Membership problems in finite groups

Markus Lohrey[a], Andreas Rosowski[a], Georg Zetzsche[b]

[a] *University of Siegen, Siegen, Germany*
[b] *Max Planck Institute for Software Systems, Kaiserslautern, Germany*

Abstract

We show that the subset sum problem, the knapsack problem and the rational subset membership problem for permutation groups are **NP**-complete. Concerning the knapsack problem we obtain **NP**-completeness for every fixed $n \geq 3$, where n is the number of permutations in the knapsack equation. In other words: membership in products of three cyclic permutation groups is **NP**-complete. This sharpens a result of Luks [34], which states **NP**-completeness of the membership problem for products of three abelian permutation groups. We also consider the context-free membership problem in permutation groups and prove that it is **PSPACE**-complete but **NP**-complete for a restricted class of context-free grammars where acyclic derivation trees must have constant Horton-Strahler number. Our upper bounds hold for black box groups. The results for context-free membership problems in permutation groups yield new complexity bounds for various intersection non-emptiness problems for DFAs and a single context-free grammar. This paper is an extended version of the conference paper [31].

Keywords: algorithmic group theory, finite groups, membership problems, automata theory

1. Introduction

Membership problems in groups. The general problem that we study in this paper is the following: Fix a class $\mathcal{C}$ of formal languages. We assume that members of $\mathcal{C}$ have a finite description; typical choices are the

Email addresses: `lohrey@eti.uni-siegen.de` (Markus Lohrey), `rosowski@eti.uni-siegen.de` (Andreas Rosowski), `georg@mpi-sws.org` (Georg Zetzsche)

class of regular or context-free languages, or a singleton class $\mathcal{C} = \{L\}$. We are given a language $L \in \mathcal{C}$ with $L \subseteq \Sigma^*$, a group G together with a morphism $h : \Sigma^* \to G$ from the free monoid Σ^* to the group G, and a word $w \in \Sigma^*$. The question that we want to answer is whether $w \in h^{-1}(h(L))$, i.e., whether the group element $h(w)$ belongs to $h(L)$. One can study this problem under several settings, and each of these settings has a different motivation. First of all, one can consider the case, where G is a fixed finitely generated group that is finitely generated by Σ, and the input consists of L. One could call this problem the $\mathcal{C}$-membership problem for the group G. The best studied case is the *rational subset membership problem*, where $\mathcal{C}$ is the class of regular languages. It generalizes the subgroup membership problem for G, a classical decision problem in group theory. Other special cases of the rational subset membership problem that have been studied in the past are the submonoid membership problem, the knapsack problem and the subset problem, see e.g. [29, 35]. It is a simple observation that for the rational subset membership problem one can assume that the word w (that is tested for membership in $h^{-1}(h(L))$ can be assumed to be the empty word, see [24, Theorem 3.1].

In this paper, we study another setting of the above generic problem, where G is a finite group that is part of the input (and L still comes from a languages class $\mathcal{C}$). For the rest of the introduction we restrict to the case, where G is a finite symmetric group S_m (the set of all permutations on $\{1, \ldots, m\}$) that is represented in the input by the integer m in unary representation, i.e., by the word $\$^m$.[1] Our applications only make use of this case, but we remark that our upper complexity bounds can be proven in the more general black box setting [6] (in particular, one could replace symmetric groups by matrix groups over a finite field and still obtain the same complexity bounds). Note that $|S_m| = m!$, hence the order of the group is exponential in the input length.

Membership problems for permutation groups. One of the best studied membership problems for permutation groups is the *subgroup membership problem*: the input is a unary encoded number m and a list of permutations $a, a_1, \ldots, a_n \in S_m$, and it is asked whether a belongs to the sub-

[1] We could also consider the case where G is a subgroup of S_m that is given by a list of generators (i.e., G is a permutation group), but this makes no difference for our problems.

group of S_m generated by $a_1, \ldots, a_n$. The well-known Schreier-Sims algorithm solves this problem in polynomial time [37], and the problem is known to be in **NC** (the class of all problems that can be solved in polylogarithmic time with polynomially many processors) [5].

Several generalizations of the subgroup membership problem have been studied. Luks defined the k-membership problem ($k \geq 1$) as follows: The input is a unary encoded number m, a permutation $a \in S_m$ and a list of k permutation groups $G_1, G_2, \ldots, G_k \leq S_m$ (every G_i is given by a list of generators). The question is whether a belongs to the product $G_1 \cdot G_2 \cdots G_k$. It is a famous open problem whether 2-membership can be solved in polynomial time. This problem is equivalent to several other important algorithmic problems in permutation groups: computing the intersection of permutation groups, computing set stabilizers or centralizers, checking equality of double cosets, see [34] for details. On the other hand, Luks has shown in [34] that m-membership is **NP**-complete for every $k \geq 3$. In fact, he proved **NP**-hardness of membership in a product $G \cdot H \cdot G$, where G and H are both abelian permutation groups.

Note that the k-membership problem is a special case of the rational subset membership for symmetric groups. Let us define this problem again for the setting of symmetric groups (here, 1 denotes the identity permutation and we identify a word over the alphabet S_m with the permutation to which it evaluates):

Problem 1.1 (rational subset membership problem for symmetric groups)**.**

Input: a unary encoded number $m \in \mathbb{N}$ and a nondeterministic finite automaton (NFA) $\mathcal{A}$ over the alphabet S_m.
Question: Does $1 \in L(\mathcal{A})$ hold?

This problem was shown to be **NP**-complete in [26] (the result has been independently shown in the conference version [31] of this paper).

An obvious generalization of the rational subset membership problem for symmetric groups is the *context-free subset membership problem for symmetric groups*; it is obtained by replacing the NFA $\mathcal{A}$ in Problem 1.1 by a context-free grammar $\mathcal{G}$.

Two restrictions of the rational subset membership problem that have been intensively studied for infinite groups in recent years are the *knapsack problem* and *subset sum problem*, see e.g. [4, 7, 8, 16, 17, 27, 30, 32, 35].

For symmetric groups, these problems are defined as follows (note that the number $n+1$ of permutations is part of the input):

Problem 1.2 (subset sum problem for symmetric groups).
Input: a unary encoded number $m \in \mathbb{N}$ and permutations $a, a_1, \dots, a_n \in S_m$.
Question: Are there $i_1, \dots, i_n \in \{0, 1\}$ such that $a = a_1^{i_1} \cdots a_n^{i_n}$?

The subset sum problem is the membership problem for the cubes from [6].

Problem 1.3 (knapsack problem for symmetric groups).
Input: a unary encoded number $m \in \mathbb{N}$ and permutations $a, a_1, \dots, a_n \in S_m$.
Question: Are there $i_1, \dots, i_n \in \mathbb{N}$ such that $a = a_1^{i_1} \cdots a_n^{i_n}$?

We will also consider the following restrictions of these problems.

Problem 1.4 (abelian subset sum problem for symmetric groups).
Input: a unary encoded number $m \in \mathbb{N}$ and pairwise commuting permutations $a, a_1, \dots, a_n \in S_m$.
Question: Are there $i_1, \dots, i_n \in \{0, 1\}$ such that $a = a_1^{i_1} \cdots a_n^{i_n}$?

The following problem is the special case of Luks' k-membership problem for cyclic groups. Note that k is a fixed constant here.

Problem 1.5 (k-knapsack problem for symmetric groups).
Input: a unary encoded number $m \in \mathbb{N}$ and $k+1$ permutations $a, a_1, \dots, a_k \in S_m$.
Question: Are there $i_1, \dots, i_k \in \mathbb{N}$ such that $a = a_1^{i_1} \cdots a_k^{i_k}$?

Main results. Our main result for the rational subset membership problem in symmetric groups is:

Theorem 1.6. *The problems 1.1–1.4 as well as Problem 1.5 for $k \geq 3$ are all* **NP***-complete.*

In contrast, we will show that the 2-knapsack problem can be solved in polynomial time (Theorem 5.8). The **NP** upper bound for the rational subset membership problem will be shown for black-box groups.

We also prove **NP**-completeness of the membership problem in products $\langle g \rangle \langle h_1, h_2, h_3 \rangle \langle g \rangle$, for given permutations g, h_1, h_2, h_3 where h_1, h_2, h_3 pairwise commute; see Theorem 5.9. This sharpens Luks' **NP**-completeness result for products GHG (with G and H abelian permutation groups) in another direction (G can be chosen to be cyclic).

Remark 1.7. The abelian variant of the knapsack problem, i.e., Problem 1.3 with the additional restriction that the permutations $s_1, \dots, s_n$ pairwise commute is of course the abelian subgroup membership problem and hence belongs to **NC**.

Remark 1.8. Analogously to the k-knapsack problem one might consider the k-subset sum problem, where the number n in Problem 1.2 is fixed to k and not part of the input. This problem can be solved in time $2^k \cdot m^{\mathcal{O}(1)}$ (check all 2^k assignments for exponents $i_1, \dots, i_k$) and hence in polynomial time for every fixed k.

Finally, for the context-free subset membership problem for symmetric groups we show:

Theorem 1.9. *The context-free membership problem for symmetric groups is* **PSPACE**-*complete.*

If we restrict the class of context-free grammars in Theorem 1.9 we can improve the complexity to **NP**. For this we need the concept of the *Horton-Strahler number*. The Horton-Strahler number $\mathrm{hs}(t)$ of a full binary tree[2] t (introduced by Horton and Strahler in the context of hydrology [21, 38]; see [14] for a good survey emphasizing the importance of Horton-Strahler numbers in computer science) is recursively defined as follows: If t consists of a single node then $\mathrm{hs}(t) = 0$. Otherwise, assume that t_1 and t_2 are the subtrees rooted in the two children of the node. If $\mathrm{hs}(t_1) = \mathrm{hs}(t_2)$ then $\mathrm{hs}(t) = 1 + \mathrm{hs}(t_1)$, and if $\mathrm{hs}(t_1) \neq \mathrm{hs}(t_2)$ then $\mathrm{hs}(t) = \max\{\mathrm{hs}(t_1), \mathrm{hs}(t_2)\}$.

Consider now a context-free grammar $\mathcal{G}$ in *Chomsky normal form*, i.e., all productions of $\mathcal{G}$ have the form $A \to a$ or $A \to BC$ for nonterminals A, B and a terminal symbol a. A derivation tree of $\mathcal{G}$ is called *acyclic* if no nonterminal appears twice on a path from the root to a leaf. When we refer to the Horton-Strahler number of a derivation tree T of $\mathcal{G}$ we mean the Horton-Strahler number of the tree obtained by removing from T the terminal-labelled leaves so that the tree becomes a full binary tree. The same convention is used for the height of a derviation tree. Note that the height of an acyclic derivation tree is bounded by the number of nonterminals of $\mathcal{G}$ minus 1. For $k \geq 1$ let $\mathrm{CFG}(k)$ be the set of all context-free grammars

[2] A full binary tree t is a tree where every node is either a leaf or has exactly two children.

in Chomsky normal form such that every acyclic derivation tree has Horton-Strahler number at most k.

Theorem 1.10. *For every $k \geq 1$, the context-free membership problem for symmetric groups restricted to context-free grammars from CFG(k) is* **NP**-*complete.*

Note that this result generalizes the statement for the rational subset membership problem in Theorem 1.6 since every regular grammar (when brought into Chomsky normal form) belongs to CFG(1). Also linear context-free grammars belong to CFG(1). We remark that Theorem 1.10 is a promise problem in the sense that **coNP** is the best upper complexity bound for testing whether a given context-free grammar belongs to CFG(k) that we are aware of; see the appendix.

The upper bounds in Theorems 1.6, 1.9, and 1.10 will be actually shown for black box groups.

Application to intersection non-emptiness problems. We can apply Theorems 1.9 and 1.10 to intersection non-emptiness problems. The *intersection non-emptiness problem for deterministic finite automata* (DFAs) is the following problem:

Problem 1.11 (intersection non-emptiness problem for DFAs)**.**
Input: DFAs $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$
Question: Is $\bigcap_{1 \leq i \leq n} L(\mathcal{A}_i)$ non-empty?

Kozen [28] has shown that this problem is **PSPACE**-complete. When restricted to group DFAs (see Section 2) the intersection non-emptiness problem was shown to be **NP**-complete by Blondin et al. [10]. Based on Cook's characterization of **EXPTIME** by polynomially space bounded AuxPDAs [11], Swernofsky and Wehar [39] showed that the intersection non-emptiness problem is **EXPTIME**-complete[3] for a list of general DFAs and a single context-free grammar; see also [19, p. 275] and see [13] for a related **EXPTIME**-complete problem. Using Theorems 1.9 and 1.10 we can easily show the following new results:

[3]The intersection non-emptiness problem becomes undecidable if one allows more than one context-free grammar.

	no CFG	one CFG(k)	one CFG
DFAs	**PSPACE**-c. [28]	**EXPTIME**-c. for k large enough	**EXPTIME**-c. [39]
group DFAs	**NP**-c. [10]	**NP**-c. for all $k \geq 1$	**PSPACE**-c.

Table 1: Complexity of various intersection non-emptiness problems

Theorem 1.12. *The following problem is* **NP***-complete for every* $k \geq 1$*:*

Input: A list of group DFAs $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ *and a context-free grammar* $\mathcal{G} \in$ *CFG*(k).
Question: Is $L(\mathcal{G}) \cap \bigcap_{1 \leq i \leq n} L(\mathcal{A}_i)$ *non-empty?*

Theorem 1.13. *The following problem is* **PSPACE***-complete:*

Input: A list of group DFAs $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ *and a context-free grammar* $\mathcal{G}$.
Question: Is $L(\mathcal{G}) \cap \bigcap_{1 \leq i \leq n} L(\mathcal{A}_i)$ *non-empty?*

Table 1 gives an overview on the complexity of intersection non-emptiness problems. For the intersection non-emptiness problem for arbitrary DFAs and one grammar from CFG(k) one has to notice that in the **EXPTIME**-hardness proof from [39] one can choose a fixed context-free grammar. Moreover, every fixed context-free grammar belongs to CFG(k) for some $k \geq 1$.

Related work. Computational problems for permutation groups have a long history (see e.g. the text book [36]), and have applications, e.g. for graph isomorphism testing [3, 33]. A problem that is similar to subset sum is the *minimum generator sequence problem* (MGS) [15]: The input consists of unary encoded numbers m, ℓ and a list of permutations $a, a_1, \ldots, a_n \in S_m$. The question is, whether a can be written as a product $b_1 b_2 \cdots b_k$ with $k \leq \ell$ and $b_1, \ldots, b_k \in \{a_1, \ldots, a_n\}$. The problem MGS was shown to be **NP**-complete in [15]. For the case, where the number ℓ is given in binary representation, the problem is **PSPACE**-complete [23]. This yields in fact the **PSPACE**-hardness in Theorem 1.9.

Intersection nonemptiness problems for finite automata have been studied intensively in recent years, see e.g. [2, 12]. The papers [9, 22] prove **PSPACE**-hardness of the intersection nonemptiness problem for inverse automata (DFAs, where the transition monoid is an inverse monoid).

Horton-Strahler numbers have been used in the study of context-free languages before, see [14] for further information and references.

2. Preliminaries

Groups. Let G be a finite group and let G^* be the free monoid of all finite words over the alphabet G. There is a canonical morphism $\phi_G : G^* \to G$ that is the identity mapping on $G \subseteq G^*$. Throughout this paper we suppress applications of ϕ_G and identify words over the alphabet G with the corresponding group elements. For a subset $S \subseteq G$ we denote with $\langle S \rangle$ the subgroup generated by S. The following folklore lemma is a straightforward consequence of Lagrange's theorem (if A and B are subgroups of G with $A < B$, then $|B| \geq 2 \cdot |A|$).

Lemma 2.1. *Let G be a finite group and $S \subseteq G$ a generating set for G. Then, there exists a subset $S' \subseteq S$ such that $\langle S' \rangle = G$ and $|S'| \leq \log_2 |G|$.*

Assume that $G = \langle S \rangle$. A *straight-line program* over the generating set S is a sequence of definitions $\mathcal{S} = (x_i := r_i)_{1 \leq i \leq n}$ where the x_i are variables and every right-hand side r_i is either from S or of the form $x_j x_k$ with $1 \leq j, k < i$. Every variable x_i evaluates to a group element $g_i \in G$ in the obvious way: if $r_i \in S$ then $g_i = r_i$ and if $r_i = x_j x_k$ then $g_i = g_j g_k$. We say that $\mathcal{S}$ produces g_n. The size of $\mathcal{S}$ is n. The following result is known as the reachability theorem from [6, Theorem 3.1].

Theorem 2.2 (reachability theorem)**.** *Let G be a finite group, $S \subseteq G$ a generating set for G, and $g \in G$. Then there exists a straight-line program over S of size at most $(1 + \log_2 |G|)^2$ that produces the element g.*

For a set Q let S_Q be the symmetric group on Q, i.e., the set of all permutations on Q with composition of permutations as the group operation. If $Q = \{1, \dots, m\}$ we also write S_m for S_Q. Let $a \in S_Q$ be a permutation and let $q \in Q$. We also write q^a for $a(q)$. We multiply permutations from left to right, i.e., for $a, b \in S_Q$, ab is the permutation with $q^{ab} = (q^a)^b$ for all $q \in Q$. A permutation group is a subgroup of some S_Q.

Quite often, the permutation groups we consider are actually direct products $\prod_{1 \leq i \leq d} S_{m_i}$ for small numbers m_i. Clearly, we have $\prod_{1 \leq i \leq d} S_{m_i} \leq S_m$ for $m = \sum_{1 \leq i \leq d} m_i$ and an embedding of $\prod_{1 \leq i \leq d} S_{m_i}$ into S_m can be computed in polynomial time.

Horton-Strahler number. We defined the Horton-Strahler number $\mathrm{hs}(t)$ of a full binary tree t in the introduction. The following simple fact will be needed. The height of a binary tree is the maximal number of edges on a path from the root to a leaf.

Lemma 2.3. *Let t be a full binary tree of height $d \geq 0$ and let $s = hs(t)$. Then, t has at most $(d+1)^s$ many leaves and therefore at most $2 \cdot (d+1)^s$ many nodes.*

Proof. We prove the statement by induction on the height d. If $d = 0$ then t consists of a single leaf and we have $s = 0$. So, the statement of the lemma holds. Otherwise take a path $v_1, v_2, \ldots, v_k$ in t, where v_1 is the root, v_k is a leaf, and for every $2 \leq i \leq k$, if t_i is the subtree rooted in v_i and t'_i is the subtree rooted in the sibling node of v_i, then $\mathrm{hs}(t_i) \geq \mathrm{hs}(t'_i)$. Note that $k - 1 \leq d$. Let $t_1 = t$. Then we must have $\mathrm{hs}(t'_{i+1}) < \mathrm{hs}(t_i) \leq s$ for every $1 \leq i \leq k-1$. Moreover, every t'_i has height at most $d-1$. Using induction, we can bound the number of leaves in t by

$$1 + \sum_{i=2}^{k} d^{s-1} \leq 1 + d \cdot d^{s-1} \leq 1 + d^s \leq (d+1)^s.$$

This shows the lemma. □

Computational complexity. We assume that the reader has some basic knowledge in complexity theory; see [1] for more details. The following complexity classes are used in the following:

- **NP**: the class of all problems that can be accepted by a nondeterministic Turing machine in polynomial time.
- **PSPACE**: the class of all problems that can be accepted by a nondeterministic Turing machine whose work space is polynomially bounded by the input length. By Savitch's theorem this is the same as the class of all problems that can be accepted by a deterministic Turing machine whose work space is polynomially bounded by the input length.

Formal languages. We assume that the reader is familiar with basic definitions from automata theory; see e.g. [20] for a classical reference. Our definitions of deterministic finite automata (DFA), nondeterministic finite automata (NFA), and context-free grammars are the standard ones.

Consider a DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, where Q is the finite set of states, Σ is the set of terminal symbols, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \to Q$ is the transition mapping and $F \subseteq Q$ is the set of final states. The *transformation monoid* of $\mathcal{A}$ is the submonoid of Q^Q (the set of all mappings on Q and composition of functions as the monoid operation) generated by all mappings $q \mapsto \delta(q, a)$ for $a \in \Sigma$. A *group DFA* is a DFA whose transformation monoid is a group.

Consider now a context-free grammar $\mathcal{G} = (N, \Sigma, P, S)$ (N is the set of nonterminals, Σ is the set of terminal symbols, $S \in N$ is the start nonterminal and P is the set of productions). We always assume *Chomsky normal form*, i.e., all productions have the form $A \to BC$ or $A \to a$ for $A, B, C \in N$ and $a \in \Sigma$. We can also assume that $\mathcal{G}$ is *reduced*, i.e., every nonterminal A appears in a derivation that starts with S and ends with a word $w \in \Sigma^*$ (otherwise A would be useless and therefore can be eliminated). When we speak of a *derivation tree* T of $\mathcal{G}$ we assume that every leaf is labelled with a terminal symbol, whereas the root can be labelled with an arbitrary nonterminal $A \in N$. In other words, derivation trees correspond to derivations that start with a nonterminal A and end with a word $w \in \Sigma^*$. In a *partial derivation tree*, we also allow leafs labelled with nonterminals. When we refer to the Horton-Strahler number of a (partial) derivation tree T, we take the Horton-Strahler number of the tree obtained from T by removing all terminal-labelled leaves. Due to the Chomsky normal form, the resulting tree is a full binary tree. The same convention is used for the height of a (partial) derivation tree.

3. Black box groups

More details on black box groups can be found in [6, 36]. Roughly speaking, in the black box setting group elements are encoded by bit strings of a certain length b and there exist oracles for multiplying two group elements, computing the inverse of a group element, checking whether a given group element is the identity, and checking whether a given bit string of length b is a valid encoding of a group element.[4] As usual, each execution of an oracle

[4] The latter operation is not allowed in [6].

operation counts one time unit, but the parameter b enters the input length additively.

Formally, a *black box* is a tuple

$$B = (b, c, \mathsf{valid}, \mathsf{inv}, \mathsf{prod}, \mathsf{id}, G, f),$$

such that G is a finite group (the group in the box), $b, c \in \mathbb{N}$, and the following properties hold:

- $f : \{0,1\}^b \to G \uplus \{*\}$ is a mapping with

  $$G \subseteq f(\{0,1\}^b)$$

  ($f^{-1}(g) \neq \emptyset$ is the set names of the group element g).

- $\mathsf{valid} : \{0,1\}^b \to \{\mathsf{yes}, \mathsf{no}\}$ is a mapping such that

  $$\forall x \in \{0,1\}^b : f(x) \in G \iff \mathsf{valid}(x) = \mathsf{yes}.$$

- $\mathsf{inv} : \{0,1\}^b \to \{0,1\}^b$ is a mapping such that for all $x \in f^{-1}(G)$:

  $$f(\mathsf{inv}(x)) = f(x)^{-1}.$$

- $\mathsf{prod} : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}^b$ is a mapping such that for all $x, y \in f^{-1}(G)$:

  $$f(\mathsf{prod}(x,y)) = f(x)f(y).$$

- $\mathsf{id} : \{0,1\}^b \times \{0,1\}^c \to \{\mathsf{yes}, \mathsf{no}\}$ is a mapping such that for all $x \in f^{-1}(G)$:

  $$f(x) = 1 \iff \exists y \in \{0,1\}^c : \mathsf{id}(x,y) = \mathsf{yes}$$

  (such a y is called a witness for $f(x) = 1$).

We call b the code length of the black box.

A black box Turing machine is a deterministic or nondeterministic oracle Turing machine $\mathcal{M}$ that has four special oracle query states q_{valid}, q_{inv}, q_{prod}, q_{id}, together with a special oracle tape, on which a binary string is written. The input for $\mathcal{M}$ consists of two unary encoded numbers b and c and some additional problem specific input. In order to determine the behavior of $\mathcal{M}$ on the four special states q_{valid}, q_{inv}, q_{prod}, q_{id}, $\mathcal{M}$ must be coupled with a black box $B = (b, c, \mathsf{valid}, \mathsf{inv}, \mathsf{prod}, \mathsf{id}, G, f)$ (where b and c must match the first part of the input of $\mathcal{M}$). Then $\mathcal{M}$ behaves as follows:

- After entering q_{valid} the oracle tape is overwritten by $\mathsf{valid}(x)$ where $x \in \{0,1\}^b$ is the bit string consisting of the first b bits on the oracle tape.
- After entering q_{inv} the oracle tape is overwritten by $\mathsf{inv}(x)$ where $x \in \{0,1\}^b$ is the bit string consisting of the first b bits on the oracle tape.
- After entering q_{prod} the oracle tape is overwritten by $\mathsf{prod}(x,y)$ where $x, y \in \{0,1\}^b$ and xy is the bit string consisting of the first $2b$ bits on the oracle tape.
- After entering q_{id} the oracle tape is overwritten by $\mathsf{id}(x,y)$ where $x \in \{0,1\}^b$, $y \in \{0,1\}^c$ and xy is the bit string consisting of the first $b+c$ bits on the oracle tape.

As usual with oracle Turing machines, each of these four operations happens instantaneously and counts time $\mathcal{O}(1)$ for the total running time. We denote the machine with the above behaviour by $\mathcal{M}_B$. Note that the black box

$$B = (b, c, \mathsf{valid}, \mathsf{inv}, \mathsf{prod}, \mathsf{id}, G, f)$$

is not part of the input of $\mathcal{M}$, only the unary encoded numbers b and c are part of the input.

Assume that $\mathcal{P}$ is an algorithmic problem for finite groups. The input for $\mathcal{P}$ is a finite group G and some additional data X (e.g. a context-free grammar with terminal alphabet G in the next section). We do not specify exactly, how G is represented. The additional input X may contain elements of G. We will say that $\mathcal{P}$ belongs to **NP** for black box groups if there is a nondeterministic black box Turing machine $\mathcal{M}$, whose input is of the form b, c, X with unary encoded numbers b and c, such that for every black box $B = (b, c, \mathsf{valid}, \mathsf{inv}, \mathsf{prod}, \mathsf{id}, G, f)$ the following holds: $\mathcal{M}_B$ accepts the input b, c, X (where X denotes the additional input for $\mathcal{P}$ and group elements in X are represented by bit strings from $f^{-1}(G)$) if and only if (G, X) belongs to $\mathcal{P}$. The running time of $\mathcal{M}_B$ is polynomial in $b + c + |X|$. Note that since G may have order 2^b, the order of G may be exponential in the input length. We will use the analogous definition for other complexity classes, in particular for **PSPACE**.

For the rest of the paper we prefer a slightly more casual handling of black box groups. We always identify bits strings from $x \in f^{-1}(G)$ with the corresponding group elements. So, we will never talk about bit strings

$x \in f^{-1}(G)$, but instead directly deal with elements of G. The reader should notice that we cannot directly verify whether a given element $g \in G$ is the identity. This is only possible in a nondeterministic way by guessing a witness $y \in \{0,1\}^c$. The same applies to the verification of an identity $g = h$, which is equivalent to $gh^{-1} = 1$. This allows to cover also quotient groups by the black box setting; see [6].

We need the following well-known fact from [6]:

Lemma 3.1. *The subgroup membership problem for black box groups (given group elements $g, g_1, \ldots, g_n$, does $g \in \langle g_1, \ldots, g_n \rangle$ hold?) belongs to* **NP**.

This is a consequence of the reachability theorem: Let b be the code length of the black box. Hence there are at most 2^b group elements. By the reachability theorem (Theorem 2.2) it suffices to guess a straight-line program over $\{g_1, \ldots, g_n\}$ of size at most $(1 + \log_2 2^b)^2 = (b+1)^2$, evaluate it using the oracle for prod (let g' be the result of the evaluation) and check whether $g'g^{-1} = 1$. The later can be done nondeterministically using the oracle for id.

4. Context-free membership in black box groups

The goal of this section is to prove the following two results. Recall the definition of the class CFG(k) from the introduction.

Theorem 4.1. *The context-free subset membership problem for black box groups is in* **PSPACE**.

Theorem 4.2. *For every $k \geq 1$, the context-free membership problem for black box groups restricted to context-free grammars from CFG(k) is in* **NP**.

Before we prove these results, let us derive some corollaries. Theorem 1.10 is a direct corollary of Theorem 4.2. Restricted to regular grammars (which are in CFG(1) after bringing them to Chomsky normal form) we get:

Corollary 4.3. *The rational subset membership problem for black box groups is in* **NP**. *In particular, the rational subset membership problem for symmetric groups is in* **NP**.

Also Theorem 1.9 can be easily obtained now: The upper bound follows directly from Theorem 4.1. The lower bound can be obtained from a result of Jerrum [23]. In the introduction we mentioned that Jerrum proved the **PSPACE**-completeness of the MGS problem for the case where the number ℓ is give in binary notation. Given permutations $a_1, \dots, a_n \in S_m$ and a binary encoded number ℓ one can easily construct a context-free grammar for $\{1, a_1, \dots, a_n\}^\ell \subseteq S_m$. Hence, the MGS problem with ℓ given in binary notation reduces to the context-free membership problem for symmetric groups, showing that the latter is **PSPACE**-hard.

In the rest of the section we prove Theorems 4.1 and 4.2. We fix a finite group G that is only accessed via a black box.

The spanning tree technique. We start with subgroups of G that are defined by finite nondeterministic automata (later, we will apply the following construction to a different group that is also given via a black box). Assume that $\mathcal{A} = (Q, G, \{q_0\}, \delta, \{q_0\})$ is a finite nondeterministic automaton with terminal alphabet G. Note that q_0 is the unique initial and the unique final state. This ensures that the language $L(\mathcal{A})$ defined by $\mathcal{A}$ (which, by our convention, is identified with a subset of the group G) is a subgroup of G: the set $L(\mathcal{A})$ is clearly a submonoid and every submonoid of a finite group is a subgroup. We now show a classical technique for finding a generating set for $L(\mathcal{A})$.

In a first step we remove from $\mathcal{A}$ all states $p \in Q$ such that there is no path from q_0 to p as well as all states p such that there is no path from p to q_0. Let $\mathcal{A}_1$ be the resulting NFA. We have $L(\mathcal{A}) = L(\mathcal{A}_1)$.

In the second step we add for every transition (p, g, q) the inverse transition (q, g^{-1}, p) (unless it already exists). Let $\mathcal{A}_2$ be the resulting NFA. We claim that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$. Note that by the first step, there must be a path from q to p in $\mathcal{A}_1$. Let $h \in G$ be the group element produced by this path. Take a $k > 0$ such that $(gh)^k = 1$ in G. Hence, $g^{-1} = h(gh)^{k-1}$. Moreover, there is a path in $\mathcal{A}_1$ from q to p which produces the group element $h(gh)^{-1} = g^{-1}$. This shows that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$.

In the third step we compute the generating set for $L(\mathcal{A}_2) = L(\mathcal{A})$ using the spanning tree technique (see [25] for an application in the context of free groups). Consider the automaton $\mathcal{A}_2$ as an undirected multi-graph $\mathcal{G}$. The nodes of $\mathcal{G}$ are the states of $\mathcal{A}_2$. Moreover, every undirected pair $\{(p, g, q), (q, g^{-1}, p)\}$ of transitions in the NFA $\mathcal{A}_2$ is an undirected edge in $\mathcal{G}$ connecting the nodes p and q. Note that there can be several edges between two nodes (as well as loops); hence $\mathcal{G}$ is indeed a multi-graph. We then compute a spanning tree $\mathcal{T}$ of $\mathcal{G}$. For every state of p of $\mathcal{A}_2$ we fix a directed simple path π_p in $\mathcal{T}$ from q_0 to p; its length is bounded by the number of states of $\mathcal{A}_2$ minus 1. We can view this path π_p as a path in $\mathcal{A}_2$. Let g_p be the group element produced by the path π_p. For every undirected edge $e = \{(p,g,q),(q,g^{-1},p)\}$ in $\mathcal{G} \setminus \mathcal{T}$ let $g_e := g_p g g_q^{-1}$ (we could also take $g_q g^{-1} g_p^{-1}$). A standard argument shows that the set $\{g_e \mid e \text{ is an edge in } \mathcal{G} \setminus \mathcal{T}\}$ indeed generates $L(\mathcal{A})$.

The above construction can be carried out in polynomial time for black box groups. This is straightforward. The only detail that we want to emphasize is that in the black box setting we have to allow multiple copies of undirected edges $\{(p,g,q),(q,g^{-1},p)\}$. The reason is that we may have several names (bit strings) denoting the same group element and we can only verify nondeterministically whether two bit strings represent the same group element. But this is not a problem; it just implies that we may output copies of the same generator.

The operations Δ and Γ. Let $\mathcal{G} = (N, G, P, S)$ be a context-free grammar in Chomsky normal form that is part of the input, whose terminal alphabet is the finite group G. When we speak of the input size in the following, we refer to $|\mathcal{G}| + b + c$, where b and c are the two unary encoded numbers from the black box for G and the size $|\mathcal{G}|$ is defined as the number of productions of the grammar. Recall from the introduction that a derivation tree is acyclic if in every path from the root to a leaf every nonterminal appears at most once. The height of an acyclic derivation tree is bounded by $|N| - 1$.

Also recall that $\phi_G : G^* \to G$ is the canonical morphism from Section 2. With $L(A)$ we denote the set of all words $w \in G^*$ that are derived from the nonterminal $A \in N$ and, as usual, we identify $L(A)$ with $\phi_G(L(A)) \subseteq G$. Let $\hat{G}$ be the dual group of G: it has the same underlying set as G and if $g \cdot h$ denotes the product in G then the multiplication $\circ$ in $\hat{G}$ is defined by $g \circ h = h \cdot g$. The direct product $G \times \hat{G}$ will be important for the following construction. Note that it is straightforward to define a black box for $G \times \hat{G}$ from a black box for G. For every nonterminal $A \in N$ we define the subgroup $G_A \leq G \times \hat{G}$ by

$$G_A = \{(\phi_G(u), \phi_G(v)) \mid u, v \in G^*, A \Rightarrow^*_{\mathcal{G}} uAv\}. \qquad (1)$$

Note that G_A is indeed a group. To see this, it suffices to argue that $H_A \subseteq G_A \leq G \times \hat{G}$ is a monoid (every submonoid a finite group is a subgroup).

The latter follows from the fact that two derivations $A \Rightarrow_{\mathcal{G}}^{*} u_1 A v_1$ and $A \Rightarrow_{\mathcal{G}}^{*} u_2 A v_2$ can be composed to the derivation $A \Rightarrow_{\mathcal{G}}^{*} u_1 u_2 A v_2 v_1$ and in $G_A \leq G \times \hat{G}$ we have $(\phi_G(u_1), \phi_G(v_1))(\phi_G(u_2), \phi_G(v_2)) = (\phi_G(u_1)\phi_G(u_2), \phi_G(v_2)\phi_G(v_1))$.

We now define two important operations Δ and Γ. The operation Δ maps a tuple $s = (H_A)_{A \in N}$ of subgroups $H_A \leq G \times \hat{G}$ to a tuple $\Delta(s) = (L_A)_{A \in N}$ of subsets $L_A \subseteq G$ (not necessarily subgroups), whereas Γ maps a tuple $t = (L_A)_{A \in N}$ of subsets $L_A \subseteq G$ to a tuple $\Gamma(t) = (H_A)_{A \in N}$ of subgroups $H_A \leq G \times \hat{G}$.

We start with Δ. Let $s = (H_A)_{A \in N}$ be a tuple of subgroups $H_A \leq G \times \hat{G}$. The tuple $\Delta(s) = (L_A)_{A \in N}$ of subsets $L_A \subseteq G$ is obtained as follows: Let T be an acyclic derivation tree with root r labelled by $A \in N$. We assign inductively a set $L_v \subseteq G$ to every inner node v: Let B the label of v. If v has only one child it must be a leaf since our grammar is in Chomsky normal form. Let $g \in G$ be the label of this leaf. Then we set $L_v = \{h_1 g h_2 \mid (h_1, h_2) \in H_B\}$. If v has two children v_1, v_2 (where v_1 is the left child and v_2 the right child), then the sets $L_{v_1} \subseteq G$ and $L_{v_2} \subseteq G$ are already determined and we set

$$L_v = \{h_1 g_1 g_2 h_2 \mid (h_1, h_2) \in H_B, g_1 \in L_{v_1}, g_2 \in L_{v_2}\}.$$

We set $L(T) = L_r$ and finally define L_A as the union of all sets $L(T)$ where T is an acyclic derivation tree whose root is labelled with A.

The second operation Γ is defined as follows: Let $t = (L_A)_{A \in N}$ be a tuple of subsets $L_A \subseteq G$. Then we define the tuple $\Gamma(t) = (H_A)_{A \in N}$ with $H_A \leq G \times \hat{G}$ as follows: Fix a nonterminal $A \in N$. Consider a sequence $p = (A_i \to A_{i,0} A_{i,1})_{1 \leq i \leq m}$ of productions $(A_i \to A_{i,0} A_{i,1}) \in P$ and a sequence $d = (d_i)_{1 \leq i \leq m}$ of directions $d_i \in \{0, 1\}$ such that $A_{i+1} = A_{i,d_i}$ for all $1 \leq i \leq m$, $A_1 = A = A_{m,d_m}$. Basically, p and d define a path from A back to A. For every $1 \leq i \leq m$ we define the sets

$$M_i = \begin{cases} L_{A_{i,0}} \times \{1\} \text{ if } d_i = 1 \\ \{1\} \times L_{A_{i,1}} \text{ if } d_i = 0 \end{cases}$$

We view M_i as a subset of $G \times \hat{G}$ and define

$$M(p, d) = \prod_{1 \leq i \leq m} M_i,$$

where $\prod$ refers to the product in $G \times \hat{G}$. If p and d are the empty sequences ($m = 0$) then $M(p, d) = \{(1, 1)\}$. Finally we define H_A as the union of all

$M(p,d)$, where $p = (A_i \to A_{i,0}A_{i,1})_{1\le i\le m}$ and $d = (d_i)_{1\le i\le m}$ are as above (including the empty sequences). This set H_A is a subgroup of $G \times \hat{G}$ (for the same reason that G_A in (1) is a subgroup of $G \times \hat{G}$).

One should see Δ and Γ as saturation operations that when applied alternatingly finally yield the sets $L(A) \subseteq G$ and the subgroups $G_A \le G \times \hat{G}$. This intuition is captured by the following two lemmas.

Lemma 4.4. $\Delta((G_A)_{A\in N}) = (L(A))_{A\in N}$.

Proof. To see this, let $\Delta((G_A)_{A\in N}) = (L_A)_{A\in N}$. The inclusion $L_A \subseteq L(A)$ is clear: the definition of Δ and G_A directly yields a derivation tree with root labelled by A for every element in L_A. For the inclusion $L(A) \subseteq L_A$ take an arbitrary derivation tree T for an element $w \in L(A)$ with root labelled by A. We can get an acyclic derivation tree from T by contracting paths from a B-labelled node down to another B-labelled node in T for an arbitrary $B \in N$. If we choose these paths maximal, then they will not overlap, which means that we can contract all chosen paths in parallel and thereby obtain an acyclic derivation tree. Each path produces a pair from G_B for some $B \in N$. This shows that $w \in L_A$ and proves the lemma. ☐

Let $s_0 = (H_A)_{A\in N}$ with $H_A = \{(1,1)\}$ for all $A \in N$ be the tuple of trivial subgroups of $G \times \hat{G}$. For two tuples $s_1 = (H_{A,1})_{A\in N}$ and $s_2 = (H_{A,2})_{A\in N}$ of subgroups of $G \times \hat{G}$ we write $s_1 \le s_2$ if $H_{A,1} \le H_{A,2}$ for every $A \in N$. By induction over $i \ge 0$ we show that $(\Gamma\Delta)^i(s_0) \le (\Gamma\Delta)^{i+1}(s_0)$ for all i: For $i = 0$ this is clear and the induction step holds since Γ as well as Δ are monotone with respect to componentwise inclusion. Hence, we can define $\lim_{i\to\infty}(\Gamma\Delta)^i(s_0)$.

Lemma 4.5. $(G_A)_{A\in N} = \lim\limits_{i\to\infty}(\Gamma\Delta)^i(s_0) = (\Gamma\Delta)^j(s_0)$ *for* $j = 2|N|\cdot\lfloor\log_2|G|\rfloor$.

Proof. From the definition of Γ and Δ we directly get $(\Gamma\Delta)^i(s_0) \le (G_A)_{A\in N}$ for every $i \ge 0$. Let us next show that $(G_A)_{A\in N} \le \lim_{i\to\infty}(\Gamma\Delta)^i(s_0)$. Let $\lim_{i\to\infty}(\Gamma\Delta)^i(s_0) = (H_A)_{A\in N}$ and $(g,h) \in G_A$. Hence, there exists a derivation $A \Rightarrow^*_{\mathcal{G}} uAv$ such that $g = \phi_G(u)$ and $h = \phi_G(v)$. We prove $(g,h) \in H_A$ by induction on the length of this derivation. Let T be the partial derivation tree corresponding to the derivation $A \Rightarrow^*_{\mathcal{G}} uAv$. From the derivation $A \Rightarrow^*_{\mathcal{G}} uAv$ we obtain a sequence $p = (A_i \to A_{i,0}A_{i,1})_{1\le i\le m}$ of productions $(A_i \to A_{i,0}A_{i,1}) \in P$ and a sequence $d = (d_i)_{1\le i\le m}$ of directions $d_i \in \{0,1\}$ such that $A_{i+1} = A_{i,d_i}$ for all $1 \le i \le m$, $A_1 = A = A_{m,d_m}$. Assume that

$A_{i,1-d_i}$ derives to $w_i \in G^*$ in the derivation $A \Rightarrow^*_{\mathcal{G}} uAv$ for all $1 \le i \le m$ and define

$$(u_i, v_i) = \begin{cases} (w_i, 1) & \text{if } d_i = 1 \\ (1, w_i) & \text{if } d_i = 0. \end{cases}$$

Then we obtain

$$(g, h) = \prod_{1 \le i \le m} (\phi_G(u_i), \phi_G(v_i))$$

where the product is computed in the group $G \times \hat{G}$. Let T_i be the subtree of T that corresponds to the derivation $A_{i,1-d_i} \Rightarrow^*_{\mathcal{G}} w_i$. We now apply the same argument that we used for the proof of Lemma 4.4 to each of the trees T_i, i.e., we contract maximal subpaths from a B-labelled node down to a B-labelled node (for $B \in N$ arbitrary). Each of these subpaths corresponds to a derivation $B \Rightarrow^*_{\mathcal{G}} u'Bv'$ that is of course shorter than the derivation $A \Rightarrow^*_{\mathcal{G}} uAv$. By induction, we get $(\phi_G(u'), \phi_G(v')) \in G_B$. Moreover, from the construction, it follows that (i) $\phi_G(w_i)$ belongs to the $A_{i,1-d_i}$-component of $\Delta((H_B)_{B \in N})$ and (ii) (g, h) belongs to the A-component of $\Gamma(\Delta((H_B)_{B \in N}))$, which is H_A. The construction is shown in Figure 1. All the paths between identical nonterminals in the subtrees below $A'_{1,1}, \ldots, A'_{6,0}$ are contracted and replaced by their "effects", which by induction are already in the corresponding groups G_X ($X \in \{B, C, \ldots, G\}$). After these contractions the subtrees are acyclic. This concludes the proof of the first equality in the lemma.

Since all G_A are finite groups there is a smallest number $j \ge 0$ such that

$$(\Gamma\Delta)^j(s_0) = (\Gamma\Delta)^{j+1}(s_0).$$

We then have $(\Gamma\Delta)^j(s_0) = \lim_{i\to\infty}(\Gamma\Delta)^i(s_0)$. It remains to show that $j \le 2|N| \cdot \log_2 |G|$. In each component of the $|N|$-tuples $(\Gamma\Delta)^i(s_0)$ $(0 \le i \le j)$ we have a chain of subgroups of $G \times \hat{G}$. By Lagrange's theorem, any chain $\{(1, 1)\} = H_0 < H_1 < \cdots < H_{k-1} < H_k \le G \times \hat{G}$ satisfies $k \le 2 \cdot \log_2 |G|$. This shows that $j \le 2|N| \cdot \log_2 |G|$ □

One could use Lemma 4.4 and Lemma 4.5 to compute the sets $L(A)$ and the groups G_A by applying the operations Δ and Γ alternatingly, starting with the tuple of trivial subgroups of $G \times \hat{G}$. This is clearly not a **PSPACE** machine, simply because the $L(A)$ and G_A can be of size exponential in the input length and therefore cannot be stored in polynomial space. On the other hand, all we have to do is to check whether 1 belongs to $L(S)$ (for the start nonterminal S) and for this we can use nondeterminism.

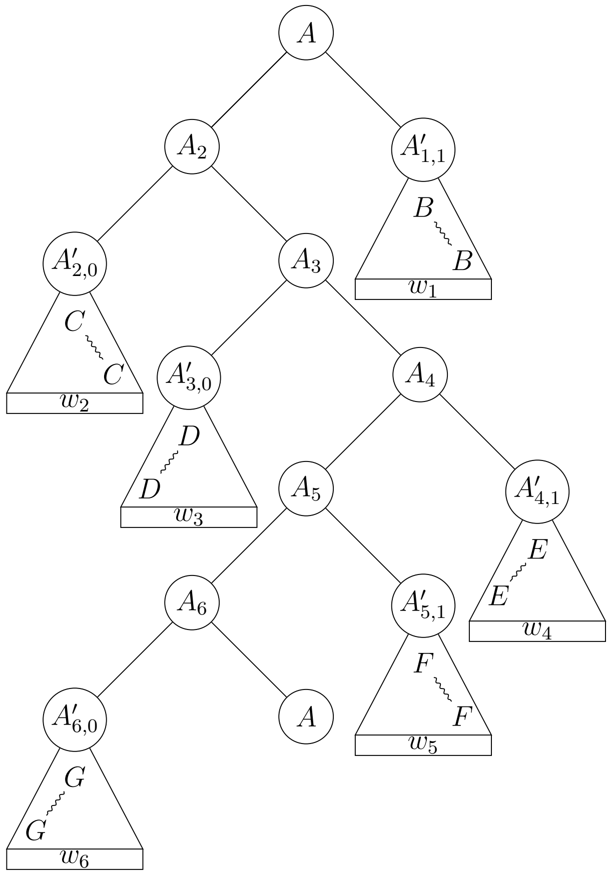

Figure 1: The situation in the proof of $(G_A)_{A\in N} \leq \lim_{i\to\infty}(\Gamma\Delta)^i(s_0)$.

In the following, we will speak of **NP** machines with oracles. Here, we mean nondeterministic polynomial-time Turing machines $\mathcal{M}$ with oracles, where the oracles are arbitrary languages $L_1, \ldots, L_m$ (that are fixed in the beginning). In the standard setting, the machine $\mathcal{M}$ has for every $1 \leq i \leq m$ a special instruction that allows to test in a single step whether the word w that is currently written on a distinguished oracle tape belongs to the set L_i. In the following, we make the additional restriction that the oracle languages L_i can only be queried positively: If $w \in L_i$ then the instruction succeeds and returns the answer "yes" (and the computation of $\mathcal{M}$ continues). If $w \notin L_i$ then the instruction does not succeed and the computation of $\mathcal{M}$ stops in a rejecting state. As for ordinary nondeterministic machines (without oracles), the machine $\mathcal{M}$ accepts its input word if and only if there is at least one computation ending in the accepting state. This implies that if all L_i belong to **NP** (resp. **PSPACE**), then the language accepted by $\mathcal{M}$ also belongs to **NP** (resp. **PSPACE**). Likewise, we will use the notion of a **PSPACE** machine with oracles.

Lemma 4.6. *For a tuple $(L_A)_{A \in N}$ of subsets $L_A \subseteq G$ there is an* **NP** *machine with the oracle sets L_A ($A \in N$) such that the machine tests membership in a specified entry of the tuple $\Gamma((L_A)_{A \in N})$.*

Proof. Let $\Gamma((L_A)_{A\in N}) = (H_A)_{A\in N}$. For every nonterminal $A \in N$ we define the NFA

$$\mathcal{A}_A = (N, (G \times \hat{G}), \{A\}, \delta, \{A\}),$$

whose input alphabet is the finite group $G \times \hat{G}$. The NFAs $\mathcal{A}_A$ only differ in the initial and final state. The transition relation δ contains all triples $(B, (g, h), C) \in N \times (G \times \hat{G}) \times N$ such that for some $D \in N$ either $(B \to CD) \in P$, $g = 1$, and $h \in L_D$ or $(B \to DC) \in P$, $h = 1$, and $g \in L_D$. Then we have $L(\mathcal{A}_A) = H_A$. As in the spanning tree approach we add for every transition $(B, (g, h), C)$ in the NFA $\mathcal{A}_A$ also the inverse transition $(C, (g^{-1}, h^{-1}), B)$. In the following, $\mathcal{A}_A$ refers to this NFA. The number of transitions of the NFA $\mathcal{A}_A$ can be exponential in the input size, so we cannot afford to construct $\mathcal{A}_A$ explicitly. But this is not necessary, since we only aim to come up with a nondeterministic polynomial time machine.

Recall the spanning tree technique, which yields a generating set for the subgroup $L(\mathcal{A}_A) = H_A$. This generating set will be in general of exponential size. On the other hand, Lemma 2.1 guarantees that the generating set produced by the spanning tree approach contains a subset of size at most $\log_2 |G \times \hat{G}| = 2 \cdot \log_2 |G|$ that still generates $L(\mathcal{A}_A)$. Note that $2 \cdot \log_2 |G|$ is linearly bounded in the input size. We can therefore nondeterministically produce a set of at most $2 \cdot \log_2 |G|$ loops in the NFA $\mathcal{A}_A$ (starting and ending in A) of length at most $2|N| - 1$. Note that $\mathcal{A}_A$ has $|N|$ states and that the loops produced by the spanning tree approach have length at most $2|N| - 1$. We do not even have to produce a spanning tree before: every generator produced by the spanning tree approach is a loop in $\mathcal{A}_A$ (starting and ending in A) of length at most $2|N| - 1$ and every such loop certainly yields an element of H_A. For every transition that appears on one of the guessed loops we guess a transition label (either a pair $(1, h)$ or a pair $(g, 1)$) and verify, using the oracle for membership to L_B, that we guessed a transition in the NFA $\mathcal{A}_A$. Then we multiply the guessed pairs along the loop (starting and ending in A) in the group $G \times \hat{G}$. Let us denote with $S_A \subseteq G \times \hat{G}$ the resulting subset of $G \times \hat{G}$ (of course, it depends on the nondeterministic choices). For every nondeterministic choice we have $S_A \subseteq H_A$ and there exists at least one nondeterministic choice for which $\langle S_A \rangle = H_A$. By Lemma 3.1 we can finally check in **NP** whether a given pair (g, h) belongs to $\langle S_A \rangle$. □

Lemma 4.7. *Assume that the input grammar $\mathcal{G}$ is restricted to the class CFG(k) for some fixed k. For a tuple $(H_A)_{A \in N}$ of subgroups $H_A \leq G \times \hat{G}$ there is an* **NP** *machine with the oracle sets H_A ($A \in N$) such that the machine tests membership in a specified entry of the tuple $\Delta((H_A)_{A \in N})$.*

Proof. By assumption, the Horton-Strahler number of every acyclic derivation tree of $\mathcal{G}$ is bounded by the constant k. Since the height of an acyclic derivation tree is bounded by $|N|$ the total number of nodes in the tree is bounded by $2(|N|+1)^k$ by Lemma 2.3. Let $\Delta((H_A)_{A \in N}) = (L_A)_{A \in N}$. Fix an $A \in N$ and a group element $g \in G$. We want to verify whether $g \in L_A$. For this we guess an acyclic derivation tree T with root A. This can be done by a nondeterministic polynomial time machine. Moreover we guess for every inner node v of T that is labelled with the nonterminal B a pair $(h_{v,1}, h_{v,2}) \in G \times \hat{G}$ and verify using the oracle for membership to H_A that $(h_{v,1}, h_{v,2}) \in H_A$. If the verification is successful, we evaluate every inner node v to a group element $g_v \in G$. If v has a single child, it must be labelled with a group element $h \in G$ (due to a production $B \to h$) and we set $g_v = h_{v,1} h h_{v,2}$. If v has two children v_1 (the left child) and v_2 (the right child) then we set $g_v = h_{v,1} g_{v_1} g_{v_2} h_{v,2}$. At the end, we check whether $g = g_r$, where r is the root of the tree T. □

Lemma 4.8. *For a tuple $(H_A)_{A \in N}$ of subgroups $H_A \leq G \times \hat{G}$ there is a* **PSPACE** *machine with the oracle sets H_A ($A \in N$) such that the machine tests membership in a specified entry of the tuple $\Delta((H_A)_{A \in N})$.*

Proof. The proof is similar to Lemma 4.7. However, without the restriction that the input grammar belongs to CFG(k) for a fixed constant k, an acyclic derivation tree of the grammar $\mathcal{G}$ may be of size exponential in the input length. But we will see that we never have to store the whole tree but only a polynomial sized part of the tree. To check $g \in L_A$ we do the following: We guess a production for A and a pair $(h_1, h_2) \in G \times \hat{G}$ and verify using our oracle that $(h_1, h_2) \in H_A$. If the guessed production for A is of the form $A \to h$ for a group element $h \in G$ then we only have to check $h_1 h h_2 = g$

and we are done. If the production is of the form $A \to BC$ for nonterminals $B, C \in N$ then we guess additional group elements $g_1, g_2 \in G$ and check that $g = h_1 g_1 g_2 h_2$. If this holds, we continue with two recursive calls for $g_1 \in L_B$ and $g_2 \in L_C$. We have to make sure that this eventually terminates. In order to ensure termination for every computation path we store the nonterminals that we already have seen. By this the recursion depth is bounded by $|N|$. This also ensures that we traverse an acyclic derivation tree. We obtain a nondeterministic polynomial space machine since the recursion depth is bounded by $|N|$ and the space used for the first recursive call can be reused for the second one. □

Lemma 4.9. *For a tuple $(H_A)_{A \in N}$ of subgroups $H_A \leq G \times \hat{G}$ there is an* **NP** *machine $\mathcal{M}$ with the oracle sets H_A ($A \in N$) such that $\mathcal{M}$ has the following properties:*

- *On every accepting computation path, $\mathcal{M}$ outputs a tuple $(S_A)_{A \in N}$ of subsets $S_A \subseteq H_A$.*
- *There is at least one accepting computation path on which $\mathcal{M}$ outputs a tuple $(S_A)_{A \in N}$ such that every S_A generates H_A.*

Proof. By Lemma 2.1 we know that every subgroup $H_A \leq G \times \hat{G}$ is generated by a set of at most $\log_2 |G \times \hat{G}| = 2 \cdot \log_2 |G|$ generators. The machine $\mathcal{M}$ simply guesses for every $A \in N$ a subset $R_A \subseteq G \times \hat{G}$ of size at most $2 \cdot \log_2 |G|$. Then it verifies, using the oracles, for every $A \in N$ and every $(g, h) \in R_A$, whether $(g, h) \in H_A$ holds. If all these verification steps succeed, $\mathcal{M}$ outputs the set R_A for every $A \in N$. □

If membership for H_A is in **PSPACE** for every $A \in N$, then we could actually compute deterministically in polynomial space a generating set for every H_A by iterating over all elements of $G \times \hat{G}$. But we will not need this stronger fact.

We are now in the position to prove Theorems 4.1 and 4.2.

Proofs of Theorems 4.1 and 4.2. We start with the proof of Theorem 4.2. By Lemma 4.4 and Lemma 4.7 it suffices to show that membership for the subgroups G_A is in **NP**. For this, we construct a nondeterministic polynomial time machine that computes on every computation path a subset $S_A \subseteq G_A$ for every $A \in N$ such that on at least one computation path it computes a generating set for groups G_A for all $A \in N$. Then we can decide membership for the $\langle S_A \rangle$ in **NP** by Lemma 3.1.

The set S_A is computed by initializing $S_A = \{(1,1)\}$ for every $A \in N$ and then doing $2|N| \cdot \log_2 |G|$ iterations of the following procedure: Assume that we have already produced the subsets $(S_A)_{A \in N}$. Membership in $\langle S_A \rangle$ can be decided in **NP** by Lemma 3.1. Hence, by Lemmas 4.6 and 4.7 one can decide membership in every entry of the tuple $\Gamma(\Delta((\langle S_A \rangle)_{A \in N}))$ in **NP**. Finally, by Lemma 4.9 we can produce nondeterministically in polynomial time a subset $S'_A \subseteq G \times \hat{G}$ for every $A \in N$ such that for every computation path we have $(\langle S'_A \rangle)_{A \in N} \leq \Gamma(\Delta((\langle S_A \rangle)_{A \in N}))$ and for at least one computation path the machine produces subsets S'_A with $(\langle S'_A \rangle)_{A \in N} = \Gamma(\Delta((\langle S_A \rangle)_{A \in N}))$. With the sets S'_A we go into the next iteration. By Lemma 4.5 there will be at least one computation path on which after $2|N| \cdot \log_2 |G|$ iterations we get generating sets for all the groups G_A. This concludes the proof of Theorem 4.2.

The proof of Theorem 4.1 is identical except that Lemma 4.8 instead of Lemma 4.7 is used. □

5. Restrictions of rational subset membership in symmetric groups

In this section, we want to contrast the general upper bounds from the previous sections with lower bounds for symmetric groups and restricted versions of the rational subset membership problem. We start with the subset sum problem.

5.1. Subset sum in permutation groups

The following result refers to the abelian group $\mathbb{Z}_3^m$, for which we use the additive notation.

Theorem 5.1. *The following problem is* **NP***-hard:*
Input: unary encoded number m and a list of group elements $g, g_1, \ldots, g_n \in \mathbb{Z}_3^m$.
Question: Are there $i_1, \ldots, i_n \in \{0,1\}$ such that $g = \sum_{1 \leq k \leq n} i_k \cdot g_k$?

Proof. We prove the theorem by a reduction from the problem *exact 3-hitting set problem* (X3HS):

Problem 5.2 (X3HS)**.**
Input: a finite set A and a set $\mathcal{B} \subseteq 2^A$ of subsets of A, all of size 3.
Question: Is there a subset $A' \subseteq A$ such that $|A' \cap C| = 1$ for all $C \in \mathcal{B}$?

X3HS is the same problem as positive 1-in-3-SAT, which is **NP**-complete [18, Problem LO4].

Let A be a finite set and $\mathcal{B} \subseteq 2^A$ be a set of subsets of A, all of size 3. W.l.o.g. assume that $A = \{1, \ldots, n\}$ and let $\mathcal{B} = \{C_1, C_2, \ldots, C_d\}$. We work in the group $\mathbb{Z}_3^d$. For every $1 \leq i \leq n$ let

$$X_i = (a_{i,1}, a_{i,2}, \ldots, a_{i,d}) \in \mathbb{Z}_3^d,$$

where

$$a_{i,j} = \begin{cases} 0 \text{ if } i \notin C_j \\ 1 \text{ if } i \in C_j. \end{cases}$$

Then there exists $A' \subseteq A$ such that $|A' \cap C_j| = 1$ for every $1 \leq j \leq d$ if and only if the following equation has a solution $y_1, \ldots, y_n \in \{0, 1\}$:

$$\sum_{i=1}^{n} y_i \cdot X_i = (1, 1, \ldots, 1).$$

This proves the theorem. □

Clearly $\mathbb{Z}_3^d \leq S_{3d}$. We obtain the following corollary:

Corollary 5.3. *The abelian subset sum problem for symmetric groups is* **NP***-hard.*

Let us remark that the subset sum problem for $\mathbb{Z}_2^d$ (with d part of the input) is equivalent to the subgroup membership problem for $\mathbb{Z}_2^d$ (since every element of $\mathbb{Z}_2^d$ has order two) and therefore can be solved in polynomial time.

5.2. Knapsack in permutation groups

We now come to the knapsack problem in permutation groups. **NP**-hardness of the general version of knapsack can be easily deduced from a result of Luks:

Theorem 5.4 ([34])**.** *The knapsack problem for symmetric groups is* **NP***-hard.*

Proof. Recall from the introduction that Luks [34] proved **NP**-completeness of 3-membership for the special case of membership in a product GHG where G and H are abelian subgroups of S_m.

Let us now assume that $G, H \leq S_m$ are abelian. Let $g_1, g_2, \ldots, g_k$ be the given generators of G and let $h_1, h_2, \ldots, h_l$ be the given generators of H. Then $s \in GHG$ is equivalent to the solvability of the equation

$$s = g_1^{x_1} g_2^{x_2} \cdots g_k^{x_k} h_1^{y_1} h_2^{y_2} \cdots h_l^{y_l} g_1^{z_1} g_2^{z_2} \cdots g_k^{z_k}$$

This is an instance of the knapsack problem, which is therefore **NP**-hard. □

We next want to prove that already 3-knapsack is **NP**-hard. In other words: the k-membership problem is **NP**-hard for every $k \geq 3$ even if the groups are cyclic. We prove this by a reduction from X3HS; see Problem 5.2. For this, we need two lemmas.

Let $a < b$ be integers. For the rest of the paper we write $[a, b]$ for the cycle permutation $(a, a+1, \ldots, b)$ mapping i to $i+1$ for $a \leq i < b$ and mapping b to a. In this subsection, only cycles of the form $[1, a]$ appear, and we use the shorthand notation $[a]$ for $[1, a]$.

Lemma 5.5. *Let $p, q \in \mathbb{N}$ such that q is odd and $p > q > 0$ holds. Then the products $[p][q]$ and $[q][p]$ are cycles of length p.*

Proof. Let p and q be as in the lemma. It is easy to verify that

$$[q][p] = (1, 3, 5, \ldots, q-2, q, 2, 4, 6, \ldots, q-1, q+1, q+2, q+3, \ldots, p),$$

which is a cycle of length p. Because of $[p][q] = [q]^{-1}([q][p])[q]$, also $[p][q]$ is a cycle of length p. □

Lemma 5.6. *Let $p, q \in \mathbb{N}$ be primes such that $2 < q < p$ holds. Then*

$$[p]^{-x_2}[q]^{x_1}([p][q])^{x_2} = [q] = [q]^{x_1}[p]^{-x_2}([p][q])^{x_2} \tag{2}$$

if and only if ($x_1 \equiv 1 \bmod q$ and $x_2 \equiv 0 \bmod p$) or ($x_1 \equiv 0 \bmod q$ and $x_2 \equiv 1 \bmod p$).

Proof. Let p and q be as in the lemma. By Lemma 5.5, $[p][q]$ is a cycle of length p. Therefore, ($x_1 \equiv 1 \bmod q$ and $x_2 \equiv 0 \bmod p$) or ($x_1 \equiv 0 \bmod q$ and $x_2 \equiv 1 \bmod p$) ensures that (2) holds.

For the other direction, assume that x_1 and x_2 are such that (2) holds. We obtain

$$[p]^{-x_2}[q]^{x_1} = [q]^{x_1}[p]^{-x_2}. \tag{3}$$

First of all we show that $x_1 \not\equiv 0 \bmod q$ implies $x_2 \equiv 0 \bmod p$. Assume that $x_1 \not\equiv 0 \bmod q$ and $x_2 \not\equiv 0 \bmod p$. We will deduce a contradiction. We first multiply both sides of (3) by $[p]^{x_2}$ and obtain

$$[q]^{x_1} = [p]^{x_2}[q]^{x_1}[p]^{-x_2}.$$

Since q is prime and $x_1 \not\equiv 0 \bmod q$ we can raise both sides to the power of $x_1^{-1} \bmod q$ and get

$$[q] = [p]^{x_2}[q][p]^{-x_2},$$

from which we obtain

$$[q][p]^{x_2}[q]^{-1} = [p]^{x_2}$$

by multiplying with $[p]^{x_2}[q]^{-1}$. Since $x_2 \not\equiv 0 \bmod p$ and p is prime, we can raise both sides to the power of $x_2^{-1} \bmod p$ which finally gives us

$$[q][p][q]^{-1} = [p].$$

By evaluating of both sides at position p (recall that $p > q$) we get the contradiction

$$p^{[q][p][q]^{-1}} = p^{[p][q]^{-1}} = 1^{[q]^{-1}} = q \neq 1 = p^{[p]},$$

which shows that $x_1 \not\equiv 0 \bmod q$ implies $x_2 \equiv 0 \bmod p$. Obviously $x_1 \equiv 0 \bmod q, x_2 \equiv 0 \bmod p$ is not a solution of (2). This shows that $x_1 \not\equiv 0 \bmod q$ if and only if $x_2 \equiv 0 \bmod p$. It remains to exclude the cases $x_1 \equiv \gamma_1 \bmod q$ for $2 \leq \gamma_1 \leq q-1$ and $x_2 \equiv \gamma_2 \bmod p$ for $2 \leq \gamma_2 \leq p-1$. The equation

$$[p]^{-0}[q]^{x_1}([p][q])^0 = [q] = [q]^{x_1}[p]^{-0}([p][q])^0$$

can only be true if $x_1 \equiv 1 \bmod q$. Hence it remains to show that the equation

$$[p]^{-x_2}[q]^0([p][q])^{x_2} = [q] = [q]^0[p]^{-x_2}([p][q])^{x_2}$$

can only be true if $x_2 \equiv 1 \bmod p$. First we multiply with $([p][q])^{-x_2}$ and get

$$[p]^{-x_2} = [q]([p][q])^{-x_2}.$$

We obtain

$$[p]^{-x_2} = [q][q]^{-1}[p]^{-1}([p][q])^{-x_2+1} = [p]^{-1}([p][q])^{-(x_2-1)}.$$

We multiply with $[p]$ and invert both sides:

$$[p]^{x_2-1} = ([p][q])^{x_2-1}$$

Assume that this equation holds for some $x_2 \not\equiv 1 \bmod p$. By Lemma 5.5 $[p][q]$ is a cycle of length p. Hence we can raise both sides to the power of $(x_2-1)^{-1} \bmod p$ and obtain $[p] = [p][q]$, which is a contradiction since $[q] \neq 1$. This concludes the proof of the lemma. $\square$

Lemma 5.6 makes a reduction of X3HS (Problem 5.2) to 3-knapsack possible. More precisely, the two cases in Lemma 5.6 allow us to simulate for each $a \in A$ the boolean choice, whether a belongs to $A' \subseteq A$ are not.

Theorem 5.7. *The problem* 3*-knapsack for symmetric groups is* **NP***-hard.*

Proof. We provide a log-space reduction from the **NP**-complete problem X3HS (Problem 5.2) to 3-knapsack. Let A be a finite set and $\mathcal{B} \subseteq 2^A$ such that every $C \in \mathcal{B}$ has size 3. W.l.o.g. let $A = \{1, \ldots, n\}$ and let $\mathcal{B} = \{C_1, C_2, \ldots, C_d\}$ where $C_i = \{\alpha(i,1), \alpha(i,2), \alpha(i,3)\}$ for a mapping $\alpha : \{1, \ldots, d\} \times \{1,2,3\} \to \{1, \ldots, n\}$.

Let $p_1, \ldots, p_n, r_1, \ldots, r_n, q_1, \ldots, q_d$ be the first $2m + d$ odd primes such that $p_j > r_j > 2$ and $p_j > q_i > 2$ for $1 \leq i \leq d$ and $1 \leq j \leq n$ hold. Moreover let $P = \max_{1 \leq j \leq n} p_j$. Intuitively, the primes p_j and r_j $(1 \leq j \leq n)$ belong to $j \in \{1, \ldots, n\}$ and the prime q_i $(1 \leq i \leq d)$ belongs to the set C_i.

We will work in the group

$$G = \prod_{j=1}^{n} \mathcal{V}_j \times \prod_{i=1}^{d} \mathcal{C}_i,$$

where $\mathcal{V}_j \leq S_{4p_j + r_j}$ and $\mathcal{C}_i \leq S_{q_i + 3P}$. More precisely we have

$$\mathcal{V}_j = S_{p_j} \times S_{p_j} \times \mathbb{Z}_{p_j} \times \mathbb{Z}_{p_j} \times \mathbb{Z}_{r_j} \text{ and } \mathcal{C}_i = \mathbb{Z}_{q_i} \times S_P \times S_P \times S_P.$$

In the following, we denote the identity element of a symmetric group S_m with id in order to not confuse it with the generator of a cyclic group $\mathbb{Z}_m$.

We now define four group elements $g, g_1, g_2, g_3 \in G$. We write $g = (v_1, \ldots, v_n, c_1, \ldots c_d)$ and $g_k = (v_{k,1}, \ldots, v_{k,n}, c_{k,1}, \ldots, c_{k,d})$ with $v_j, v_{k,j} \in \mathcal{V}_j$ and $c_i, c_{k,i} \in \mathcal{C}_i$. These elements are defined as follows:

$$v_j = ([r_j], \quad [r_j], \quad 0, \quad 0, \quad 0)$$

$$\begin{array}{rcl} v_{1,j} & = & ([r_j], \quad [p_j]^{-1}, \quad 1, \quad 1, \quad 1) \\ v_{2,j} & = & ([p_j]^{-1}, \quad [r_j], \quad -1, \quad 0, -1) \\ v_{3,j} & = & ([p_j][r_j], [p_j][r_j], \quad 0, -1, \quad 0) \end{array}$$

$$\begin{array}{rcl} c_i & = & (1, \mathsf{id}, \quad \mathsf{id}, \quad \mathsf{id}) \\ c_{1,i} & = & (1, [q_i]^{-1}, \quad [p_{\alpha(i,2)}]^{-1}, \quad [q_i][p_{\alpha(i,3)}]) \\ c_{2,i} & = & (1, [q_i][p_{\alpha(i,1)}], [q_i]^{-1}, \quad [p_{\alpha(i,3)}]^{-1}) \\ c_{3,i} & = & (1, [p_{\alpha(i,1)}]^{-1}, \quad [q_i][p_{\alpha(i,2)}], [q_i]^{-1}) \end{array}$$

We claim that there is a subset $A' \subseteq \{1, \ldots, n\}$ such that $|A' \cap C_i| = 1$ for every $1 \leq i \leq d$ if and only if there are $z_1, z_2, z_3 \in \mathbb{Z}$ with

$$g = g_1^{z_1} g_2^{z_2} g_3^{z_3}$$

in the group G. Due to the direct product decomposition of G and the above definition of g, g_1, g_2, g_3, the statement $g = g_1^{z_1} g_2^{z_2} g_3^{z_3}$ is equivalent to the conjunctions of the following statements (read the above definitions of the $v_j, v_{k,j}, c_i, c_{k,i}$ columnwise) for all $1 \leq j \leq n$ and $1 \leq i \leq d$:

(a) $[r_j] = [r_j]^{z_1}[p_j]^{-z_2}([p_j][r_j])^{z_3}$

(b) $[r_j] = [p_j]^{-z_1}[r_j]^{z_2}([p_j][r_j])^{z_3}$

(c) $z_1 \equiv z_2 \bmod p_j$

(d) $z_1 \equiv z_3 \bmod p_j$

(e) $z_1 \equiv z_2 \bmod r_j$

(f) $1 \equiv z_1 + z_2 + z_3 \bmod q_i$

(g) $\mathsf{id} = [q_i]^{-z_1}([q_i][p_{\alpha(i,1)}])^{z_2}[p_{\alpha(i,1)}]^{-z_3}$

(h) $\mathsf{id} = [p_{\alpha(i,2)}]^{-z_1}[q_i]^{-z_2}([q_i][p_{\alpha(i,2)}])^{z_3}$

(i) $\mathsf{id} = ([q_i][p_{\alpha(i,3)}])^{z_1}[p_{\alpha(i,3)}]^{-z_2}[q_i]^{-z_3}$

Recall that by Lemma 5.5, $[p_j][r_j]$ and $[q_i][p_j]$ are cycles of length p_j. Due to the congruences in (c), (d), and (e), the conjunction of (a)–(i) is equivalent to the conjunction of the following equations:

(j) $z_1 \equiv z_2 \equiv z_3 \bmod p_j$

(k) $z_1 \equiv z_2 \bmod r_j$

(l) $[p_j]^{-z_1}[r_j]^{z_2}([p_j][r_j])^{z_1} = [r_j] = [r_j]^{z_2}[p_j]^{-z_1}([p_j][r_j])^{z_1}$

(m) $1 \equiv z_1 + z_2 + z_3 \bmod q_i$

(n) $\mathsf{id} = [q_i]^{-z_1}([q_i][p_{\alpha(i,1)}])^{z_1}[p_{\alpha(i,1)}]^{-z_1}$

(o) $\mathsf{id} = [p_{\alpha(i,2)}]^{-z_1}[q_i]^{-z_2}([q_i][p_{\alpha(i,2)}])^{z_1}$

(p) $\mathsf{id} = ([q_i][p_{\alpha(i,3)}])^{z_1}[p_{\alpha(i,3)}]^{-z_1}[q_i]^{-z_3}$

By Lemma 5.6, the conjunction of (j)–(p) is equivalent to the conjunction of the following statements:

(q) ($z_1 \equiv z_2 \equiv z_3 \equiv 0 \bmod p_j$ and $z_1 \equiv z_2 \equiv 1 \bmod r_j$) or
($z_1 \equiv z_2 \equiv z_3 \equiv 1 \bmod p_j$ and $z_1 \equiv z_2 \equiv 0 \bmod r_j$)

(r) $1 \equiv z_1 + z_2 + z_3 \bmod q_i$

(s) $\mathsf{id} = [q_i]^{-z_1}([q_i][p_{\alpha(i,1)}])^{z_1}[p_{\alpha(i,1)}]^{-z_1}$

(t) $\mathsf{id} = [p_{\alpha(i,2)}]^{-z_1}[q_i]^{-z_2}([q_i][p_{\alpha(i,2)}])^{z_1}$

(u) $\mathsf{id} = ([q_i][p_{\alpha(i,3)}])^{z_1}[p_{\alpha(i,3)}]^{-z_1}[q_i]^{-z_3}$

Let us now assume that $A' \subseteq \{1, \ldots, n\}$ is such that $|A' \cap C_i| = 1$ for every $1 \leq i \leq d$. Let $\sigma : \{1, \ldots, n\} \to \{0, 1\}$ such that $\sigma(j) = 1$ iff $j \in A'$. Thus, $\alpha(i, 1) + \alpha(i, 2) + \alpha(i, 3) = 1$ for all $1 \leq i \leq d$. By the Chinese remainder theorem, we can set $z_1, z_2, z_3 \in \mathbb{Z}$ such that

- $z_1 \equiv z_2 \equiv z_3 \equiv \sigma(j) \bmod p_j$ and $z_1 \equiv z_2 \equiv 1 - \sigma(j) \bmod r_j$ for $1 \leq j \leq n$,
- $z_k \equiv \sigma(\alpha(i, k)) \bmod q_i$ for $1 \leq i \leq d$ and $1 \leq k \leq 3$.

Then (q) and (r) hold. For (s), one has to consider two cases: if $\sigma(\alpha(i, 1)) = 0$, then $z_1 \equiv 0 \bmod q_i$ and $z_1 \equiv 0 \bmod p_{\alpha(i,1)}$. Hence, the right-hand side of (s) evaluates to

$$[q_i]^{-0}([q_i][p_{\alpha(i,1)}])^0[p_{\alpha(i,1)}]^{-0} = \mathsf{id}.$$

On the other hand, if $\sigma(\alpha(i,1)) = 1$, then $z_1 \equiv 1 \bmod q_i$ and $z_1 \equiv 1 \bmod p_{\alpha(i,1)}$ and the right-hand side of (s) evaluates again to

$$[q_i]^{-1}[q_i][p_{\alpha(i,1)}][p_{\alpha(i,1)}]^{-1} = \mathsf{id}.$$

In the same way, one can show that also (t) and (u) hold.

For the other direction, assume that $z_1, z_2, z_3 \in \mathbb{Z}$ are such that (q)–(u) hold. We define $A' \subseteq \{1, \dots, n\}$ such that for every $1 \le j \le n$:

- $j \notin A'$ if $z_1 \equiv z_2 \equiv z_3 \equiv 0 \bmod p_j$ and $z_1 \equiv z_2 \equiv 1 \bmod r_j$, and
- $j \in A'$ if $z_1 \equiv z_2 \equiv z_3 \equiv 1 \bmod p_j$ and $z_1 \equiv z_2 \equiv 0 \bmod r_j$.

Consider a set $C_i = \{\alpha(i,1), \alpha(i,2), \alpha(i,3)\}$. From the equations (s), (t), and (u) we get for every $1 \le i \le d$ and $1 \le k \le 3$:

- if $z_1 \equiv 0 \bmod p_{\alpha(i,k)}$ then $z_k \equiv 0 \bmod q_i$
- if $z_1 \equiv 1 \bmod p_{\alpha(i,k)}$ then $z_k \equiv 1 \bmod q_i$

Together with $1 \equiv z_1 + z_2 + z_3 \bmod q_i$ and $q_i \ge 3$, this implies that there must be exactly one $k \in \{1,2,3\}$ such that $z_1 \equiv 1 \bmod p_{\alpha(i,k)}$. Hence, for every $1 \le i \le d$ there is exactly one $k \in \{1,2,3\}$ such that $\alpha(i,k) \in A'$, i.e., $|\{\alpha(i,1), \alpha(i,2), \alpha(i,3)\} \cap A'| = 1$. □

Theorem 1.6 is an immediate consequence of Corollaries 4.3 and 5.3 and Theorem 5.7.

Theorem 5.7 leads to the question what the exact complexity of the 2-knapsack problem for symmetric groups is. Recall that the complexity of Luks' 2-membership problem is a famous open problem in the algorithmic theory of permutation groups. The restriction of the 2-membership problem to cyclic groups is easier:

Theorem 5.8. *The 2-knapsack problem for symmetric groups belongs to* P.

Proof. Let $a, a_1, a_2 \in S_m$ be permutations and let A, A_1, A_2 be the corresponding permutation matrices. Recall the definition of the Kronecker product of two m-dimensional square matrices A and B: $A \otimes B = (a_{i,j} \cdot B)_{1 \le i,j \le m}$, so it is an m^2-dimensional square matrix with the m^2 blocks $a_{i,j} \cdot B$ for $1 \le i,j \le m$. By [7, Theorem 4], the equation $a_1^{x_1} a_2^{x_2} = a$ is equivalent to

$$(A_2^T \otimes I_m)^{x_2} (I_m \otimes A_1)^{x_1} \mathrm{vec}(I_m) = \mathrm{vec}(A), \tag{4}$$

where $\text{vec}(A) = (A_{1,1}, \ldots, A_{n,1}, A_{1,2}, \ldots, A_{n,2}, \ldots, A_{1,n}, \ldots, A_{n,n})^T$ is the m^2-dimensional column vector obtained from A by stacking all columns of A on top of each other and I_m is the m-dimensional identity matrix. The matrices $A_2^T \otimes I_m$ and $I_m \otimes A_1$ commute; see [7]. By [4, Theorem 1.4] one can finally check in deterministic polynomial time whether (4) has a solution. $\square$

5.3. Another **NP**-*complete special case of 3-membership*

We have already mentioned that Luks [34] proved **NP**-completeness of the membership problem in products GHG where G and H are abelian permutation groups that are given by generators. In the previous section we proved **NP**-completeness of 3-knapsack for symmetric groups, i.e., membership in products $\langle g_1 \rangle \langle g_2 \rangle \langle g_3 \rangle$ for three given permutations g_1, g_2, g_3 (Theorem 5.7). Since our proof yields an instance with $g_1 \neq g_3$, we do not reprove Luks' result. Vice versa, Luks' result does not cover Theorem 5.7 since the permutation groups G and H in Luks' construction are not cyclic.

This leads to the question whether membership in products $\langle g \rangle \langle h \rangle \langle g \rangle$ for given permutations g and h is still **NP**-complete. We do not solve this problem, but we can show the following:

Theorem 5.9. *The following problem is* **NP**-*complete:*
Input: $m \geq 1$ *and permutations* $f, g, h_1, h_2, h_3 \in S_m$ *such that* h_1, h_2, h_3 *pairwise commute.*
Question: Does $f \in \langle g \rangle \langle h_1, h_2, h_3 \rangle \langle g \rangle$ *hold?*

Before we show Theorem 5.9, we first prove two lemmas. Recall that $[a, b]$ denotes the cycle $(a, a + 1, \ldots, b)$.

Lemma 5.10. *For all primes* $q < p$ *and all* $0 \leq e \leq p - q - 1$ *the equation*

$$(1, \ldots, q - 1, p) = [1, p]^x [e + 1, e + q][1, p]^{-x} \tag{5}$$

holds if and only if $x \equiv e + 1 \bmod p$.

Proof. First suppose $x \equiv e + 1 \bmod p$. Recall the general formula for the conjugation of the cycle $(a_1, a_2, \ldots, a_k)$ by a permutation g:

$$g^{-1}(a_1, a_2, \ldots, a_k)g = (a_1^g, a_2^g, \ldots, a_k^g).$$

In our situation this yields

$$[1, p]^x [e + 1, e + q][1, p]^{-x} = ((e + 1)^{[1,p]^{-x}}, (e + 2)^{[1,p]^{-x}}, \ldots, (e + q)^{[1,p]^{-x}}).$$

Since $x \equiv e+1 \bmod p$ we obtain for all $i \in \{1, \dots, q\}$:

$$(e+i)^{[1,p]^{-x}} = (e+i)^{[1,p]^{-(e+1)}} = \begin{cases} p & \text{if } i = 1 \\ i-1 & \text{if } 2 \leq i \leq q. \end{cases}$$

Hence, we obtain

$$[1,p]^x[e+1,e+q][1,p]^{-x} = (p,1,\dots,q-1) = (1,\dots,q-1,p).$$

Vice versa suppose that (5.10) holds. As shown above we have

$$(1,\dots,q-1,p) = [1,p]^{e+1}[e+1,e+q][1,p]^{-(e+1)}.$$

Now suppose there is another solution of (5.10) with $x \not\equiv e+1 \bmod p$, i.e.,

$$(1,\dots,q-1,p) = [1,p]^x[e+1,e+q][1,p]^{-x}.$$

We obtain the equation

$$[1,p]^{e+1}[e+1,e+q][1,p]^{-(e+1)} = [1,p]^x[e+1,e+q][1,p]^{-x},$$

which is equivalent to

$$[e+1,e+q][1,p]^{x-(e+1)}[e+1,e+q]^{-1} = [1,p]^{x-(e+1)}.$$

Since $x \not\equiv e+1 \bmod p$ we can raise both sides to the power of $(x-(e+1))^{-1} \bmod p$ and obtain

$$[e+1,e+q][1,p][e+1,e+q]^{-1} = [1,p]$$

We finally obtain by

$$\begin{aligned} (e+q)^{[e+1,e+q][1,p][e+1,e+q]^{-1}} &= (e+1)^{[1,p][e+1,e+q]^{-1}} \\ &= (e+2)^{[e+1,e+q]^{-1}} \\ &= e+1 \\ &\neq e+q+1 = (e+q)^{[1,p]} \end{aligned}$$

a contradiction. □

Lemma 5.11. *For all $k \geq 1$ and all primes q, p satisfying $q \geq 2$ and $p > kq$ we have: the equation*

$$(1, \ldots, q-1, p) = [1, p]^{x_0} \prod_{i=1}^{k} [(i-1)q+1, iq]^{x_i} [1, p]^{-x_0} \tag{6}$$

holds if and only if for some $i \in \{1, \ldots, k\}$ we have

$$\begin{aligned} x_0 &\equiv (i-1)q + 1 \bmod p, & (7)\\ x_i &\equiv 1 \bmod q \text{ and} & (8)\\ x_j &\equiv 0 \bmod q \text{ for all } j \in \{1, \ldots, k\} \setminus \{i\}. & (9) \end{aligned}$$

Proof. First, suppose $i \in \{1, \ldots, k\}$ is such that (7)–(9) hold. Then we have

$$\begin{aligned} [1,p]^{x_0}[(i-1)q+1, iq]^{x_i}[1,p]^{-x_0} &= [1,p]^{x_0}[(i-1)q+1, iq][1,p]^{-x_0} \\ &= (1, \ldots, q-1, p) \end{aligned}$$

by Lemma 5.10.

Vice versa suppose that (6) hold. Note that this implies that we cannot have $x_i \equiv 0 \bmod q$ for all $i \in \{1, \ldots, k\}$. Let $j \in \{1, \ldots, k\}$ be the number of variables x_i ($1 \leq i \leq k$) such that $x_i \not\equiv 0 \bmod q$. This implies that the right-hand side of (6) consists of j pairwise disjoint cycles of length q plus some fixpoints. Since $(1, \ldots, q-1, p)$ is a single cycle of length q, we must have $j = 1$. Let $i \in \{1, \ldots, k\}$ be the unique element with $x_i \not\equiv 0 \bmod q$. Then (6) simplifies to

$$(1, \ldots, q-1, p) = [1, p]^{x_0}[(i-1)q+1, iq]^{x_i}[1, p]^{-x_0}. \tag{10}$$

Clearly $x_0 \equiv 0 \bmod p$ is not possible in (10). Let y_0 and y_i be such that $1 \leq y_0 < p$, $1 \leq y_i < q$, $y_0 \equiv x_0 \bmod p$ and $y_i \equiv x_i \bmod q$. If $y_0 \notin \{(i-1)q+1, \ldots, iq\}$, we get by

$$p^{[1,p]^{x_0}[(i-1)q+1,iq]^{x_i}[1,p]^{-x_0}} = y_0^{[(i-1)q+1,iq]^{x_i}[1,p]^{-x_0}} = y_0^{[1,p]^{-x_0}} = p \neq 1 = p^{(1,\ldots,q-1,p)}$$

a contradiction to (10). Thus we have $(i-1)q + 1 \leq y_0 \leq iq$. We get

$$\begin{aligned} 1 = p^{(1,\ldots,q-1,p)} &= p^{[1,p]^{x_0}[(i-1)q+1,iq]^{x_i}[1,p]^{-x_0}} \\ &= y_0^{[(i-1)q+1,iq]^{x_i}[1,p]^{-x_0}} \end{aligned}$$

$$= \begin{cases} (y_0 + y_i)^{[1,p]^{-x_0}} & \text{if } y_0 + y_i \leq iq \\ (y_0 + y_i - q)^{[1,p]^{-x_0}} & \text{otherwise} \end{cases}$$
$$= \begin{cases} y_i & \text{if } y_0 + y_i \leq iq \\ p + y_i - q & \text{otherwise.} \end{cases}$$

The case $1 = p+y_i-q$ leads to a contradiction since $p+y_i-q \geq p+1-q \geq 2$. Hence we obtain $y_i = 1$, i.e., $x_i \equiv 1 \bmod q$. This simplifies (10) further to

$$(1, \dots, q-1, p) = [1,p]^{x_0}[(i-1)q+1, iq][1,p]^{-x_0}.$$

Since $0 \leq (i-1)q \leq (k-1)q \leq p-q-1$, Lemma 5.10 yields $x_0 \equiv (i-1)q + 1 \bmod p$. □

In the proof of Theorem 5.9 below we use Lemma 5.11 only for the case $k = 3$. It will allow us to simulate in the problem X3HS (Problem 5.2) the selection of a unique element from each $C \in \mathcal{B}$ (recall that all the set in $\mathcal{B}$ have size 3.

We use the following notation in the proof below: For a finite set $U = \{s_1, s_2, \dots, s_m\} \subseteq \mathbb{N}$ with $s_1 < s_2 < \cdots < s_m$ and $1 \leq k \leq m$ we write $U(k) = s_k$.

Proof of Theorem 5.9. We give a log-space reduction from X3HS. Let A be a finite set and $\mathcal{B} \subseteq 2^A$ be a set of subsets of A all of size 3. W.l.o.g. assume that $A = \{1, \dots, n\}$ and let $\mathcal{B} = \{C_1, \dots, C_d\}$. For $j \in \{1, \dots, n\}$ we denote by $D_j \subseteq \{1 \dots, d\}$ the set of all numbers i such that $j \in C_i$. Let $q_1 < \cdots < q_d$ be the first d primes. Let $p_1 < \cdots < p_d$ be the next d primes with $p_1 > 3q_d$ and let $r_1 < \cdots < r_n$ be the next n primes with $r_1 > p_d$. Intuitively we associate $C_i \in \mathcal{B}$ with the prime q_i and $j \in \{1, \dots, n\}$ with the prime r_j. We will work with the group

$$G = \prod_{i=1}^{d}(S_{p_i} \times \mathbb{Z}_{p_i}) \times \prod_{j=1}^{n}(S_{r_j}^{|D_j|} \times \mathbb{Z}_{r_j}),$$

which naturally embedds into S_m for $m = 2\sum_{i=1}^{d} p_i + \sum_{j=1}^{n}(|D_j|+1) \cdot r_j$.

We define the input group elements $f, g, h_1, h_2, h_3 \in G$ as follows, where i ranges over $\{1 \dots, d\}$, j ranges over $\{1 \dots, n\}$, k ranges over $\{1, \dots, |D_j|\}$, and ℓ ranges over $\{1, 2, 3\}$:

$$f \;=\; (f_1, \dots, f_d, f_1', \dots, f_n') \text{ with}$$

$$
\begin{aligned}
f_i &= ((1,\ldots,q_i-1,p_i),0)\\
f'_j &= (f_{j,1},\ldots,f_{j,|D_j|},0)\\
f_{j,k} &= (1,\ldots,q_{D_j(k)}-1,r_j)\\
g &= (g_1,\ldots,g_d,g'_1,\ldots,g'_n) \text{ with}\\
g_i &= ([1,p_i],1)\\
g'_j &= (\underbrace{[1,r_j],\ldots,[1,r_j]}_{|D_j| \text{ many}},1)\\
h_\ell &= (h_{\ell,1},\ldots,h_{\ell,d},h'_{\ell,1},\ldots,h'_{\ell,n})\\
h_{\ell,i} &= ([(\ell-1)q_i+1,\ell q_i],0)\\
h'_{\ell,j} &= (\alpha_{\ell,j,1},\ldots,\alpha_{\ell,j,|D_j|},0)\\
\alpha_{\ell,j,k} &= \begin{cases}[1,q_{D_j(k)}] & \text{if } j=C_{D_j(k)}(\ell)\\ [q_d+1,q_d+q_{D_j(k)}] & \text{otherwise.}\end{cases}
\end{aligned}
$$

Note that $j = C_{D_j(k)}(\ell)$ means that j is the ℓ^{th} element (in natural order) in the k^{th} set among all sets C_i containing j, where the sets C_i containing j are ordered with respect to their position in the list $C_1, C_2, \ldots, C_d$. Observe that the elements h_1, h_2, h_3 pairwise commute.

Now we will show that $f \in \langle g\rangle\langle h_1,h_2,h_3\rangle\langle g\rangle$ if and only if there is a subset $A' \subseteq \{1,\ldots,n\}$ such that $|A' \cap C_i| = 1$ for all $i \in \{1,\ldots,d\}$.

First suppose that $f \in \langle g\rangle\langle h_1,h_2,h_3\rangle\langle g\rangle$. Since h_1, h_2, h_3 pairwise commute, we obtain integers x_0, x'_0, x_1, x_2, x_3 with

$$f = g^{x_0} h_1^{x_1} h_2^{x_2} h_3^{x_3} g^{x'_0}.$$

Note that the projections onto the direct factors $\mathbb{Z}_{p_i}$ and $\mathbb{Z}_{r_j}$ of G ensure that

$$x'_0 + x_0 \equiv 0 \bmod \prod_{i=1}^{d} p_i \prod_{j=1}^{n} r_j.$$

Since the order of g is $\prod_{i=1}^{d} p_i \prod_{j=1}^{n} r_j$, we can w.l.o.g. assume that $x'_0 = -x_0$. We therefore obtain

$$f = g^{x_0} h_1^{x_1} h_2^{x_2} h_3^{x_3} g^{-x_0}. \tag{11}$$

We define the subset $A' \subseteq \{1,\ldots,n\}$ by

$$A' = \{j \mid 1 \leq j \leq n, x_0 \equiv 1 \bmod r_j\}.$$

We claim that for every $i \in \{1, \ldots, d\}$ we have $|C_i \cap A'| = 1$. Consider a specific $i \in \{1, \ldots, d\}$. The projection onto the direct factor S_{p_i} of G gives us

$$(1, \ldots, q_i - 1, p_i) = [1, p_i]^{x_0}[1, q_i]^{x_1}[q_i + 1, 2q_i]^{x_2}[2q_i + 1, 3q_i]^{x_3}[1, p_i]^{-x_0}. \quad (12)$$

By Lemma 5.11 this equation ensures that there is exactly one $a \in \{1, 2, 3\}$ such that $x_a \equiv 1 \bmod q_i$ and $x_b \equiv 0 \bmod q_i$ for all $b \in \{1, 2, 3\} \setminus \{a\}$.

Consider a $j \in C_i$. Then $i \in D_j$ and there is a $k \in \{1, \ldots, |D_j|\}$ such that $i = D_j(k)$. Projecting (11) onto the k^{th} direct factor S_{r_j} in $S_{r_j}^{|D_j|}$ yields

$$\begin{aligned}(1, \ldots, q_i - 1, r_j) &= (1, \ldots, q_{D_j(k)} - 1, r_j) \\ &= [1, r_j]^{x_0}\alpha_{1,j,k}^{x_1}\alpha_{2,j,k}^{x_2}\alpha_{3,j,k}^{x_3}[1, r_j]^{-x_0}. \end{aligned} \quad (13)$$

Since $x_a \equiv 1 \bmod q_i$, $x_b \equiv 0 \bmod q_i$ for all $b \in \{1, 2, 3\} \setminus \{a\}$, and every $\alpha_{\ell,j,k}$ is a cycle of length $q_{D_j(k)} = q_i$, the expression (13) simplifies to $[1, r_j]^{x_0}\alpha_{a,j,k}[1, r_j]^{-x_0}$. Hence, we obtain

$$(1, \ldots, q_i - 1, r_j) = [1, r_j]^{x_0}\alpha_{a,j,k}[1, r_j]^{-x_0}.$$

Moreover, with the definition of $\alpha_{a,j,k}$ this yields

$$(1, \ldots, q_i - 1, r_j) = [1, r_j]^{x_0} \left\{ \begin{array}{ll} [1, q_i] & \text{if } j = C_i(a) \\ [q_d + 1, q_d + q_i] & \text{otherwise} \end{array} \right\} [1, r_j]^{-x_0}.$$

With Lemma 5.10 this implies the following:

- $x_0 \equiv 1 \bmod r_j$ (and hence $j \in A'$) if $j = C_i(a)$, and
- $x_0 \equiv q_d + 1 \bmod r_j$ (and hence $j \notin A'$) if $j \neq C_i(a)$.

Hence, there is exactly one $j \in A' \cap C_i$, namely $C_i(a)$.

Vice versa, suppose there is a subset $A' \subseteq \{1, \ldots, n\}$ such that $|A' \cap C_i| = 1$ for all $i \in \{1, \ldots, d\}$. In order to satisfy for every $i \in \{1, \ldots, d\}$ equation 12 (the projection of (11) onto the direct factor S_{p_i} of G) we consider the unique element in $A' \cap C_i$. Assume that it is $C_i(a)$ $(1 \leq a \leq 3)$. We then set

- $x_a \equiv 1 \bmod q_i$,
- $x_b \equiv 0 \bmod q_i$ for $b \in \{1, 2, 3\} \setminus \{a\}$, and

- $x_0 \equiv (a-1)q_i + 1 \bmod p_i$.

By Lemma 5.11 this choice satisfies (12). Moreover, for different $i \in \{1, \dots, d\}$ the above choices do not conflict with each other since they refer to different primes.

Now consider the projection

$$(1, \dots, q_{D_j(k)} - 1, r_j) = [1, r_j]^{x_0} \alpha_{1,j,k}^{x_1} \alpha_{2,j,k}^{x_2} \alpha_{3,j,k}^{x_3} [1, r_j]^{-x_0} \tag{14}$$

of (11) onto the k^{th} direct factor S_{r_j} in $S_{r_j}^{|D_j|}$, where $1 \le j \le n$ and $1 \le k \le |D_j|$. In order to satisfy this equation, we choose $x_0 \equiv 1 \bmod r_j$ if $j \in A'$ and $x_0 \equiv q_d + 1 \bmod r_j$ if $j \notin A'$.

Let $D_j(k) = i \in \{1, \dots, d\}$ so that $j \in C_i$. As above, assume that the unique element in $A' \cap C_i$ is $C_i(a)$ $(1 \le a \le 3)$. By the above choices for $x_\ell \bmod q_i$ $(1 \le \ell \le 3)$ and the fact that all $\alpha_{\ell,j,k}$ are cycles of length $q_{D_j(k)} = q_i$, equation 14 reduces to

$$(1, \dots, q_i - 1, r_j) = [1, r_j]^{x_0} \alpha_{a,j,k} [1, r_j]^{-x_0}. \tag{15}$$

There are now two cases:

Case 1. $j \in A'$ and thus $x_0 \equiv 1 \bmod r_j$ and $j \in A' \cap C_i$. Hence, we have $j = C_i(a) = C_{D_j(k)}(a)$. We thus have $\alpha_{a,j,k} = [1, q_{D_j(k)}] = [1, q_i]$ and (15) becomes

$$(1, \dots, q_i - 1, r_j) = [1, r_j]^{x_0} [1, q_i] [1, r_j]^{-x_0},$$

which is true by Lemma 5.10 since $x_0 \equiv 1 \bmod r_j$.

Case 2. $j \notin A'$ and thus $x_0 \equiv q_d + 1 \bmod r_j$. Hence, $j \neq C_i(a)$ and we get $\alpha_{a,j,k} = [q_d + 1, q_d + q_i]$. Hence, (15) becomes

$$(1, \dots, q_i - 1, r_j) = [1, r_j]^{x_0} [q_d + 1, q_d + q_i] [1, r_j]^{-x_0},$$

which is true by Lemma 5.10 since $x_0 \equiv q_d + 1 \bmod r_j$. □

The above proof also shows the following result:

Theorem 5.12. *The following problem is* **NP***-complete:*
Input: $m \ge 1$ and permutations $f, g, h_1, h_2, h_3 \in S_m$ such that h_1, h_2, h_3 pairwise commute.
Question: Is there $i \in \mathbb{N}$ such that $f \in g^i \langle h_1, h_2, h_3 \rangle g^{-i}$?

6. Application to intersection problems

In this section we prove Theorems 1.12 and 1.13. The proofs of the two results are almost identical. Let us show how to deduce Theorem 1.12 from Theorem 1.10. Let $\mathcal{G}$ be a grammar from CFG(k) and let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be a list of group DFAs. Let $\mathcal{A}_i = (Q_i, \Sigma, q_{i,0}, \delta_i, F_i)$. W.l.o.g. we assume that the Q_i are pairwise disjoint and let $Q = \bigcup_{1 \le i \le n} Q_i$. To every $a \in \Sigma$ we can associate a permutation $\pi_a \in S_Q$ by setting $\pi_a(q) = \delta_i(q, a)$ if $q \in Q_i$. Let $\mathcal{G}' \in \mathrm{CFG}(k)$ be the context-free grammar over the terminal alphabet S_Q obtained by replacing in $\mathcal{G}$ every occurence of $a \in \Sigma$ by π_a. Then, we have $L(\mathcal{G}) \cap \bigcap_{1 \le i \le n} L(\mathcal{A}_i) \neq \emptyset$ if and only if there exists a permutation $\pi \in L(\mathcal{G}')$ such that $\pi(q_{i,0}) \in F_i$ for every $1 \le i \le n$. We can nondeterministically guess such a permutation and check $\pi \in L(\mathcal{G}')$ in **NP** using Theorem 1.10. This proves the upper bound from Theorems 1.12. The lower bound already holds for the case that $L(\mathcal{G}) = \Sigma^*$ [10].

The proof of the upper bound in Theorem 1.13 is identical to the above proof, except that we use Theorem 1.9. For the lower bound, notice that the **PSPACE**-complete context-free membership problem for symmetric groups can be directly reduced to the intersection non-emptiness problem from Theorem 1.13 (several group DFAs and a single context-free grammar): Take a context-free gammar $\mathcal{G}$ over the terminal alphabet S_m. Let $\{\pi_1, \ldots, \pi_n\}$ be the permutations that appear as terminal symbols in $\mathcal{G}$. Let $\mathcal{G}'$ be the context-free gammar obtained from $\mathcal{G}$ by replacing every occurrence of π_i by a new terminal symbol a_i. We construct m group DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_m$ over the terminal alphabet $\{a_1, \ldots, a_n\}$ and state set $\{1, \ldots, m\}$. The initial and (unique) final state of $\mathcal{A}_i$ is i and the transition function of every $\mathcal{A}_i$ is the same function δ with $\delta(q, a_i) = q^{\pi_i}$ for $1 \le q \le m$. Then we have $L(\mathcal{G})$ contains the identity permutation if and only if $L(\mathcal{G}') \cap \bigcap_{1 \le i \le m} L(\mathcal{A}_i)$ is non-empty.

Acknowledgement

This work has been supported by the DFG research project LO 748/12-2.

References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[2] Emmanuel Arrighi, Henning Fernau, Stefan Hoffmann, Markus Holzer, Ismaël Jecker, Mateus de Oliveira Oliveira, and Petra Wolf. On the complexity of intersection non-emptiness for star-free language classes. In *Proceedings of the 41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021*, volume 213 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.34.

[3] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

[4] László Babai, Robert Beals, Jin-Yi Cai, Gábor Ivanyos, and Eugene M. Luks. Multiplicative equations over commuting matrices. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1996*, pages 498–507. ACM/SIAM, 1996.

[5] László Babai, Eugene M. Luks, and Ákos Seress. Permutation groups in NC. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC 1987*, pages 409–420. ACM, 1987. doi:10.1145/28395.28439.

[6] László Babai and Endre Szemerédi. On the complexity of matrix group problems I. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science, FOCS 1984*, pages 229–240. IEEE Computer Society, 1984. doi:10.1109/SFCS.1984.715919.

[7] Paul Bell, Vesa Halava, Tero Harju, Juhani Karhumäki, and Igor Potapov. Matrix equations and Hilbert's tenth problem. *International Journal of Algebra and Computation*, 18(8):1231–1241, 2008. doi:10.1142/S0218196708004925.

[8] Pascal Bergsträßer, Moses Ganardi, and Georg Zetzsche. A characterization of wreath products where knapsack is decidable. In *Proceeding of the 38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021*, volume 187 of *LIPIcs*, pages 11:1–11:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.11.

[9] Jean-Camille Birget, Stuart Margolis, John Meakin, and Pascal Weil. PSPACE-complete problems for subgroups of free groups and inverse finite automata. *Theoretical Computer Science*, 242(1-2):247–281, 2000. doi:10.1016/S0304-3975(98)00225-4.

[10] Michael Blondin, Andreas Krebs, and Pierre McKenzie. The complexity of intersecting finite automata having few final states. *Computational Complexity*, 25(4):775–814, 2016. doi:10.1007/s00037-014-0089-9.

[11] Stephen A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *Journal of the Association for Computing Machinery*, 18(1):4–18, 1971. doi:10.1145/321623.321625.

[12] Mateus de Oliveira Oliveira and Michael Wehar. On the fine grained complexity of finite automata non-emptiness of intersection. In *Proceedings of the 24th International Conference Developments in Language Theory, DLT 2020*, volume 12086 of *Lecture Notes in Computer Science*, pages 69–82. Springer, 2020. doi:10.1007/978-3-030-48516-0_6.

[13] Javier Esparza, Antonín Kucera, and Stefan Schwoon. Model checking LTL with regular valuations for pushdown systems. *Information and Computation*, 186(2):355–376, 2003. doi:10.1016/S0890-5401(03)00139-1.

[14] Javier Esparza, Michael Luttenberger, and Maximilian Schlund. A brief history of Strahler numbers. In *Proceedings of the 8th International Conference on Language and Automata Theory and Applications, LATA 2014*, volume 8370 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2014. doi:10.1007/978-3-319-04921-2_1.

[15] Shimon Even and Oded Goldreich. The minimum-length generator sequence problem is NP-hard. *Journal of Algorithms*, 2(3):311–313, 1981. doi:10.1016/0196-6774(81)90029-8.

[16] Michael Figelius, Moses Ganardi, Markus Lohrey, and Georg Zetzsche. The complexity of knapsack problems in wreath products. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, volume 168 of *LIPIcs*, pages 126:1–126:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.126.

[17] Elizaveta Frenkel, Andrey Nikolaev, and Alexander Ushakov. Knapsack problems in products of groups. *Journal of Symbolic Computation*, 2015. doi:10.1016/j.jsc.2015.05.006.

[18] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–completeness*. Freeman, 1979.

[19] Alexander Heußner, Jérôme Leroux, Anca Muscholl, and Grégoire Sutre. Reachability analysis of communicating pushdown systems. In *Proceedings of the 13th International Conference on Foundations of Software Science and Computational Structures, FOSSACS 2010*, volume 6014 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 2010. doi:10.1007/978-3-642-12032-9_19.

[20] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[21] Robert E. Horton. Erosional development of streams and their drainage basins: hydro-physical approach to quantitative morphology. *Geological Society of America Bulletin*, 56(3):275–370, 1945. doi:https://doi.org/10.1130/0016-7606(1945)56[275:EDOSAT]2.0.CO;2.

[22] Trevor Jack. On the complexity of properties of partial bijection semigroups, 2021. doi:10.48550/ARXIV.2101.00324.

[23] Mark Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36:265–289, 1985. doi:10.1016/0304-3975(85)90047-7.

[24] Mark Kambites, Pedro V. Silva, and Benjamin Steinberg. On the rational subset problem for groups. *Journal of Algebra*, 309(2):622–639, 2007. doi:10.1016/j.jalgebra.2006.05.020.

[25] Ilya Kapovich and Alexei Myasnikov. Stallings foldings and subgroups of free groups. *Journal of Algebra*, 248(2):608–668, 2002. doi:10.1006/jabr.2001.9033.

[26] Arthur A. Khashaev. On the membership problem for finite automata over symmetric groups. *Discrete Mathematics and Applications*, 32(6):389–395, 2022. URL: https://doi.org/10.1515/dma-2022-0033, doi:doi:10.1515/dma-2022-0033.

[27] Daniel König, Markus Lohrey, and Georg Zetzsche. Knapsack and subset sum problems in nilpotent, polycyclic, and co-context-free groups. In *Algebra and Computer Science*, volume 677 of *Contemporary Mathematics*, pages 138–153. American Mathematical Society, 2016. doi:10.1090/conm/677.

[28] Dexter Kozen. Lower bounds for natural proof systems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science, FOCS 1977*, pages 254–266. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.16.

[29] Markus Lohrey. *The rational subset membership problem for groups: a survey*, page 368–389. London Mathematical Society Lecture Note Series. Cambridge University Press, 2015. doi:10.1017/CBO9781316227343.024.

[30] Markus Lohrey. Knapsack in hyperbolic groups. *Journal of Algebra*, 545(1):390–415, 2020. doi:https://doi.org/10.1016/j.jalgebra.2019.04.008.

[31] Markus Lohrey, Andreas Rosowski, and Georg Zetzsche. Membership problems in finite groups. In *Proceedings of the 47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, volume 241 of *LIPIcs*, pages 71:1–71:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.MFCS.2022.71.

[32] Markus Lohrey and Georg Zetzsche. Knapsack in graph groups. *Theory of Computing Systems*, 62(1):192–246, 2018. doi:10.1007/s00224-017-9808-3.

[33] 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. doi:10.1016/0022-0000(82)90009-5.

[34] Eugene M. Luks. Permutation groups and polynomial-time computation. In *Groups And Computation, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, October 7-10, 1991*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 139–175. DIMACS/AMS, 1991. doi:10.1090/dimacs/011/11.

[35] Alexei Myasnikov, Andrey Nikolaev, and Alexander Ushakov. Knapsack problems in groups. *Mathematics of Computation*, 84:987–1016, 2015. doi:10.1090/S0025-5718-2014-02880-9.

[36] Ákos Seress. *Permutation Group Algorithms*. Cambridge Tracts in Mathematics. Cambridge University Press, 2003. doi:10.1017/CBO9780511546549.

[37] Charles C. Sims. Computational methods in the study of permutation groups. In *Computational Problems in Abstract Algebra*, pages 169–183. Pergamon, 1970. doi:10.1016/B978-0-08-012975-4.50020-5.

[38] Arthur N. Strahler. Hypsometric (area-altitude) analysis of erosional topology. *Geological Society of America Bulletin*, 63(11):1117–1142, 1952. doi:https://doi.org/10.1130/0016-7606(1952)63[1117:HAAOET]2.0.CO;2.

[39] Joseph Swernofsky and Michael Wehar. On the complexity of intersecting regular, context-free, and tree languages. In *Proceedings of the 42nd International Colloquium Automata, Languages, and Programming, Part II, ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 414–426. Springer, 2015. doi:10.1007/978-3-662-47666-6_33.

Appendix A. Testing membership in CFG(k)

In view of Theorem 1.10, the reader might ask how difficult it is to check for a fixed constant k whether a given context-free grammar in Chomsky normal form belongs to the class $\mathrm{CFG}(k)$. We do not know whether a polynomial time algorithm exists for this problem. In this appendix, we show that the problem belongs to **coNP** (the class of all problems whose complement belongs to NP.

Lemma Appendix A.1. *Let $\mathcal{G} = (N, T, P, S)$ be a context-free grammar. We have $L(\mathcal{G}) \neq \emptyset$ if and only if $\mathcal{G}$ has an acyclic derivation tree whose root is labelled with S.*

Proof. Clearly, if there is an acyclic derivation tree, then there is a derivation tree and hence $L(\mathcal{G}) \neq \emptyset$. For the reverse implication note that an arbitrary derivation tree can be made acyclic (as in the proof of the pumping lemma for context-free languages). □

Theorem Appendix A.2. *For every fixed $k \geq 1$, the problem of checking whether a given context-free grammar belongs to CFG(k) is in* **coNP**.

Proof. Let $\mathcal{G} = (N, \Sigma, P, S)$ be a context-free grammar in Chomsky normal form. We have $\mathcal{G} \in \mathrm{CFG}(k)$ if and only if for every acyclic derivation tree the Horton-Strahler number is at most k. By this we obtain $\mathcal{G} \notin \mathrm{CFG}(k)$ if and only if there is an acyclic derivation tree with Horton-Strahler number greater than k. By Lemma 2.3 this holds if and only if one of the following conditions holds:

- There is an acyclic derivation tree with at most $2(|N|+1)^k$ nodes and Horton-Strahler number greater than k.
- There is an acyclic derivation tree with more than $2(|N|+1)^k$ nodes.

The second statement holds if and only if there is a partial acyclic derivation tree T with $2(|N|+1)^k < |T| \leq 2(|N|+1)^k + 2$ ($|T|$ denotes the number of nodes of T) and for every leave v in T that is labelled with a nonterminal A there is an acyclic derivation tree T_v of arbitrary size whose root is labelled with A and which contains no nonterminal that has already appeared on the path from the root of T to node v. This holds, since in an acyclic derivation tree with more than $2(|N|+1)^k$ nodes we can remove subtrees such that the resulting partial acyclic derivation tree T' satisfies $2(|N|+1)^k < |T'| \leq 2(|N|+1)^k + 2$.

These conditions can be checked in **NP** as follows: First, we guess an acyclic derivation tree with at most $2(|N|+1)^k$ nodes and compute in polynomial time its Horton-Strahler number s. If $s > k$ then we accept. If $s \leq k$, then we guess a partial acyclic derivation tree T with $2(|N|+1)^k < |T| \leq 2(|N|+1)^k + 2$. For every leaf v of T that is labelled with a nonterminal A we define the subgrammar $G_v = (N_v, T, P_v, A)$: let $A_1, \ldots, A_d$ ($A_d = A$) be the nonterminals that appear on the path from the root of T to the leaf v. Then we set $N_v = N \setminus \{A_1, \ldots, A_{d-1}\}$. Moreover, P_v is obtained from P by removing every production that contains one of the nonterminals $A_1, \ldots, A_{d-1}$. Finally the algorithm verifies deterministically in polynomial time whether G_v has an acyclic derivation tree T_v of arbitrary size that is rooted in A. By Lemma Appendix A.1 this holds if and only if $L(G_v) \neq \emptyset$. □