mathit{Aut}(G/CN)$.

Proof. For "$(i) \Rightarrow (ii)$" we observe the following.

$$\begin{aligned} \beta i(gN) \beta^{-1} &\overset{(i)}{=} \varphi i(gN) \varphi^{-1} \\ &= \{h \mapsto \varphi(\tilde{g} \varphi^{-1}(h) \tilde{g}^{-1}) \mid \tilde{g} \in gN\} \\ &\overset{3.24}{=} \{h \mapsto \varphi(\tilde{g}) h \varphi(\tilde{g})^{-1} \mid \tilde{g} \in gN\} \\ &= i(\varphi(gN)) \\ &\overset{(i)}{=} i(\alpha(gN)) \end{aligned}$$

Next, we prove "$(i) \Rightarrow (iv)$".

$$\begin{aligned}
\alpha(CN/N) &\overset{(i)}{=} \varphi(CN/N) \\
&\overset{(i)}{=} \varphi(C)\varphi(N)/\varphi(N) \\
&\overset{(i)}{=} \varphi(C)N/N \\
&= \varphi(\{g \in G \mid \quad gh = hg \quad \forall h \in N\})N/N \\
&= \{\varphi(g) \in G \mid \quad gh = hg \quad \forall h \in N\}N/N \\
&= \{\varphi(g) \in G \mid \varphi(gh) = \varphi(hg) \quad \forall h \in N\}N/N \\
&\overset{(i)}{=} \{\varphi(g) \in G \mid \varphi(g)\varphi(h) = \varphi(h)\varphi(g) \quad \forall h \in N\}N/N \\
&\overset{(i)}{=} \{\varphi(g) \in G \mid \varphi(g)h = h\varphi(g) \quad \forall h \in N\}N/N \\
&= CN/N
\end{aligned}$$

For "$(iv) \Rightarrow (v)$" we need to verify that the induced function, defined as follows, is a well-defined automorphism.

$$\tilde{\alpha}(gCN) := \alpha(gN)C \quad \forall g \in G$$

To show that $\tilde{\alpha}$ is well-defined, we need $\tilde{\alpha}(gkCN) = \tilde{\alpha}(gCN)$ for all $k \in CN$.

$$\begin{aligned}
\tilde{\alpha}(gkCN) &= \alpha(gkN)C \\
&\overset{(\circledast)}{=} \underbrace{\alpha(gN)}_{\in G/N}\underbrace{\alpha(kN)}_{\in CN/N}C \\
&\overset{(iv)}{=} \alpha(gN)C \\
&= \tilde{\alpha}(gCN)
\end{aligned}$$

Now, we observe that C is normal in G.

$$\begin{aligned}
gCg^{-1} &= g\{k \in G \mid \quad kh = hk \quad \forall h \in N\}g^{-1} \\
&= \{gkg^{-1} \in G \mid \quad kh = hk \quad \forall h \in N\} \\
&= \{k \in G \mid g^{-1}kgh = hg^{-1}kg \quad \forall h \in N\} \\
&= \{k \in G \mid kghg^{-1} = ghg^{-1}k \quad \forall h \in N\} \\
&= \{k \in G \mid \quad kh = hk \quad \forall h \in gNg^{-1}\} \\
&= \{k \in G \mid \quad kh = hk \quad \forall h \in N\} \\
&= C
\end{aligned} \tag{3.5}$$

The next equation implies that the function $\tilde{\alpha}$ is an automorphism.

$$\begin{aligned}\tilde{\alpha}(g_1CN)\tilde{\alpha}(g_2CN) &= \alpha(g_1N)C\alpha(g_2N)C \\ &\overset{(3.5)}{=} \alpha(g_1N)\alpha(g_2N)C \\ &\overset{(\circledast)}{=} \alpha(g_1g_2N)C \\ &= \tilde{\alpha}(g_1g_2CN) \qquad\qquad \forall g_1, g_2 \in G\end{aligned}$$

To prove "$(ii) \Leftrightarrow (iii)$", we claim that i is a bijection between G/CN and $i(G/CN)$. It is worth remarking that the following holds by definition.

$$i(C) = \{\mathrm{id}\} = \{e\} \tag{3.6}$$

Since we already noticed that i is a homomorphism, it suffices to show that $\ker i$ is trivial.

$$\begin{aligned}i(gCN) = i(CN) &\Leftrightarrow i(g)i(C)i(N) = i(C)i(N) \\ &\overset{(3.6)}{\Leftrightarrow} i(g)i(N) = i(N) \\ &\Leftrightarrow i(g) \in i(N) \\ &\Leftrightarrow \exists h \in N\ \forall x \in N : gxg^{-1} = hxh^{-1} \\ &\Leftrightarrow \exists h \in N\ \forall x \in N : h^{-1}gx = xh^{-1}g \\ &\Leftrightarrow \exists h \in N : h^{-1}g \in C \\ &\Leftrightarrow g \in CN \\ &\Leftrightarrow gCN = CN\end{aligned}$$

Since i^{-1} is a bijection as well, we get the desired equivalence.

$$\begin{aligned}\beta i(gN)\beta^{-1} = i(\alpha(gN)) \quad &\overset{(3.6)}{\Leftrightarrow} \quad \beta i(gN)\beta^{-1} = i(\alpha(gN)C) \\ &\Leftrightarrow \quad i^{-1}(\beta i(gN)\beta^{-1}) = \alpha(gN)C\end{aligned}$$

Last but not least, we need to prove "$(ii) \Rightarrow (i)$". At first, we fix a representative $\tilde{g}$ for each coset of N. Then, for each coset $\tilde{g}N$, we

$$\text{fix } a_{\tilde{g}} \in \alpha(\tilde{g}N) \overset{(\circledast)}{=} a_{\tilde{g}}N \text{ such that } i(a_{\tilde{g}}) \overset{(ii)}{=} \beta i(\tilde{g})\beta^{-1}. \tag{3.7}$$

We define

$$\varphi(x) := \begin{cases} \beta(x), & \text{if } x \in N \\ a_{\tilde{g}}\beta(\tilde{g}^{-1}x), & \text{if } x \in \tilde{g}N \text{ for a fixed a representative } \tilde{g} \notin N \end{cases}. \tag{3.8}$$

We claim that φ lies in the set $L \cap \Theta^{-1}(\alpha, \beta)$ which we want to compute. To show $\varphi \in \Theta^{-1}(\alpha, \beta)$, we notice that

$$\begin{aligned} \varphi|_{G/N}(\tilde{g}N) &\overset{(3.8)}{=} a_{\tilde{g}}\beta(\tilde{g}^{-1}\tilde{g}N) \\ &= a_{\tilde{g}}\beta(N) \\ &\overset{(\circledast)}{=} a_{\tilde{g}}N \\ &\overset{(3.7)}{=} \alpha(\tilde{g}N) \end{aligned} \qquad \text{and} \qquad \varphi|_N(h) \overset{(3.8)}{=} \beta(h).$$

To prove the claim, we show $\varphi \in L$. Since G can be decomposed into cosets, it suffices to check the condition of L for an arbitrary coset $\tilde{g}N$.

$$\begin{aligned} \varphi(gh) &\overset{(3.8)}{=} a_{\tilde{g}}\beta(\tilde{g}^{-1}gh) \\ &\overset{(\circledast)}{=} a_{\tilde{g}}\beta(\tilde{g}^{-1}g)\beta(h) \\ &\overset{(3.8)}{=} \varphi(g)\varphi(h) \end{aligned} \qquad \forall g \in \tilde{g}N, h \in N \tag{3.9}$$

$$
\begin{aligned}
\varphi(hg) &= \varphi(gg^{-1}hg) \\
&\overset{(3.9)}{=} \varphi(g)\varphi(g^{-1}hg) \\
&\overset{(3.8)}{=} \varphi(g)\beta(g^{-1}hg) \\
&= \varphi(g)\beta i(g)^{-1}(h) \\
&= \varphi(g)\beta i(\tilde{g}\tilde{g}^{-1}g)^{-1}(h) \\
&= \varphi(g)(i(\tilde{g})i(\tilde{g}^{-1}g)\beta^{-1})^{-1}(h) \\
&= \varphi(g)(\beta^{-1}\beta i(\tilde{g})\beta^{-1}\beta i(\tilde{g}^{-1}g)\beta^{-1})^{-1}(h) \\
&\overset{(3.7)}{=} \varphi(g)(\beta^{-1}i(a_{\tilde{g}})\beta i(\tilde{g}^{-1}g)\beta^{-1})^{-1}(h) \\
&\overset{(\circledast)}{=} \varphi(g)(\beta^{-1}i(a_{\tilde{g}})i(\beta(\tilde{g}^{-1}g)))^{-1}(h) \\
&= \varphi(g)(\beta^{-1}i(a_{\tilde{g}}\beta(\tilde{g}^{-1}g)))^{-1}(h) \\
&\overset{(3.8)}{=} \varphi(g)(\beta^{-1}i(\varphi(g)))^{-1}(h) \\
&= \varphi(g)i(\varphi(g))^{-1}\beta(h) \\
&= \varphi(g)\varphi(g)^{-1}\beta(h)\varphi(g) \\
&= \beta(h)\varphi(g) \\
&\overset{(3.8)}{=} \varphi(h)\varphi(g) \qquad\qquad \forall g \in \tilde{g}N, h \in N
\end{aligned}
$$

Therefore, $\varphi \in L \cap \Theta^{-1}(\alpha, \beta)$. Now, we can calculate the whole set $L \cap \Theta^{-1}(\alpha, \beta) = \varphi \ker \Theta|_L$ in polynomial time by applying Lemma 3.25. □

We will reduce to the following problem.

Problem	SetStabilizer
Input	Generators for the almost-solvable group $G \leq \mathrm{Sym}(\mathcal{X})$, $\mathcal{Z} \subseteq \mathcal{X}$.
Output	Generators for the stabilizer subgroup $G_{\mathcal{Z}}$.

Theorem 3.28. AutoLift *is polynomial-time Turing reducible to* SetStabilizer.

Proof. We present Algorithm 3.4.

Correctness. Since we are looking for $\mathrm{Aut}(G) \cap \Theta^{-1}(A \times B)$, we only need to compute the set $\mathrm{Aut}(G) \cap \Theta^{-1}(\alpha, \beta)$ for all pairs (α, β), for which the set is not empty. For such a pair Lemma 3.27 implies $\alpha \in \mathrm{Aut}(G/N)_{CN/N}$ and therefore the cut down in line 2 is correct. With the same argumentation, we justify the

Algorithm 3.4 Reduction from AutoLift to SetStabilizer.

1: **function** AutoLift($A \leq \mathrm{Aut}(G/N),\ B \leq \mathrm{Aut}(N)$)
2: $\quad \tilde{A} := A_{CN/N}$
3: $\quad \tilde{B} := \{\beta \in B \mid \beta^{-1} i(G) \beta = i(G)\}$
4: $\quad D := \{(\alpha, \beta) \in \tilde{A} \times \tilde{B} \mid i^{-1}(\beta i(gN)\beta^{-1}) = \alpha(gN)C\ \forall g \in G\}$
5: $\quad E := \Theta(D)^{-1} \cap L$
6: $\quad$ **return** $E \cap \mathrm{Aut}(G)$
7: **end function**

cut down of B in line 3, since Lemma 3.27 implies the following.

$$\begin{aligned} \beta i(G)\beta^{-1} &= \beta i(\bigcup_{gN \in G/N} gN)\beta^{-1} \\ &\overset{3.27}{=} i(\alpha(\bigcup_{gN \in G/N} gN)) \\ &= i(\bigcup_{gN \in G/N} \alpha(gN)) \\ &= i(G) \end{aligned}$$

Then again, by Lemma 3.27 we deduce the correctness of the cut down in line 4. For line 5 we notice that $\mathrm{Aut}(G) \subseteq L$ and therefore we preserve every solution. Finally, we reach line 6 and cut down to $\mathrm{Aut}(G)$. Now we are allowed to return the computed set, since it is already restricted to $\Theta^{-1}(A \times B)$ due to line 5.

Complexity. We show that the computations can be done in polynomial time using SetStabilizer as oracle. Line 2 can be expressed as a set-stabilizer problem for the almost-solvable group $A \leq \mathrm{Sym}(G/N)$ stabilizing the subset CN/N. Next, we discuss line 3. In [LM11] Luks and Miyazaki showed that for $H, G \leq \mathrm{Sym}(\mathcal{X})$ the normalizer $N_G(H) := \{g \in G \mid gHg^{-1} = H\}$ can be found in polynomial time if G lies in Γ_d. This algorithm can be extended to almost-solvable groups by using the divide-and-conquer strategy described in Remark 3.15. To compute line 3, we express the group $\tilde{B}$ as a normalizer $N_B(i(G))$. The almost-solvability of B ensures polynomial time. Remark that line 3 could also be expressed as a set-stabilizer as in [Luk15] if we would use an unstated property that B has due to our previous reduction. Next, we show how to express line 4 as a set-stabilizer problem. We define a group action for the group $\tilde{B} \leq B$, we know to be almost-solvable by Lemma 3.16.

$$\beta \bullet gCN := i^{-1}(\beta i(gN)\beta^{-1}) \in gCN$$

This definition is valid since $\beta i(gN)\beta^{-1} \in i(G)$ due to line 3. The group $\tilde{A} \leq A$ is almost-solvable for the same reason that $\tilde{B}$ is. The cut down in line 2 together with Lemma 3.27 implies that $\alpha \in \tilde{A}$ induces an automorphism $\tilde{\alpha} \in \mathrm{Aut}(G/CN)$ and therefore the following is a well-defined group action.

$$\alpha \bullet gCN := \tilde{\alpha}(gCN) = \alpha(gN)C \in gCN$$

Moreover, the direct product of two almost-solvable groups is almost-solvable as well.

$$\begin{aligned} |(\tilde{A} \times \tilde{B})/\mathrm{Rad}(\tilde{A} \times \tilde{B})| &= |\tilde{A}/\mathrm{Rad}(\tilde{A})| \cdot |\tilde{B}/\mathrm{Rad}(\tilde{B})| \\ &\leq |G/N|^4 |N|^4 \\ &= |(G/N) \times N|^4 \end{aligned}$$

Now, line 4 can be expressed as a set-stabilizer problem for the almost-solvable group $\tilde{A} \times \tilde{B}$ acting on the set $G/CN \times G/CN$, stabilizing the diagonal subset $\{(gCN, gCN) \in G/CN \times G/CN\}$. Next, we show how to compute line 5 in polynomial time. Since Lemma 3.27 shows that $L \cap \Theta^{-1}(\alpha, \beta) \neq \emptyset$ for all $(\alpha, \beta) \in D$, we have by the First isomorphism theorem that

$$\begin{aligned} (L \cap \Theta^{-1}(D))/\ker \Theta|_L \cong \Theta|_L(L \cap \Theta^{-1}(D)) \\ = D. \end{aligned} \tag{3.10}$$

Therefore, $L \cap \Theta^{-1}(D)$ is generated by the generating automorphisms of $\ker \Theta|_L$ together with the lifts $L \cap \Theta^{-1}(\alpha, \beta)$ of the generators $(\alpha, \beta) \in D$. Both can be computed in polynomial time due to Lemma 3.25 and Lemma 3.27 respectively. To express line 6 as a set-stabilizer problem, we require E to be almost-solvable. The next equation shows that E is almost-solvable if D is.

$$\begin{aligned} |E/\mathrm{Rad}(E)| &\overset{(3.10)}{=} |(D \times \ker \Theta|_L)/\mathrm{Rad}(D \times \ker \Theta|_L)| \\ &= |D/\mathrm{Rad}(D)| \cdot |\ker \Theta|_L/\mathrm{Rad}(\ker \Theta|_L)| \\ &\overset{(3.25)}{=} |D/\mathrm{Rad}(D)| \end{aligned}$$

But since we know an almost-solvable overgroup $\tilde{A} \times \tilde{B} \geq D$, we can deduce the almost-solvability of D from Lemma 3.16. Now, we can express the last line as a set-stabilizer problem for the almost-solvable group E acting on G^3, stabilizing the set $\{(g_1, g_2, g_3) \in G^3 \mid g_1 g_2 = g_3\}$. □

3.5 SetStabilizer

At first, we generalize stabilizer subgroups to cosets.

Definition 3.29 (Stabilizer coset)**.** *Let $N \trianglelefteq G \leq Sym(\mathcal{X})$ and $\mathcal{Z}, \mathcal{W} \subseteq \mathcal{X}$ such that $N(\mathcal{W}) = \mathcal{W}$. The stabilizer coset of gN for $\mathcal{Z}, \mathcal{W}$ is*

$$gN_{\mathcal{Z}|\mathcal{W}} := \{gh \in gN \mid gh(\mathcal{Z} \cap \mathcal{W}) = \mathcal{Z} \cap g(\mathcal{W})\}.$$

The stabilizer coset is in fact a generalization.

$$\begin{aligned} G_{\mathcal{Z}|\mathcal{X}} &= \{eg \in eG \mid eg(\mathcal{Z} \cap \mathcal{X}) = \mathcal{Z} \cap e(\mathcal{X})\} \\ &= \{g \in G \mid g(\mathcal{Z} \cap \mathcal{X}) = \mathcal{Z} \cap \mathcal{X}\} \\ &= \{g \in G \mid g(\mathcal{Z}) = \mathcal{Z}\} \\ &= G_{\mathcal{Z}} \end{aligned}$$

Theorem 3.30. SetStabilizer *is polynomial-time computable.*

Proof. To compute $G_{\mathcal{Z}|\mathcal{X}}$, we execute Algorithm 3.5 with input $(G, \mathcal{Z}, \mathcal{X})$.

Algorithm 3.5 SetStabilizer.

function SetStabilizer($gN \in G/N,\ \mathcal{Z} \subseteq \mathcal{X},\ \mathcal{W} \subseteq \mathcal{X}$)
  if $\mathcal{W} = \{w\}$ **then** ▷ Base case
    if $|\mathcal{Z} \cap \{w, g(w)\}| = 1$ **and** $w \neq g(w)$ **then**
      $gN_{\mathcal{Z}|\mathcal{W}} := \emptyset$
    else
      $gN_{\mathcal{Z}|\mathcal{W}} := gN$
    end if ▷ Intransitive case
  else if $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2$ such that $N(\mathcal{W}_i) = \mathcal{W}_i$ for $i = 1, 2$ **then**
    $gN_{\mathcal{Z}|\mathcal{W}_1} :=$ SetStabilizer($gN,\ \mathcal{Z},\ \mathcal{W}_1$)
    $gN_{\mathcal{Z}|\mathcal{W}} :=$ SetStabilizer($gN_{\mathcal{Z}|\mathcal{W}_1},\ \mathcal{Z},\ \mathcal{W}_2$)
  else ▷ Transitive case
    Decompose $\mathcal{W} = \mathcal{W}_1 \cup \ldots \cup \mathcal{W}_m$ with a minimal $m \geq 2$ such that N preserves the partition: $N(\{\mathcal{W}_1, \ldots, \mathcal{W}_m\}) = \{\mathcal{W}_1, \ldots, \mathcal{W}_m\}$
    $N_{\text{stab}} := \{h \in N \mid h(\mathcal{W}_i) = \mathcal{W}_i \quad \forall i \in \{1, \ldots, m\}\}$
    Decompose $N = \bigcup_{1 \leq j \leq |N/N_{\text{stab}}|} g_j N_{\text{stab}}$
    $gN_{\mathcal{Z}|\mathcal{W}} := \bigcup_{1 \leq j \leq |N/N_{\text{stab}}|}$ SetStabilizer($gg_j N_{\text{stab}},\ \mathcal{Z},\ \mathcal{W}$)
  end if
  return $gN_{\mathcal{Z}|\mathcal{W}}$
end function

Correctness. By induction on the recursion depth we show that Algorithm 3.5 with input $(gN, \mathcal{Z}, \mathcal{X})$ returns $gN_{\mathcal{Z}|\mathcal{W}}$. For the base case the algorithm does

not have a recursive call and returns the following.

$$\begin{aligned} gN_{\mathcal{Z}|\mathcal{W}} &= \{gh \in gN \mid \quad gh(\mathcal{Z} \cap \mathcal{W}) = \mathcal{Z} \cap g(\mathcal{W}) \quad \} \\ &= \{gh \in gN \mid gh(\mathcal{Z} \cap \{w\}) = \mathcal{Z} \cap \{g(w)\}\} \\ &= \{gh \in gN \mid \quad g(\mathcal{Z} \cap \{w\}) = \mathcal{Z} \cap \{g(w)\}\} \\ &= \begin{cases} \emptyset, & \text{if } |\mathcal{Z} \cap \{w, g(w)\}| = 1 \text{ and } w \neq g(w) \\ gN, & \text{else} \end{cases} \end{aligned}$$

For the inductive step we assume a recursive call. We first show the correctness of the intransitive case. Since N acts intransitively we have $N(\mathcal{W}_i) = \mathcal{W}_i$ for $i = 1, 2$ and therefore the following.

$$\begin{aligned} gh(\mathcal{Z} \cap \mathcal{W}_i) \cap \mathcal{Z} \cap g(\mathcal{W}_j) &\subseteq gh(\mathcal{W}_i) \cap g(\mathcal{W}_j) \\ &\subseteq g(\mathcal{W}_i) \cap g(\mathcal{W}_j) \\ &\subseteq g(\mathcal{W}_i \cap \mathcal{W}_j) \\ &= \emptyset \qquad \text{for } \{i, j\} = \{1, 2\} \end{aligned} \tag{3.11}$$

The next equation ensures the correctness of the intransitive case.

$$\begin{aligned} gN_{\mathcal{Z}|\mathcal{W}} &= gN_{\mathcal{Z}|\mathcal{W}_1 \uplus \mathcal{W}_2} \\ &= \left\{ gh \in gN \;\middle|\; \begin{array}{lll} & gh(\mathcal{Z} \cap (\mathcal{W}_1 \uplus \mathcal{W}_2)) & \\ = & \mathcal{Z} \cap g(\mathcal{W}_1 \uplus \mathcal{W}_2) & \end{array} \right\} \\ &= \left\{ gh \in gN \;\middle|\; \begin{array}{lll} & gh((\mathcal{Z} \cap \mathcal{W}_1) \uplus (\mathcal{Z} \cap \mathcal{W}_2)) & \\ = & \mathcal{Z} \cap (g(\mathcal{W}_1) \uplus g(\mathcal{W}_2)) & \end{array} \right\} \\ &= \left\{ gh \in gN \;\middle|\; \begin{array}{lll} & gh(\mathcal{Z} \cap \mathcal{W}_1) \quad \uplus \quad gh(\mathcal{Z} \cap \mathcal{W}_2) & \\ = & (\mathcal{Z} \cap g(\mathcal{W}_1)) \quad \uplus \quad (\mathcal{Z} \cap g(\mathcal{W}_2)) & \end{array} \right\} \\ &\overset{(3.11)}{=} \{gh \in gN \quad \mid gh(\mathcal{Z} \cap \mathcal{W}_i) = \mathcal{Z} \cap g(\mathcal{W}_i) \text{ for } i = 1, 2 \} \\ &= \{gh \in gN_{\mathcal{Z}|\mathcal{W}_1} \mid gh(\mathcal{Z} \cap \mathcal{W}_2) = \mathcal{Z} \cap g(\mathcal{W}_2) \quad \} \\ &= (gN_{\mathcal{Z}|\mathcal{W}_1})_{\mathcal{Z}|\mathcal{W}_2} \end{aligned}$$

Next, we show the correctness of the transitive case.

$$\begin{aligned} gN_{\mathcal{Z}|\mathcal{W}} &= \{gh \in gN \mid gh(\mathcal{Z} \cap \mathcal{W}) = \mathcal{Z} \cap g(\mathcal{W}) \} \\ &= \{gh \in \bigcup_{1 \le j \le |N/N_{\text{stab}}|} gg_j N_{\text{stab}} \mid gh(\mathcal{Z} \cap \mathcal{W}) = \mathcal{Z} \cap g(\mathcal{W}) \} \\ &= \bigcup_{1 \le j \le |N/N_{\text{stab}}|} \{gg_j k \in gg_j N_{\text{stab}} \mid gg_j k(\mathcal{Z} \cap \mathcal{W}) = \mathcal{Z} \cap gg_j(\mathcal{W})\} \\ &= \bigcup_{1 \le j \le |N/N_{\text{stab}}|} (gg_j N_{\text{stab}})_{\mathcal{Z}|\mathcal{W}} \end{aligned}$$

Complexity. It suffices to be shown that the set-stabilizer algorithm runs in polynomial time for solvable groups since the almost-solvable case can reduced to the aforementioned case by using the divide-and-conquer strategy described in Remark 3.15. The reason for the tractability of this case is that there is a constant c such that the orders of all primitive solvable groups $G \le \text{Sym}(\mathcal{W})$ are bounded by $|\mathcal{W}|^c$ as shown in [BCP82]. Here, a primitive group $G \le \text{Sym}(\mathcal{W})$ is one for which a decomposition, such as in line 12 would be the trivial one, namely $\mathcal{W} = \uplus_{w \in \mathcal{W}} \{w\}$. In fact, the factor group N/N_{stab} acting on $\{\mathcal{W}_1, \ldots, \mathcal{W}_m\}$ is primitive and therefore its order is bounded by $|\{\mathcal{W}_1, \ldots, \mathcal{W}_m\}|^c = m^c$. Furthermore, one can see that the transitive and intransitive case alternate as in Example 3.31. Through such an alternation the algorithm reduces a problem of size $|\mathcal{W}|$ to $|N/N_{\text{stab}}|$ many calls, each of size $|\mathcal{W}_i|$ and therefore the total number of recursive calls $R(|\mathcal{W}|)$ satisfies the following recursion.

$$\begin{aligned} R(|\mathcal{W}|) &= |N/N_{\text{stab}}| \cdot R(|\mathcal{W}_i|) \\ &\le m^c \cdot R(|\mathcal{W}|/m) \end{aligned}$$

By induction, the total number of recursive calls equals $R(|\mathcal{W}|) \le |\mathcal{W}|^c$. Moreover, each recursive call, in particular the computations in line 12 and 13, can be done in polynomial time as shown in [Luk82].

□

Example 3.31 We call Algorithm 3.5 with the following input.

$$\begin{aligned} gN &= \text{Sym}(3) = \langle \textbf{(1 2)}, \textbf{(2 3)} \rangle \\ \mathcal{Z} &= \{123{,}321\} \\ \mathcal{W} &= \{x_1x_2x_3 \mid \{x_1, x_2, x_3\} = \{1, 2, 3\}\} \end{aligned}$$

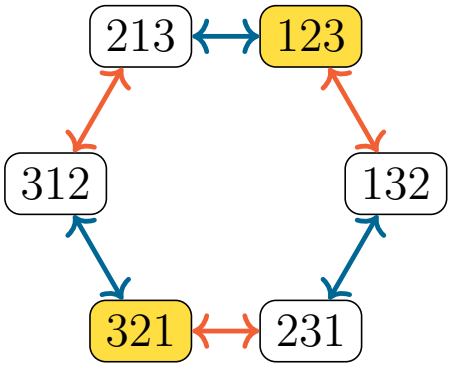

Figure 3.4: The group Sym(3) acting on strings.

It is worth remarking that the symmetric group is acting on the set $\mathcal{W}$ via $(\sigma, x_1x_2x_3) \mapsto \sigma(x_1)\sigma(x_2)\sigma(x_3)$ as seen in Figure 3.4. A segment of the corresponding recursion tree is pictured in Figure 3.5. Notice that we naturally extended the intransitive case for $\mathcal{W} = \mathcal{W}_1 \dot\cup \mathcal{W}_2 \dot\cup \mathcal{W}_3$ as seen in the deepest level of this recursion tree.

3.6 Conclusion

Corollary 3.32. GroupIso *is decidable in a time complexity of* $n^{\frac{1}{2}\log_2(n)+\mathcal{O}(1)}$.

Proof. First, we compute one composition series with the tree property of G and enumerate all composition series with the tree property of H. These collision sets $\mathcal{A}$ and $\mathcal{B}$ can be computed in time $n^{\frac{1}{2}\log_2(n)+\mathcal{O}(1)}$ and $n^{\mathcal{O}(1)}$ respectively as shown in Theorem 3.91. To decide isomorphism of G and H, we test isomorphism of the composition series of G and the enumerated composition series of H. By Theorem 3.101 this takes the desired time of $n^{\frac{1}{2}\log_2(n)+\mathcal{O}(1)}$ if we can decide isomorphism of two composition series in polynomial time. In order to decide isomorphism of two composition series, we join both and check if there is a swapping automorphism in the automorphism group of the quasi-composition series. This Turing reduction is stated in Theorem 3.13. This automorphism group can, in turn, be calculated by starting with the automorphisms of a small subgroup and lifting these automorphisms step by step as described in Theorem 3.22. Next, Theorem 3.28 shows that the deepest tools for lifting automorphisms can be described as set-stabilizer problems for almost-solvable groups. Lastly and most importantly, we solve the set-stabilizer problem for this class of groups in polynomial time by a divide-and-conquer strategy stated in Theorem 3.30. □

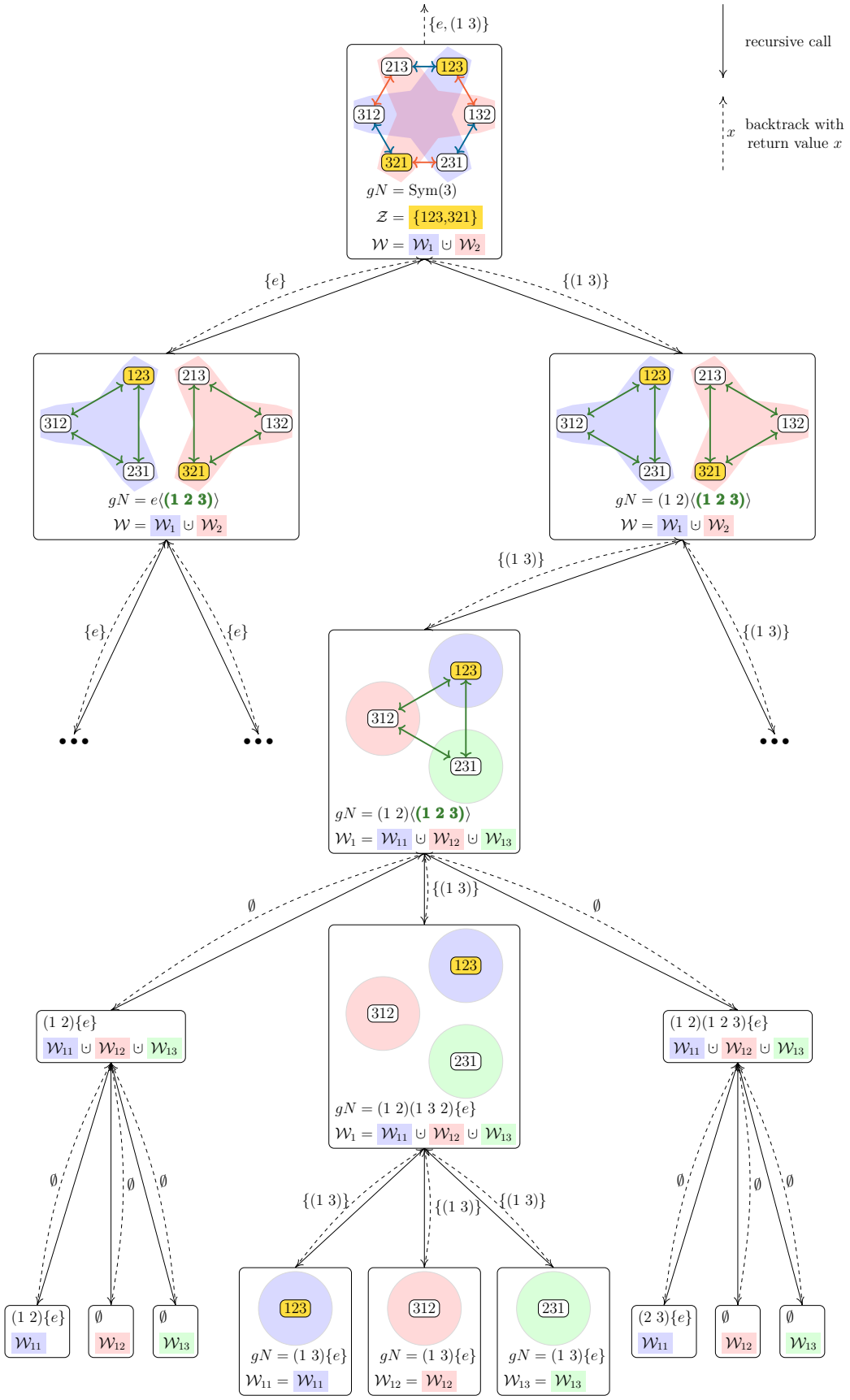

Figure 3.5: Recursion tree of Algorithm 3.5.

4 Isomorphism of Special Groups

We give a summary of the complexity results for the isomorphism problem of restricted groups.

Group class	**Complexity**	**Reference**
Abelian groups.	$\mathcal{O}(n)$	[Kav07]
Quotients of generalized Heisenberg groups.	$(\log(n)+p)^{\mathcal{O}(1)}$	[LW12]
Groups of genus 2, that are p-groups of exponent p and exponent p-class 2 and commutator of order dividing p^2.	$n^{\mathcal{O}(1)}$	[BMW15]
Semisimple groups, that are groups with abelian sylow towers.	$n^{\mathcal{O}(1)}$	[BQ12]
Groups with normal hall subgroup.	$n^{\mathcal{O}(1)}$	[QST12]
Groups that are tame extensions, that are groups G where the sylow p-subgroups of $G/\mathbb{Z}_p^d$ are cyclic, dihedral, semi-dihedral, or generalized quaternion.	$n^{\mathcal{O}(1)}$	[GQ15]
Groups G with central radical and where $G/\mathrm{Rad}(G)$ has $\mathcal{O}(\log(n)/\log(\log(n)))$ minimal normal subgroups.	$n^{\mathcal{O}(1)}$	[GQ14]
Groups with central radical.	$n^{\frac{1}{60}\log(\log(n))+\mathcal{O}(1)}$	[GQ14]

Table 4.1: Complexity of the isomorphism problem of restricted groups.

5 Outlook

We give an outlook to expected results. Eugene M. Luks claimed to publish a follow-up paper to [Luk15], in which he attempts to extend his results of the polynomial time computability of composition series to a canonical version. This would allow us to apply the latter parts of Theorems 3.9 and 3.10, which would result in a new time bound of $n^{\frac{1}{4}\log_2(n)+\mathcal{O}(1)}$ for GroupIso due a more astute collision argument.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AG84] Michael Aschbacher and R Guralnick. Some applications of the first cohomology group. *Journal of Algebra*, 90(2):446–460, 1984.

[BCP82] Lázló Babai, Peter J Cameron, and Péter Pál Pálfy. On the orders of primitive groups with restricted nonabelian composition factors. *Journal of Algebra*, 79(1):161–168, 1982.

[BMW15] P. A. Brooksbank, J. Maglione, and J. B. Wilson. A fast isomorphism test for groups of genus 2. *ArXiv e-prints*, August 2015.

[BQ12] László Babai and Youming Qiao. Polynomial-time isomorphism test for groups with abelian sylow towers. In *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 453–464. LIPIcs, 2012.

[CTW13] Arkadev Chattopadhyay, Jacobo Torán, and Fabian Wagner. Graph isomorphism is not AC^0-reducible to group isomorphism. *ACM Transactions on Computation Theory (TOCT)*, 5(4):13, 2013.

[FN14] V Felsch and J Neubüser. On a programme for the determination of the automorphism group of a finite group. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 59–60, 2014.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[GQ14] Joshua A Grochow and Youming Qiao. Algorithms for group isomorphism via group extensions and cohomology. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 110–119. IEEE, 2014.

[GQ15] Joshua A Grochow and Youming Qiao. Polynomial-time isomorphism test of groups that are tame extensions. In *International Symposium on Algorithms and Computation*, pages 578–589. Springer, 2015.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-SAT. In *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pages 237–240. IEEE, 1999.

[Kav07] T. Kavitha. Linear time algorithms for abelian group isomorphism and related problems. *Journal of Computer and System Sciences*, 73(6):986 – 996, 2007.

[KN09] Neeraj Kayal and Timur Nezhmetdinov. Factoring groups efficiently. In *International Colloquium on Automata, Languages, and Programming*, pages 585–596. Springer, 2009.

[LM11] Eugene M Luks and Takunari Miyazaki. Polynomial-time normalizers. *Discrete Mathematics and Theoretical Computer Science*, 13(4):61, 2011.

[Luk82] Eugene M Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of computer and system sciences*, 25(1):42–65, 1982.

[Luk15] Eugene M. Luks. Group isomorphism with fixed subnormal chains. *CoRR*, abs/1511.00151, 2015.

[LW12] Mark L. Lewis and James B. Wilson. Isomorphism in expanding families of indistinguishable groups. *Groups Complexity Cryptology*, 4(1):73–110, 2012.

[QST12] You-Ming Qiao, Jayalal Sarma, and Bang-Sheng Tang. On isomorphism testing of groups with normal hall subgroups. *Journal of Computer Science and Technology*, 27(4):687–701, 2012.

[Ros13] David J. Rosenbaum. Bidirectional collision detection and faster deterministic isomorphism testing. *CoRR*, abs/1304.3935, 2013.

[Ros14] David J. Rosenbaum. Beating the generator-enumeration bound for solvable-group isomorphism. *CoRR*, abs/1412.0639, 2014.

[Rot12] Joseph Rotman. *An introduction to the theory of groups*, volume 148. Springer Science & Business Media, 2012.

[RW15] David J Rosenbaum and Fabian Wagner. Beating the generator-enumeration bound for p-group isomorphism. *Theoretical Computer Science*, 593:16–25, 2015.

[Ser03] Ákos Seress. *Permutation group algorithms*, volume 152. Cambridge University Press, 2003.