Membership Problems in Finite Groups

Markus Lohrey
Universität Siegen, Germany

Andreas Rosowski
Universität Siegen, Germany

Georg Zetzsche
Max Planck Institute for Software Systems (MPI-SWS), 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 [36], 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.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases algorithms for finite groups, intersection non-emptiness problems, knapsack problems in groups

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.71

Related Version *Full Version*: https://arxiv.org/abs/2206.11756

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 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. 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 sum problem, see e.g. [31, 37]. It is a simple observation that for the rational subset membership problem, the word w (which is tested for membership in $h^{-1}(h(L))$ can be assumed to be the empty word, see [28, 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 subgroup of S_m generated by $a_1, \ldots, a_n$. The famous Schreier-Sims algorithm solves this problem in polynomial time [39], and the problem is known to be in NC [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 [36] for details. On the other hand, Luks has shown in [36] that k-membership is NP-complete for every $k \geq 3$. In fact, NP-hardness of 3-membership holds for the special case where $G_1 = G_3$ and G_1 and G_2 are both abelian.

Note that the k-membership problem is a special case of 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 and a nondeterministic finite automaton (NFA) $\mathcal{A}$ over the alphabet S_m.
Question: Does $1 \in L(\mathcal{A})$ hold?

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, 8, 9, 21, 22, 29, 32, 34, 37]. 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 and permutations $a, a_1, \ldots, a_n \in S_m$.
Question: Are there $i_1, \ldots, i_n \in \{0, 1\}$ such that $a = a_1^{i_1} \cdots a_n^{i_n}$?

Note that the subset sum problem is the membership problem for *cubes*, which are subsets of the form $\{a_1^{i_1} \cdots a_n^{i_n} \mid i_1, \ldots, i_n \in \{0, 1\}\}$ [6].

[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.

▶ **Problem 1.3** (knapsack problem for symmetric groups).
Input: a unary encoded number m and permutations $a, a_1, \ldots, a_n \in S_m$.
Question: Are there $i_1, \ldots, i_n \in \mathbb{N}$ 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 and pairwise commuting permutations $a, a_1, \ldots, a_n \in S_m$.
Question: Are there $i_1, \ldots, i_n \in \{0, 1\}$ 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 and $k+1$ permutations $a, a_1, \ldots, a_k \in S_m$.
Question: Are there $i_1, \ldots, i_k \in \mathbb{N}$ such that $a = a_1^{i_1} \cdots a_k^{i_k}$?

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

▶ **Theorem 1.6.** *Problems 1.1–1.4 and Problem 1.5 for $k \geq 3$ are* NP*-complete.*

In contrast, we will show that the 2-knapsack problem can be solved in polynomial time (Theorem 5.7). The NP upper bound for the rational subset membership problem will be shown for black-box groups.

▶ Remark 1.7. The abelian variant of the knapsack problem, i.e., Problem 1.3 with the additional restriction that the permutations $s_1, \ldots, s_n$ 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 $i_1, \ldots, i_k$) 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: A derivation tree of a context-free grammar is called *acyclic* if no nonterminal appears twice on a path from the root to a leaf. Hence, the height of an acyclic derivation tree is bounded by the number of nonterminals of the grammar. The *Horton-Strahler number* $\mathrm{hs}(t)$ of a binary tree t (introduced by Horton and Strahler in the context of hydrology [25, 40]; see [19] 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 $\mathrm{hs}(t) = \max\{\mathrm{hs}(t_1), \mathrm{hs}(t_2)\}$. For $k \geq 1$ let $\mathrm{CFG}(k)$ be the set of all context-free grammars in Chomsky normal form (hence, derivation trees are binary trees if we ignore the leaves labelled with terminal symbols) 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 $\mathrm{CFG}(k)$ is* NP*-complete.*

Table 1 Complexity of various intersection non-emptiness problems.

	no CFG	one CFG(k)	one CFG
DFAs	PSPACE-c. [30]	EXPTIME-c. for k large enough	EXPTIME-c. [41]
group DFAs	NP-c. [11]	NP-c. for all $k \geq 1$	PSPACE-c.

Theorem 1.10 generalizes the statement for rational subsets in Theorem 1.6: Every regular grammar (when brought into Chomsky normal form) belongs to CFG(1). Linear context-free grammars belong to CFG(1) as well. Note that Theorem 1.10 is a promise problem in the sense that coNP is the best upper bound for testing whether a given context-free grammar belongs to CFG(k) that we are aware of; see [33, Theorem A.2]. The upper bounds in Theorems 1.6, 1.9, and 1.10 will actually be 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 [30] 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. [11]. Based on Cook's characterization of EXPTIME by polynomially space bounded AuxPDAs [13], Swernofsky and Wehar [41] showed that the intersection non-emptiness problem is EXPTIME-complete for a list of general DFAs and a single context-free grammar; see also [24, p. 275] and see [18] for a related EXPTIME-complete problem. Using Theorems 1.9 and 1.10 we can easily show the following new results:

▶ **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* $\mathcal{G} \in \mathrm{CFG}(k)$*.*
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 the EXPTIME-hardness proof from [41] works for a fixed context-free grammar. Moreover, every fixed context-free grammar belongs to CFG(k) for some $k \geq 1$.

Related work. Computationally problems for permutation groups have a long history (see e.g. the text book [38]), and have applications, e.g. for graph isomorphism testing [3, 35]. A problem that is similar to subset sum is the *minimum generator sequence problem* (MGS) [20]: 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 [20]. For the case, where the number ℓ is given in binary representation, the problem is PSPACE-complete [27]. This yields in fact the PSPACE-hardness in Theorem 1.9.

Intersection nonemptiness problems have been studied intensively in recent years, see e.g. [1, 14]. The papers [10, 26] prove PSPACE-hardness of the intersection nonemptiness problem for inverse automata (DFAs, where the transition monoid is an inverse monoid).

There are several algorithms for context-free (and other) grammars that exploit that an input grammar is *index-m*, meaning that *all* derivation trees have Horton-Strahler number $\leq m$, where m is part of the input in unary notation [17, 12, 16, 15, 7, 2] (see [19] for a survey). This assumption, called *finite-index*, is incomparable to our CFG(k)-assumption: In CFG(k), the k is fixed but one only considers acyclic derivation trees.

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 $g_i = g_j g_k$. Then, $\mathcal{S}$ is said to *produce* g_n. The size of $\mathcal{S}$ is n. The following result is known as the reachability theorem [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 $Q = \{1, \ldots, m\}$ 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. Recall the definition of the Horton-Strahler number $\mathrm{hs}(t)$ of binary trees t from the introduction. We need the following simple fact; see [33, Lemma 2.3]. Here, the *height* of a tree is the maximal number of edges on a path from the root to a leaf.

▶ **Lemma 2.3.** *Let t be a binary tree of height d and let $s = \mathrm{hs}(t)$. Then, t has at most d^s many leaves and therefore at most $2 \cdot d^s$ many nodes.*

Formal languages. We assume that the reader is familiar with basic definitions from automata theory. 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 $q_0 \in Q$ 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.

Context-free grammars will be always in Chomsky normal form. When we speak of a *derivation tree* of a context-free grammar, we always assume that the root of the tree is labelled with the start nonterminal and every leaf is labelled with a terminal symbol. When we talk about the Horton-Strahler number of such a tree, we remove all terminal-labelled leaves so that the resulting tree is a binary tree (due to the Chomsky normal form). In a *partial derivation tree*, we also allow leaves labelled with nonterminals (but we still assume that the root is labelled with the start nonterminal).

3 Black box groups

More details on black box groups can be found in [6, 38]. 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 (the latter operation is not allowed in [6]). As usual, each execution of an oracle 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:

- $f : \{0,1\}^b \to G \uplus \{*\}$ satisfies $G \subseteq f(\{0,1\}^b)$. Here, $f^{-1}(g)$ is the set of *names* of $g \in G$.
- $\mathsf{valid} : \{0,1\}^b \to \{\mathsf{yes}, \mathsf{no}\}$ is 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 such that $\forall x \in f^{-1}(G) : f(\mathsf{inv}(x)) = f(x)^{-1}$.
- $\mathsf{prod} : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}^b$ is such that $\forall x, y \in f^{-1}(G) : f(\mathsf{prod}(x, y)) = f(x)f(y)$.
- $\mathsf{id} : \{0,1\}^b \times \{0,1\}^c \to \{\mathsf{yes}, \mathsf{no}\}$ is such that for all $x \in f^{-1}(G)$: $f(x) = 1$ iff there exists $y \in \{0,1\}^c$ with $\mathsf{id}(x, y) = \mathsf{yes}$ (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 T that is equipped with oracles for the four operations valid, inv, prod, and id. The input for T consists of two unary encoded numbers b and c and some additional problem specific input. In order to determine the behavior of T, it 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 T). Then, given bit strings $x, y \in \{0,1\}^b$ on a special tape of T, it can compute instantaneously in a single computation step the bit string $\mathsf{prod}(x, y)$. Analogous instantaneous computations can be done for the operations valid, inv, and id. We denote with T_B the machine T coupled with the black box 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 T, 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 T, 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: T_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 T_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.

In the rest of the paper we handle black box groups in a slightly more casual way. We 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

In this section, we sketch the proofs for the following two results.

▶ **Theorem 4.1.** *The context-free 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.

Recall the definition of the class CFG(k) from the introduction. Let us first derive some corollaries. Theorem 4.2 directly implies Theorem 1.10. Applied to regular grammars (which are in CFG(1) after bringing them to Chomsky normal form), it yields:

▶ **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 [27]. In the introduction we mentioned that Jerrum proved the PSPACE-completeness of the MGS problem for the case where the number ℓ is given in binary notation. Given permutations $a_1, \ldots, a_n \in S_m$ and a binary encoded number ℓ one can easily construct a context-free grammar for $\{1, a_1, \ldots, a_n\}^\ell \subseteq S_m$. Hence, the PSPACE-hard MGS problem with ℓ given in binary notation reduces to the context-free membership problem for symmetric groups. This reduction shows that Theorem 1.9 still holds if the input grammar is index-m (see the end of the introduction) with m being part of the input in unary notation.

The rest of the section is devoted to the upper bounds in Theorems 4.1 and 4.2. Detailed proofs of all lemmas can be found in the long version [33].

Outline. A straightforward algorithm for the context-free membership problem constructs an exponential-sized pushdown automaton and check the latter for emptiness. This would yield an EXPTIME upper bound. Hence, the difficulty is to achieve this in PSPACE. To this end, we exploit the fact that every subgroup of a black box group can be stored in polynomial space. Since context-free subsets of a finite group G are not necessarily subgroups, we need to find a way to encode information about derivations, as subgroups. We do this as follows. Recall that $\phi_G : G^* \to G$ is the canonical morphism from Section 2. For a context-free grammar $\mathcal{G}$ with terminal alphabet G and a nonterminal A we define

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

This is a subgroup of $G \times \hat{G}$, where $\hat{G}$ is the *dual group* of G: It has the same underlying set as G and if $g \cdot h$ is the product in G then the multiplication $\circ$ in $\hat{G}$ is defined by $g \circ h = h \cdot g$. A black box for G easily yields a black box for $G \times \hat{G}$. Thus, G_A and its subgroups have polynomial-size generating sets. When given generating sets for all G_A, one can check membership in $L(\mathcal{G})$ in PSPACE. Therefore, the idea is to compute G_A by saturating underapproximating subgroups H_A of G_A: Initially, H_A is the trivial subgroup of $G \times \hat{G}$ for each A. At each step, the groups H_A yield underapproximations of the sets $L(A)$ of all $w \in G^*$ derivable from a nonterminal A (as usual, we identify $L(A)$ with $\phi_G(L(A))$). The underapproximation of $L(A)$, in turn, is used to enlarge the subgroups H_A, yielding a better underapproximation of the G_A. Finally, we obtain the entire groups G_A.

The operations Δ and Γ. We fix a finite group G that is only accessed via a black box. We now define the operations Δ and Γ, which turn underapproximations of G_A into underapproximations of $L(A)$ (or vice versa, resp.). 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 T is acyclic if a nonterminal does not occur twice on a path from the root to a leaf of T. The height of an acyclic derivation tree is at most $|N|$. We now define two 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 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 $M(p,d) = M_1 \cdot \ldots \cdot M_m$, in which $\cdot$ 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 set of all $M(p,d)$, where $p = (A_i \to A_{i,0}A_{i,1})_{1\leq i\leq m}$ and $d = (d_i)_{1\leq i\leq m}$ are as above (including the empty sequences). This set H_A is a subgroup of $G \times \hat{G}$. To see this, it suffices to argue that H_A is a monoid. The latter follows from the fact that two pairs of sequences (p,d) and (p',d') of the above form can be composed to a single pair (pp',dd').

In the following, we will speak of NP algorithms with oracles. Here, we mean nondeterministic polynomial-time Turing machines with oracles. However, the oracle can only be queried positively: There is an instruction that succeeds if the oracle answers "yes", but cannot be executed if the oracle would answer "no". This implies that if there is an NP (resp. PSPACE algorithm) for the oracle queries, there exists an NP (resp. PSPACE) algorithm for the entire problem. Likewise, we will use the notion of oracle PSPACE algorithms.

▶ **Lemma 4.4.** *For tuples $t = (L_A)_{A\in N}$, there is an* NP *algorithm for membership to the entries of $\Gamma(t)$, with access to an oracle for the entries of t.*

The main idea for the proof is to represent the entries of $\Gamma(t)$ (which are subgroups of $G \times \hat{G}$) by NFAs over the alphabet $G \times \hat{G}$ (the NFA will not be given explicitly to us) and to guess generating sets for the subgroups in $\Gamma(t)$ using the so-called spanning tree approach. Finally, one checks membership in the entries of $\Gamma(t)$ using Lemma 3.1.

▶ **Lemma 4.5.** *Assume that the input grammar $\mathcal{G}$ is restricted to the class* CFG(k) *for some fixed k. For tuples $s = (H_A)_{A\in N}$ of subgroups $H_A \leq G \times \hat{G}$, there exists an* NP *algorithm for membership to entries in $\Delta(s)$, with access to an oracle for membership to each entry of s.*

To prove this lemma, we guess an acyclic derivation tree of $\mathcal{G}$; by Lemma 2.3 its size is bounded by $2|N|^k$. Moreover, for every internal B-labelled node v (with $B \in N$) we guess a pair $(h_{v,1}, h_{v,2}) \in G \times \hat{G}$ and verify in NP that $(h_{v,1}, h_{v,2}) \in H_A$. Then we evaluate the derivation tree in the right way (following the definition of $\Delta(s)$) and check that we obtain the the input group element that we want to test for membership.

If the input grammar $\mathcal{G}$ is not restricted to a class CFG(k) for some k then we can still prove the following PSPACE-version of Lemma 4.5.

▶ **Lemma 4.6.** *For tuples $s = (H_A)_{A\in N}$ of subgroups $H_A \leq G \times \hat{G}$, there is a* PSPACE *algorithm for membership to $\Delta(s)$, using an oracle for membership to the entries of s.*

For the proof we can no longer guess an acyclic derivation tree, since it may have exponential size. Instead, we guess the derivation tree in an incremental way and store on a stack of size at most $|N|$ (the maximal height of an acyclic derivation tree) the information that is needed in order to evaluate the derivation tree.

The following lemma is a straightforward consequence of Lemma 2.1:

▶ **Lemma 4.7.** *For tuples $s = (H_A)_{A\in N}$ subgroups $H_A \leq G \times \hat{G}$, there exists an* NP *algorithm, with access to an oracle for membership to each entry of s, with the following properties:*

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

▶ **Lemma 4.8.** $\Delta((G_A)_{A\in N}) = (L(A))_{A\in N}$.

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 $s_1 \leq s_2$ if $H_{A,1} \leq H_{A,2}$ for every $A \in N$. Since Γ and Δ are monotone w.r.t. component-wise inclusion, we have $(\Gamma\Delta)^i(s_0) \leq (\Gamma\Delta)^{i+1}(s_0)$ for all i. We can thus define $\lim_{i\to\infty}(\Gamma\Delta)^i(s_0)$.

▶ **Lemma 4.9.** $(G_A)_{A\in N} = \lim\limits_{i\to\infty}(\Gamma\Delta)^i(s_0) = (\Gamma\Delta)^j(s_0)$ *where* $j = 2 \cdot |N| \cdot \lfloor \log_2 |G| \rfloor$.

Let us briefly sketch the proof of this lemma. From the definition of Γ and Δ we easily obtain $(\Gamma\Delta)^i(s_0) \leq (G_A)_{A\in N}$ by induction on $i \geq 0$. To show $(G_A)_{A\in N} \leq \lim_{i\to\infty}(\Gamma\Delta)^i(s_0)$, we define $(H_A)_{A\in N} = \lim_{i\to\infty}(\Gamma\Delta)^i(s_0)$ and take a pair $(g,h) \in G_A$. Hence, there exists a derivation $A \Rightarrow_{\mathcal{G}}^* uAv$ such that $g = \phi_G(u)$ and $h = \phi_G(v)$. One can prove $(g,h) \in H_A$ by an induction on the length of this derivation. This shows the identity $(G_A)_{A\in N} = \lim\limits_{i\to\infty}(\Gamma\Delta)^i(s_0)$.

For the second identity in Lemma 4.8 note that since all G_A are finite groups there is a smallest number $j \geq 0$ 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 $j \leq 2 \cdot |N| \cdot \log_2 |G|$. In each component of the $|N|$-tuples $(\Gamma\Delta)^i(s_0)$ $(0 \leq i \leq j)$ we have a chain of subgroups of $G \times \hat{G}$. By Lagrange's theorem, any chain $\{(1,1)\} = H_0 < H_1 < \cdots < H_{k-1} < H_k \leq G \times \hat{G}$ satisfies $k \leq 2 \cdot \log_2 |G|$. This shows that $j \leq 2 \cdot |N| \cdot \log_2 |G|$.

Proofs of Theorem 4.1 and 4.2. We start with Theorem 4.2. By Lemmas 4.5 and 4.8 it suffices to decide membership to G_A 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, by Lemma 3.1, membership for each $\langle S_A \rangle$ is in NP.

We compute S_A by initializing $S_A = \{(1,1)\}$ for every $A \in N$ and then performing $2 \cdot |N| \cdot \log_2 |G|$ iterations of the following procedure: Suppose 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 4.4 and 4.5 one can decide membership in every entry of the tuple $\Gamma(\Delta((\langle S_A \rangle)_{A\in N}))$ in NP. Finally, by Lemma 4.7 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 4.9 there will be at least one computation path on which after $2 \cdot |N| \cdot \log_2 |G|$ iterations we get a generating set for the entire groups G_A. This concludes Theorem 4.2. The proof of Theorem 4.1 only differs by using Lemma 4.6 instead of Lemma 4.5. ◀

5 Subset sum and knapsack 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. 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 $g = \sum_{1\leq k\leq n} i_k \cdot g_k$?

The proof uses a straightforward reduction from the *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 [23, Problem LO4]. The reduction of X3HS to the problem from Theorem 5.1 can be found in [33, Theorem 5.1]. Since $\mathbb{Z}_3^m \leq S_{3m}$ 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 $\mathbb{Z}_2^m$ (with m part of the input) is equivalent to the subgroup membership problem for $\mathbb{Z}_2^m$ (since every element of $\mathbb{Z}_2^m$ has order two) and therefore can be solved in polynomial time.

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: Recall from the introduction that Luks [36] 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 $g_1, g_2, \ldots, g_k$ 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 following knapsack equation:

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

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.

Let $p > 0$ be an integer. For the rest of the section we write $[p]$ for the cycle $(1, 2, \ldots, p)$ mapping p to 1 and i to $i + 1$ for $1 \leq i \leq p - 1$. The proofs for the following two lemmas are available in the full version [33].

▶ **Lemma 5.4.** *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.*

▶ **Lemma 5.5.** *Let $p, q \in \mathbb{N}$ be primes such that $2 < q < p$ holds. Then*

$$[p]^{-x_2}[q]^{x_1}([p][q])^{x_2} = [q] = [q]^{x_1}[p]^{-x_2}([p][q])^{x_2} \tag{1}$$

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 $x_2 \equiv 1 \bmod p$).

We now come to our main result for the knapsack problem:

▶ **Theorem 5.6.** *The problem 3-knapsack for symmetric groups is* NP*-hard.*

Proof. To prove the theorem we give 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 $A = \{1, \ldots, m\}$ 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 $\alpha : \{1, \ldots, d\} \times \{1, 2, 3\} \to \{1, \ldots, m\}$.

Let $p_1, \ldots, p_m, r_1, \ldots, r_m, q_1, \ldots, q_d$ 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 $1 \leq j \leq m$ hold. Moreover let $P = \max_{1 \leq j \leq m} p_j$. Intuitively, the primes p_j and r_j ($1 \leq j \leq m$) belong to $j \in A$ and the prime q_i ($1 \leq i \leq d$) belongs to the set C_i. We will work in the group

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

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 $g = (v_1, \ldots, v_m, c_1, \ldots c_d)$ and $g_k = (v_{k,1}, \ldots, v_{k,m}, c_{k,1}, \ldots, c_{k,d})$ 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:

$$\begin{array}{rclrclrcl}
v_j &=& ([r_j], & [r_j], & 0, & 0, & 0) & c_i &=& (1, \mathsf{id}, & \mathsf{id}, & \mathsf{id}) \\
v_{1,j} &=& ([r_j], & [p_j]^{-1}, & 1, & 1, & 1) & c_{1,i} &=& (1, [q_i]^{-1}, & [p_{\alpha(i,2)}]^{-1}, & [q_i][p_{\alpha(i,3)}]) \\
v_{2,j} &=& ([p_j]^{-1}, & [r_j], & -1, & 0, & -1) & c_{2,i} &=& (1, [q_i][p_{\alpha(i,1)}], & [q_i]^{-1}, & [p_{\alpha(i,3)}]^{-1}) \\
v_{3,j} &=& ([p_j][r_j], & [p_j][r_j], & 0, & -1, & 0) & c_{3,i} &=& (1, [p_{\alpha(i,1)}]^{-1}, & [q_i][p_{\alpha(i,2)}], & [q_i]^{-1})
\end{array}$$

We claim that there is a subset $A' \subseteq A$ 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 $1 \leq j \leq m$ 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 5.4, $[p_j][r_j]$ 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 (j)–(p):

(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 5.5, the conjunction of (j)–(p) is equivalent to the conjunction of (q)–(u):

(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 $A' \subseteq A$ is such that $|A' \cap C_i| = 1$ for every $1 \leq i \leq d$. Let $\sigma : \{1, \ldots, m\} \to \{0, 1\}$ such that $\sigma(j) = 1$ iff $j \in S'$. 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 $1 \leq j \leq m$,
- $z_k \equiv \sigma(\alpha(i,k)) \bmod q_i$ 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 $A' \subseteq \{1, \ldots, m\}$ such that for every $1 \leq j \leq m$:

- $j \notin S'$ 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
- $j \in S'$ 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 $1 \leq i \leq d$ and $1 \leq k \leq 3$:

- 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 $q_i \geq 3$, 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 $1 \leq i \leq d$ there is exactly one $k \in \{1,2,3\}$ such that $\alpha(i,k) \in A'$. ◀

Theorem 1.6 is an immediate consequence of Corollaries 4.3 and 5.3 and Theorem 5.6.

Theorem 5.6 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. The proof of the following theorem uses a reduction to the membership problem for commutative subgroups of matrix groups [8] and then applies [4, Theorem 1.4]; see [33, Theorem 5.8].

▶ **Theorem 5.7.** *The 2-knapsack problem for symmetric groups belongs to* P.

6 Application to intersection problems

It remains to show Theorems 1.12 and 1.13. We obtain the upper bound in Theorem 1.12 from Theorem 1.10: Let $\mathcal{G}$ be a grammar from CFG(k) and let $\mathcal{A}_i = (Q_i, \Sigma, q_{i,0}, \delta_i, F_i)$ be be a group DFA for $1 \leq i \leq n$. W.l.o.g. assume that the Q_i are pairwise disjoint and let $Q = \bigcup_{1 \leq i \leq n} Q_i$. 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 $L(\mathcal{G}) \cap \bigcap_{1 \leq i \leq n} L(\mathcal{A}_i) \neq \emptyset$ if and only if there exists a permutation $\pi \in L(\mathcal{G}')$ such that $\pi(q_{i,0}) \in F_i$ for every $1 \leq i \leq n$. 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 Theorem 1.12.

The upper bound from Theorem 1.13 can be obtained in the same way from Theorem 1.9. For the lower bounds in Theorems 1.12 and 1.13, a simple reduction from the lower bounds in Theorems 1.10 and 1.9 can be employed; see [33, Section 6] for details.

References

1 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.

2 Mohamed Faouzi Atig and Pierre Ganty. Approximating petri net reachability along context-free traces. In *Proceedings of the 31st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011*, volume 13 of *LIPIcs*, pages 152–163. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.FSTTCS.2011.152.

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 Georg Bachmeier, Michael Luttenberger, and Maximilian Schlund. Finite automata for the sub- and superword closure of CFLs: Descriptional and computational complexity. In *Proceedings of the 9th International Conference on Language and Automata Theory and Applications, LATA 2015*, volume 8977 of *Lecture Notes in Computer Science*, pages 473–485. Springer, 2015. doi:10.1007/978-3-319-15579-1_37.

8 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.

9 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.

10 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.

11 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.

12 Michal Chytil and Burkhard Monien. Caterpillars and context-free languages. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1990*, volume 415 of *Lecture Notes in Computer Science*, pages 70–81. Springer, 1990. doi:10.1007/3-540-52282-4_33.

13 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.

14 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.

15 Javier Esparza, Andreas Gaiser, and Stefan Kiefer. Computing least fixed points of probabilistic systems of polynomials. In *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010*, volume 5 of *LIPIcs*, pages 359–370. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.STACS.2010.2468.

16 Javier Esparza, Pierre Ganty, Stefan Kiefer, and Michael Luttenberger. Parikh's theorem: A simple and direct automaton construction. *Information Processing Letters*, 111(12):614–619, 2011. doi:10.1016/j.ipl.2011.03.019.

17 Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. In *Proceedings of the 25th International Conference on Computer Aided Verification, CAV 2013*, volume 8044 of *Lecture Notes in Computer Science*, pages 124–140. Springer, 2013. doi:10.1007/978-3-642-39799-8_8.

18 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.

19 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.

20 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.

21 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.

22 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.

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

24 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.

25 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:10.1130/0016-7606(1945)56[275:EDOSAT]2.0.CO;2.

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

27 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.

28 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.

29 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.

30 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.

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

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

33 Markus Lohrey, Andreas Rosowski, and Georg Zetzsche. Membership problems in finite groups, 2022. doi:10.48550/ARXIV.2206.11756.

34 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.

35 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.

36 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.

37 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.

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

39 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.

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

41 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.