Theory of Computing Systems
https://doi.org/10.1007/s00224-023-10122-2

Subgroup Membership in GL(2,Z)

Markus Lohrey[1]

Accepted: 14 March 2023

Abstract
It is shown that the subgroup membership problem for a virtually free group can be decided in polynomial time when all group elements are represented by so-called power words, i.e., words of the form $p_1^{z_1} p_2^{z_2} \cdots p_k^{z_k}$. Here the p_i are explicit words over the generating set of the group and all z_i are binary encoded integers. As a corollary, it follows that the subgroup membership problem for the matrix group $\mathsf{GL}(2, \mathbb{Z})$ can be decided in polynomial time when elements of $\mathsf{GL}(2, \mathbb{Z})$ are represented by matrices with binary encoded integers. For the same input representation, it also shown that one can compute in polynomial time the index of a given finitely generated subgroup of $\mathsf{GL}(2, \mathbb{Z})$.

Keywords Algorithmic group theory · Subgroup membership problems · Algorithms for GL(2,Z)

1 Introduction

The subgroup membership problem (also known as the generalized word problem) for a group G asks whether for given group elements $g_0, g_1, \ldots, g_k \in G$, g_0 belongs to the subgroup $\langle g_1, \ldots, g_k \rangle$ generated by $g_1, \ldots, g_k$. To make this a well-defined computational problem, one has to fix an input representation for elements of G. Here, a popular choice is to restrict to finitely generated (f.g. for short) groups. In this case, group elements can be encoded by finite words over a finite set of generators. The subgroup membership problem is one of the best studied problems in computational group theory. Let us survey some important results on subgroup membership problems.

For symmetric groups S_n, Sims [38] has developed a polynomial time algorithm for the uniform variant of the subgroup membership problem, where n is part of the input; see also [3] for efficient parallel algorithms. Here, we only consider the non-uniform subgroup membership problem, where we fix an infinite f.g. group G. For a f.g. free group, the subgroup membership problem can be solved using Nielsen reduction (see

✉ Markus Lohrey
lohrey@eti.uni-siegen.de

[1] Universität Siegen, Siegen, Germany

Published online: 09 May 2023

e.g. [25]); a polynomial time algorithm was found by Avenhaus and Madlener [1]. In fact, in [1] it is shown that the subgroup membership problem for a f.g. free group is P-complete. Another polynomial time algorithm uses Stallings's folding procedure [39]; an almost linear time implementation can be found in [40]. An extension of Stallings's folding for fundamental groups of certain graphs of groups was developed in [17]. 'The folding procedure from [17] can be used to show that subgroup membership is decidable for right-angled Artin groups with a chordal independence graph. Moreover, Friedl and Wilton [12] used the results of [17] in combination with deep results from 3-dimensional topology in order to decide the subgroup membership problem for 3-manifold groups. Other extensions of Stallings's folding and applications to subgroup membership problems can be found in [19, 27, 35]. Using completely different (more algebraic) techniques, the subgroup membership problem has been shown to be decidable for polycyclic groups [2, 26] and f.g. metabelian groups [33, 34]. For f.g. nilpotent groups the subgroup membership problem is complete for the circuit complexity class TC^0 [29].

On the undecidability side, Mihaĭlova [28] has shown that the subgroup membership problem is undecidable for the direct product $F_2 \times F_2$ (where F_2 is the free group of rank two). This implies undecidability of the subgroup membership problem for many other groups, e.g., $\mathsf{SL}(4, \mathbb{Z})$ (the group of 4×4 integer matrices with determinant one) or the 5-strand braid group B_5. Rips [31] constructed hyperbolic groups with an undecidable subgroup membership problem.

Apart from the above mentioned results for free groups [1] (P-completeness) and nilpotent groups [29] (TC^0-completeness) the authors are not aware of other precise complexity results for subgroup membership problems in infinite groups. The completeness results from [1, 29] assume that group elements are represented by finite words over the generators of the free group. In recent years, group theoretic decision problems have also been studied with respect to more succinct representations of group elements. For instance, the so-called compressed word problem, where the input group element is represented by a straight-line program (a context-free grammar that produces exactly one string) has received a lot of attention; see [4, 22] for surveys. For the subgroup membership problem in free groups, Gurevich and Schupp studied in [14] a succinct variant, where input group elements are of the form $a_1^{z_1} a_2^{z_2} \cdots a_k^{z_k}$. Here, the a_i are from a fixed free basis of the free group and the z_i are binary encoded integers. Based on an adaptation of Stallings's folding, they show that this succinct membership problem can be solved in polynomial time. Then, Gurevich and Schupp proceed in [14] by showing that their succinct folding algorithm for free groups can be adapted so that it works for the free product $\mathbb{Z}/2\mathbb{Z} * \mathbb{Z}/3\mathbb{Z}$. The particular interest in this group comes from the fact that it is isomorphic to the modular group $\mathsf{PSL}(2, \mathbb{Z})$, which is the quotient of $\mathsf{SL}(2, \mathbb{Z})$ by $\langle -\mathsf{Id}_2 \rangle \cong \mathbb{Z}/2\mathbb{Z}$ (Id_2 is the 2×2 identity matrix). As an application of the succinct folding algorithm for $\mathbb{Z}/2\mathbb{Z} * \mathbb{Z}/3\mathbb{Z}$, Gurevich and Schupp show that the subgroup membership problem for $\mathsf{PSL}(2, \mathbb{Z})$ is decidable in polynomial time when all matrix entries are encoded in binary notation.

A related result was shown in [29]: the subgroup membership problem for a f.g. nilpotent group can be solved in polynomial time, when group elements are represented by binary encoded Mal'cev coordinates.

The polynomial time algorithm for the succinct membership problem for $\mathbb{Z}/2\mathbb{Z} * \mathbb{Z}/3\mathbb{Z}$ from [14] is tailored towards this group, and it is not clear how to adapt the algorithm to related groups. The latter is the goal of this paper. For this it turns out to be useful to consider a more succinct representation of input elements for free groups. Recall that Gurevich and Schupp use words of the form $a_1^{z_1} a_2^{z_2} \cdots a_k^{z_k}$, where the integers z_i are given in binary notation and the a_i are generators from a free basis. Here, we represent group elements by so-called *power words* which were studied in [23] in the context of group theory. A power word has the form $p_1^{z_1} p_2^{z_2} \cdots p_k^{z_k}$, where as above the integers z_i are given in binary notation but the p_i are arbitrary words over the group generators. In [23] it was shown that the so-called power word problem (does a given power word represent the group identity?) for a f.g. free group F is AC^0-reducible to the ordinary word problem for F (and hence in logspace). In Section 3, we prove that the power-compressed subgroup membership problem (i.e., the subgroup membership problem with all group elements represented by power words) for a free group can be solved in polynomial time by using a folding procedure à la Stallings (Theorem 1). This generalizes the above mentioned result of Gurevich and Schupp. At first sight, the step from power words of the form $a_1^{z_1} a_2^{z_2} \cdots a_k^{z_k}$ (with the a_i generators) to general power words as defined above looks not very spectacular. But apart from the quite technical details, the power-compressed subgroup membership problem has a major advantage over the restricted version of Gurevich and Schupp: we show that if G is a f.g. group and H is a finite index subgroup of G then the power-compressed subgroup membership problem for G is polynomial time reducible to the power-compressed subgroup membership problem for H (Lemma 11). Hence, the power-compressed subgroup membership problem for every f.g. virtually free group (a finite extension of a f.g. free group) can be solved in polynomial time (Corollary 2). This result opens up new applications to matrix group algorithms. It is well-known that the group $\mathsf{GL}(2, \mathbb{Z})$ (the group of all 2×2 integer matrices with determinant ± 1) is f.g. virtually free. Moreover, given a matrix $A \in \mathsf{GL}(2, \mathbb{Z})$ with binary encoded entries one can compute a power word (over a fixed finite generating set of $\mathsf{GL}(2, \mathbb{Z})$) that represents A. Hence, the subgroup membership problem for $\mathsf{GL}(2, \mathbb{Z})$ can be decided in polynomial time when elements of $\mathsf{GL}(2, \mathbb{Z})$ are represented by matrices with binary encoded integers (Corollary 3).

In Section 5 we present another application of our folding procedure for power words: we show that the finite index problem for f.g. subgroups of $\mathsf{GL}(2, \mathbb{Z})$ can be decided in polynomial time, when elements of $\mathsf{GL}(2, \mathbb{Z})$ are represented as matrices with binary encoded integers (Corollary 5). In the finite index problem for a group G the goal is to compute the index (an element of $\mathbb{N} \cup \{\infty\}$) of a given f.g. subgroup of G. The finite index problem has been studied in [16] (for free groups), [27] (amalgamated products of finite groups), [37] (virtually free groups), [18] (quasiconvex subgroups of automatic groups), [9] (direct products of free-abelian and free groups) and [8] (solvable Baumslag-Solitar groups $\mathsf{BS}(1, q)$).

Related Work Related to the subgroup membership problem is the more general *rational subset membership problem*. A rational subset in a group G is given by a finite automaton, where transitions are labelled with elements of G. Such an automaton accepts a subset of G in the natural way. In the rational subset membership problem

for G the input consists of a rational subset $L \subseteq G$ and an element $g \in G$ and the question is, whether $g \in L$. This problem was shown to be decidable for free groups by Benois [6] via an automaton saturation procedure that moreover can be implemented in cubic time [7]. Stallings's folding can be viewed as a special case of Benois's construction.

Rational subset membership problems (and special cases) for matrix groups are a very active research field. Some recent results can be found in [5, 8, 11, 21, 30]. Closest to our work is [5], where it is shown that the identity problem for $\mathsf{SL}(2, \mathbb{Z})$ (does the identity matrix belong to a finitely generated subsemigroup of $\mathsf{SL}(2, \mathbb{Z})$?) and the rational subset membership problem for $\mathsf{PSL}(2, \mathbb{Z})$ are $\mathbb{N}P$-complete (when matrix entries are given in binary notation). For this, the authors of [5] use the ideas of Gurevich and Schupp [14]. In [8, 11], first steps towards $\mathsf{GL}(2, \mathbb{Q})$ are taken: in [11] the authors prove decidability of membership in so-called flat rational subsets of $\mathsf{GL}(2, \mathbb{Q})$, whereas [8] establishes the decidability of the full rational subset membership problem for the Baumslag-Solitar groups $\mathsf{BS}(1, q) < \mathsf{GL}(2, \mathbb{Q})$ with $q \geq 2$.

2 Preliminaries

General Notations For an integer $z \in \mathbb{Z}$ we define its signum as usual: $\mathsf{sign}(0) = 0$, and for $z > 0$, $\mathsf{sign}(z) = 1$ and $\mathsf{sign}(-z) = -1$. As usual, Σ^* denotes the set of all finite words over an alphabet Σ, ε denotes the empty word, and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ is the set of all non-empty words. The length of a word w is denoted by $|w|$. If $w = uv \in \Sigma^*$ then u is called a *prefix* of w and v is called a *suffix* of w. The word $u \in \Sigma^*$ is a *factor* of the word $w \in \Sigma^*$ if $w = sut$ for some $s, t \in \Sigma^*$. At one point, it will be convenient to work with ω-words. An ω-word over the alphabet Γ is an infinite sequence $a_1a_2a_3a_4 \cdots$ with $a_i \in \Gamma$ for all $i \geq 1$. With Γ^ω we denote the set of all ω-word over the alphabet Γ.

Groups We assume some basic background in group theory; see [25] or [32] for more details. For a group G and a subset $A \subseteq G$, we denote with $\langle A \rangle$ the subgroup of G generated by A. It is the set of all products of elements from $A \cup A^{-1}$. We say that A generates G (or A is a generating set for G) if $G = \langle A \rangle$. If $A = A^{-1}$ then A is a symmetric generating set for G. A group G is called finitely generated if it has a finite generating set.

Fix a finite set Σ of symbols and let $\Sigma^{-1} = \{a^{-1} \mid a \in \Sigma\}$ be a set of formal inverses of the symbols in Σ with $\Sigma \cap \Sigma^{-1} = \emptyset$. Let $\Gamma = \Sigma \cup \Sigma^{-1}$. We define an involution on Γ^* by setting $(a^{-1})^{-1} = a$ for $a \in \Sigma$ and $(a_1a_2 \cdots a_k)^{-1} = a_k^{-1} \cdots a_2^{-1}a_1^{-1}$ for $a_1, \ldots, a_k \in \Gamma$. A word $w \in \Gamma^*$ is called *freely reduced* if it neither contains a factor aa^{-1} nor $a^{-1}a$ for $a \in \Sigma$. With $\mathsf{red}(\Gamma^*)$ we denote the set of all freely reduced words. For every word $w \in \Gamma^*$ one obtains a unique freely reduced word that is obtained from w by deleting factors aa^{-1} and $a^{-1}a$ ($a \in \Sigma$) as long as possible. We denote this word with $\mathsf{red}(w)$; it can be computed in linear time from w.

The *free group* generated by Σ, denoted by $F(\Sigma)$, consists of the set $\mathsf{red}(\Gamma^*)$ together with the multiplication $\cdot$ defined by $u \cdot v = \mathsf{red}(uv)$ for $u, v \in \mathsf{red}(\Gamma^*)$. The group identity of $F(\Sigma)$ is the empty word ε. A group G that has a free subgroup of

finite index in G is called *virtually free*. Usually, we identify a (not necessarily freely reduced) word $w \in \Gamma^*$ with the group element $\mathsf{red}(w) \in F(\Sigma)$.

For every group G there exists a free group $F(\Sigma)$ and a surjective homomorphism $\pi : F(\Sigma) \to G$. We then have $G = \langle \pi(\Sigma) \rangle$. If G is finitely generated then we can choose Σ to be finite. In this situation, we also identify Σ with the generating set $\pi(\Sigma)$. We also say that the element $w \in F(\Sigma)$ represents the group element $\pi(w)$. For $u, v \in F(\Sigma)$ we say that $u = v$ in G if $\pi(u) = \pi(v)$. Sometimes, we identify $w \in F(\Sigma)$ (or $w \in (\Sigma \cup \Sigma^{-1})^*$) with the corresponding group element $\pi(w)$.

Fix a f.g. group G together with a surjective morphism $\pi : F(\Sigma) \to G$ with Σ finite. The *subgroup membership problem for* G is the following decision problem:
input: words $w_0, w_1, \ldots, w_n \in F(\Sigma)$.
question: Does $\pi(w_0)$ belong to the subgroup $\langle \pi(w_1), \ldots, \pi(w_n) \rangle \leq G$?
Note that we formulated the subgroup membership problem for G with respect to a fixed surjective morphism $\pi : F(\Sigma) \to G$. In other words, for every such surjective morphism $\pi : F(\Sigma) \to G$, we have another variant of the subgroup membership problem for G. On the other hand, it is easy to see that the computational complexity of the subgroup membership problem for G does not depend on the concrete choice of $\pi : F(\Sigma) \to G$, at least if we only care about complexity classes containing polynomial time (actually, a smaller complexity class such as deterministic logspace would be also fine, but this is not needed for our considerations). To see this take another surjective morphism $\pi' : F(\Theta) \to G$ (with Θ finite as well). Then for every generator $a \in \Sigma$ there is an element $h(a) \in F(\Theta)$ such that $\pi(a) = \pi'(h(a))$ in the group G. The mapping h uniquely extends to a morphism $h : F(\Sigma) \to F(\Theta)$ (this is the crucial property of free groups). We then have $\pi(w_0) \in \langle \pi(w_1), \ldots, \pi(w_n) \rangle$ if and only if $\pi(h(w_0)) \in \langle \pi(h(w_1)), \ldots, \pi(h(w_n)) \rangle$. Since the morphism h can be easily computed in polynomial time (simply replace every symbol a by $h(a)$), this shows that subgroup membership problem for G with respect to $\pi : F(\Sigma) \to G$ is polynomial time reducible to the subgroup membership problem for G with respect to $\pi' : F(\Theta) \to G$. This justifies to not mention the surjective morphism $\pi : F(\Sigma) \to G$ in the subgroup membership problem for G.

In this paper we are interested in a variant of the subgroup membership problem for G where the words $w_0, w_1, \ldots, w_n$ are given in a more succinct way. In the next section, we define this variant.

3 Stallings's Folding for Power-Compressed Words

In this section we present our succinct version of Stallings's folding that was mentioned in the introduction. We start with the definition of power words and power-compressed graphs. These graphs are basically finite automata where the transitions are labelled with power words. We prefer to use the term "graph" instead of "automaton", since the former is more common in the literature on Stallings's folding.

A *power word* over an alphabet Σ is a sequence $(p_1, n_1)(p_2, n_2) \cdots (p_k, n_k)$ of pairs where $p_1, \ldots, p_k \in \Sigma^+$ and $n_1, \ldots, n_k \in \mathbb{N} \setminus \{0\}$. Such a power word represents the ordinary word $p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$ and we usually identify a power word with the word it represents. The difference between the sequence of pairs $(p_1, n_1)(p_2, n_2) \cdots (p_k, n_k)$

and the word $p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$ comes from to the succinctness of descriptions. When a power word is part of the input for a computational problem, we always assume that the exponents n_i are given in binary notation, whereas the words p_i (also called the *periods* of the power word) are written down explicitly by listing all symbols in the words. Therefore, we define the input length $\|w\|$ of the power word $w = (p_1, n_1)(p_2, n_2) \cdots (p_k, n_k)$ as

$$\sum_{i=1}^{k} (|p_i| + \log n_i).$$

On the other hand, the length of the word $p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$ is $\sum_{i=1}^{k} n_i |p_i|$. Therefore, a power word should be seen as a succinct representation of the word it represents.

In the case of a power word over an alphabet $\Gamma = \Sigma \cup \Sigma^{-1}$ we may also allow negative exponents. Of course, p^{-n} stands for $(p^{-1})^n$.

Consider a f.g. group G together with a surjective morphism $\pi : F(\Sigma) \to G$ for Σ finite. The *power-compressed subgroup membership problem* for G is the following problem:
input: Power words $w_0, w_1, \ldots, w_n$ over the alphabet $\Gamma = \Sigma \cup \Sigma^{-1}$.[1]
question: Does $\pi(w_0)$ belong to the subgroup $\langle \pi(w_1), \ldots, \pi(w_n) \rangle \leq G$?
As for the (ordinary) subgroup membership problem, the concrete choice of the surjective morphism $\pi : F(\Sigma) \to G$ does not influence the complexity of the power-compressed subgroup membership problem. The reason is the same as for the subgroup membership problem. The morphism $h : F(\Sigma) \to F(\Theta)$ from the previous section can be also applied to a power words: the power word $w = (p_1, n_1)(p_2, n_2) \cdots (p_k, n_k)$ is mapped to

$$h(w) = (h(p_1), n_1)(h(p_2), n_2) \cdots (h(p_k), n_k).$$

This yields a polynomial time reduction from the power-compressed subgroup membership problem for G with respect to $\pi : F(\Sigma) \to G$ to the power-compressed subgroup membership problem for G with respect to $\pi^{prime} : F(\Theta) \to G$.

The goal of this section is to show that the power-compressed subgroup membership problem can be decided in polynomial time for a f.g. free group. In Section 4 we will extend this result to f.g. virtually free groups.

Our main tool for solving the power-compressed subgroup membership problem for f.g. free groups is an extension of Stallings's folding procedure for power-compressed words. First we need some combinatorial results for words. Fix a finite alphabet Σ with the inverse alphabet Σ^{-1} for the rest of Section 3 and let $\Gamma = \Sigma \cup \Sigma^{-1}$.

[1] We do not assume the w_i to be freely reduced but as explained earlier, we identify every w_i with $\mathsf{red}(w_i) \in F(\Sigma)$. It is not hard to show that one can compute from the power word w_i in polynomial time a power word for $\mathsf{red}(w_i)$, but we do not need this fact (it follows implicitly from our folding algorithm).

3.1 Combinatorics on Words

We fix an arbitrary linear order $<$ on Γ. In order to simplify notation later, it is convenient to require that $a < a^{-1}$ for every $a \in \Sigma$. With $\preceq$ we denote the lexicographic order with respect to $<$. Let $\Omega \subseteq \mathsf{red}(\Gamma^*)$ denote the set of all freely reduced words w such that

- w is non-empty,
- w is cyclically reduced (i.e, w cannot be written as aua^{-1} for $a \in \Gamma$),
- w is primitive (i.e, w cannot be written as u^n for some $n \geq 2$),
- w is lexicographically minimal among all cyclic permutations of w and w^{-1} (i.e., $w \preceq uv$ for all $u, v \in \Gamma^*$ with $vu = w$ or $vu = w^{-1}$).

Note that $\Sigma \subseteq \Omega$ and $\Sigma^{-1} \cap \Omega = \emptyset$ (since $a < a^{-1}$ for $a \in \Sigma$). For every $w \in \Omega$ and $n \in \mathbb{Z}$ we have $w^n \in \mathsf{red}(\Gamma^*)$ (since w is freely reduced and cyclically reduced).

The set Ω was introduced in [23] in order to solve the power word problem (that was mentioned in the introduction) for a free group in logspace. The crucial fact about words in Ω is that if two powers p^x and q^y ($p, q \in \Omega, x, y \in \mathbb{Z}$) have a long enough common factor then $p = q$; see Lemma 1 below.

Example 1 Assume that $a < b < a^{-1} < b^{-1}$. Then the word $w = abab^{-1}$ belongs to Ω. It is clearly freely reduced, cyclically reduced, and primitive. Moreover, the cyclic permutations of w and $w^{-1} = ba^{-1}b^{-1}a^{-1}$ are:

$$\begin{array}{rcl} w & = & a\ \ b\ \ a\ \ b^{-1} \\ & & b\ \ a\ \ b^{-1}\ \ a \\ & & a\ \ b^{-1}\ \ a\ \ b \\ & & b^{-1}\ \ a\ \ b\ \ a \\ w^{-1} & = & b\ \ a^{-1}\ \ b^{-1}\ \ a^{-1} \\ & & a^{-1}\ \ b^{-1}\ \ a^{-1}\ \ b \\ & & b^{-1}\ \ a^{-1}\ \ b\ \ a^{-1} \\ & & a^{-1}\ \ b\ \ a^{-1}\ \ b^{-1} \end{array}$$

Among those words, w is indeed the lexicographically minimal one.

The following lemma can be found in [23, Lemma 11].

Lemma 1 *Let $p, q \in \Omega$ and $x, y \in \mathbb{Z}$. If p^x and q^y have a common factor of length at least $|p| + |q| - 1$ then $p = q$.*

We also need the following statement:

Lemma 2 *If $p \in \Omega$, $u, v \in \Gamma^*$, and $upv = pp$ then $u = \varepsilon$ or $v = \varepsilon$.*

Proof Assume that $upv = pp$ such that $u \neq \varepsilon$ and $v \neq \varepsilon$. We obtain a factorization $p = qr$ such that $q \neq \varepsilon$, $r \neq \varepsilon$ and $p = rq = qr$. Hence, $q, r \in s^*$ for some string $s \in \Gamma^+$ (see e.g. [24, Proposition 1.3.2]), which implies that p is not primitive, a contradiction. □

3.2 Power-Compressed Graphs

A *power-compressed graph* is a tuple $\mathcal{G} = (V, E, \iota, \tau, \lambda, v_0)$, where V is the set of vertices, E is the set of directed edges with $V \cap E = \emptyset$, $\iota\colon E \to V$ maps an edge to its source vertex, $\tau\colon E \to V$ maps an edge to its target vertex, $\lambda\colon E \to \Gamma^+ \times (\mathbb{Z} \setminus \{0\})$ assigns to every edge its label, and $v_0 \in V$ is the so-called *base point*. Moreover, for every edge e such that $\iota(e) = u$, $\tau(e) = v$, and $\lambda(e) = (p, z)$ there is an inverse edge $e^{-1} \neq e$ such that $\iota(e^{-1}) = v$, $\tau(e^{-1}) = u$, $\lambda(e^{-1}) = (p, -z)$, and $(e^{-1})^{-1} = e$. In this paper, V and E will be always finite. Note that we may have edges $e, e' \in E$ with $e \neq e'$, $\iota(e) = \iota(e')$, $\tau(e) = \tau(e')$, and $\lambda(e) = \lambda(e')$.

When we describe a power-compressed graph we often specify for a pair of edges e, e^{-1} only one of them and implicitly assume the existence of its inverse edge. An edge e is called *short* if $\lambda(e) \in \Gamma \times \{-1, 1\}$, otherwise it is called *long*. If $\mathcal{G}$ only contains short edges, then $\mathcal{G}$ is called an *uncompressed graph*, or just *graph*.[2] We define the input length of $\mathcal{G}$ as $|\mathcal{G}| = \sum_{e \in E} \|\lambda(e)\|$ (here, we view $\lambda(e) = (p, z)$ as a power word consisting of a single power).

A *path* in $\mathcal{G}$ is a sequence

$$\rho = [v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1}],$$

where $k \geq 0$, $e_1, \ldots, e_k \in E$, $\iota(e_i) = v_i$ and $\tau(e_i) = v_{i+1}$ for $1 \leq i \leq k$. If $v_i \neq v_j$ for all i, j with $1 \leq i < j \leq k+1$ then ρ is called a *simple path*. If $v_1 = v_{k+1}$ and $k \geq 1$ then ρ is a *cycle*. If $v_i \neq v_j$ for all i, j with $1 \leq i < j \leq k$ and $v_1 = v_{k+1}$ then ρ is a *simple cycle*. Let $\iota(\rho) = v_1$ and $\tau(\rho) = v_{k+1}$. If $\lambda(e_i) = (p_i, z_i)$ then we define $\lambda(\rho)$ as the power word $(p_1, z_1)(p_2, z_2) \cdots (p_k, z_k)$. The path ρ is *oriented* if $\mathsf{sign}(z_i) = \mathsf{signsign}(z_j)$ for all i, j. The path ρ is *without backtracking* if $e_{i+1} \neq e_i^{-1}$ for all $1 \leq i \leq k-1$. The power-compressed graph $\mathcal{G}$ is connected if for all $u, v \in V$ there is a path ρ with $\iota(\rho) = u$ and $\tau(\rho) = v$. The power-compressed graph $\mathcal{G}$ is a tree if it is connected and it does not contain a cycle without backtracking.

In the following, we identify a pair $(p, z) \in \Gamma^+ \times (\mathbb{Z} \setminus \{0\})$ with the power p^z. In particular, in an uncompressed graph every edge is labelled with a symbol from Γ. With a power-compressed graph $\mathcal{G}$ we can associate an uncompressed graph $\mathsf{decompress}(\mathcal{G})$ that is obtained by replacing in $\mathcal{G}$ every p^z-labelled edge e by a path ρ of short edges from $\iota(e)$ to $\tau(e)$ and such that $\lambda(\rho) = p^z$. Moreover, if $\iota(e) \neq \tau(e)$ then ρ is a simple path and if $\iota(e) = \tau(e)$ then ρ is a simple cycle.

A power-compressed graph $\mathcal{G} = (V, E, \iota, \tau, \lambda, v_0)$ can be viewed as a finite automaton over the alphabet Γ, where transition labels are succinct words of the form p^z with z given in binary notation: V is the set of states, an edge e corresponds to a transition from $\iota(e)$ to $\tau(e)$ with label $\lambda(e)$ and v_0 is the unique initial and final state. We denote with $L(\mathcal{G})$ the set of all words $w \in \Gamma^*$ accepted by the automaton $\mathcal{G}$. With $F(\mathcal{G})$ we denote the image of $L(\mathcal{G})$ in the free group $F(\Sigma)$. Since every edge of $\mathcal{G}$ has an inverse edge, it is easy to see that $F(\mathcal{G})$ is a subgroup of $F(\Sigma)$.

[2] Sometimes it is called a dual graph since every every edge has an inverse edge. This definition of graphs is quite common in group theory and topology; see e.g. [36].

3.3 Folding Uncompressed Graphs

Before we continue with power-compressed graphs let us first explain Stallings's folding procedure [39] for uncompressed graphs, which is one of the most powerful techniques for analysing subgroups of free groups; see e.g. [16]. Let $\mathcal{G}$ and $\mathcal{H}$ be two uncompressed graphs as defined in Section 3.2. We say that $\mathcal{G}$ can be *folded* into $\mathcal{H}$ if there exist two edges $e \neq e'$ in $\mathcal{G}$ such that $\iota(e) = \iota(e')$ and $\lambda(e) = \lambda(e')$ and $\mathcal{H}$ is obtained from $\mathcal{G}$ by merging the two vertices $\tau(e)$ and $\tau(e')$ (note that we may have already $\tau(e) = \tau(e')$ in $\mathcal{G}$) into a single vertex and removing the edges e and e^{-1} (this is an arbitrary choice; we could also keep e and e^{-1} and remove e' and e'^{-1}) from the graph. One can easily show that $F(\mathcal{G}) = F(\mathcal{H})$ holds in this situation. Every vertex of $\mathcal{G}$ is mapped to a vertex of $\mathcal{H}$ in the natural way ($\tau(e)$ and $\tau(e')$ are mapped to the same vertex of $\mathcal{H}$). If a graph $\mathcal{G}$ cannot be folded further then we say that $\mathcal{G}$ is *folded*. In this case, $\mathcal{G}$ is a deterministic automaton and $w \in L(\mathcal{G})$ implies $\mathsf{red}(w) \in L(\mathcal{G})$.

Consider now a finite set of words $A = \{w_1, \ldots, w_n\} \subseteq \Gamma^+$ and let $g_i = \mathsf{red}(w_i) \in F(\Sigma)$ be the free group element represented by w_i. We construct a so-called *bouquet graph* $\mathcal{B}(A)$ such that

$$F(\mathcal{B}(A)) = \langle g_1, \ldots, g_n \rangle \leq F(\Sigma)$$

as follows:

- First we define for a non-empty word $w = a_1 a_2 \cdots a_k$ ($a_i \in \Gamma$) the *cycle graph*

$$\mathcal{C}(w) = (\{v_0, \ldots, v_{k-1}\}, \{e_i^{\pm 1} : 1 \leq i \leq k\}, \iota, \tau, v_0),$$

  where $\iota(e_i) = v_{i-1}$, $\lambda(e_i) = a_i$, and $\tau(e_i) = v_{i \bmod k}$ for $1 \leq i \leq k$.
- We then define the bouquet graph $\mathcal{B}(A)$ by taking the disjoint union of the cycle graphs $\mathcal{C}(w_1), \ldots, \mathcal{C}(w_n)$ and then merging the base points of the $\mathcal{C}(w_i)$.

Let $\mathcal{S}(A)$ be the graph obtained by folding $\mathcal{B}(A)$ as long as possible. The final graph of this procedure is in fact unique up to graph isomorphism. The graph $\mathcal{S}(A)$ is sometimes called the Stallings's graph for A. Note that as an automaton, $\mathcal{S}(A)$ is deterministic. The above discussion leads to the following crucial fact (see also [16] for a more detailed discussion):

Lemma 3 *Let A and $g_1, \ldots, g_n$ be as above and let $g \in \mathsf{red}(\Gamma^*)$ be a freely reduced word and hence an element of $F(\Sigma)$. Then g is accepted by $\mathcal{S}(A)$ if and only if $g \in \langle g_1, \ldots, g_n \rangle \leq F(\Sigma)$.*

3.4 Folding Power-Compressed Graphs

Fix a power-compressed graph $\mathcal{G} = (V, E, \iota, \tau, \lambda, v_0)$ for the rest of this section and let P be the set of all words p such that $\lambda(e) = p^z$ for some $e \in E$ and $z \in \mathbb{Z} \setminus \{0\}$. We will refer to the following numbers throughout this section:

- $\alpha := \max\{|p| : p \in P\} \geq 1$,

– $\beta := 2\alpha - 1 \geq 1$,
– $\gamma := 2(\alpha + \beta) \geq 4$.

We say that $\mathcal{G}$ is *normalized* if

– $P \subseteq \Omega$ (where Ω is defined in Section 3.1), and
– for every $e \in E$, if e is long and $\lambda(e) = p^z$ then $|z| \geq \gamma$.

Let E_ℓ be the set of long edges of $\mathcal{G}$.

Lemma 4 *From a given power-compressed graph $\mathcal{G}$ we can compute in polynomial time a normalized power-compressed graph $\mathcal{G}'$ such that $F(\mathcal{G}) = F(\mathcal{G}')$.*

Proof We first modify $\mathcal{G}$ such that for every edge label $\lambda(e) = p^z$ we have $p \in \Omega$. This can be done in polynomial time by [23, Lemma 12] which states that a given power word w over the alphabet Γ can be transformed in polynomial time (in fact, even in logspace) into a power word w' over the alphabet Γ such that (i) all periods of w' belong to Ω and (ii) $w = w'$ in $F(\Sigma)$. We finally replace every long edge e with $\lambda(e) = p^z$ and $|z| < \gamma$ by a simple path (or simple cycle) ρ of short edges such that $\lambda(\rho) = p^z$. □

We say that $\mathcal{G}$ is *weakly folded* if none of the following two conditions A and B hold:
Condition A: There exist two (long or short) edges $e_1 \neq e_2$ such that $\iota(e_1) = \iota(e_2)$, $\lambda(e_1) = p^{z_1}$ and $\lambda(e_2) = p^{z_2}$ for some $p \in P \cup P^{-1}$ and $z_1, z_2 \in \mathbb{N} \setminus \{0\}$.
Condition B: There exist a long edge e with $\lambda(e) = p^z$ and a path ρ consisting of short edges such that $\iota(e) = \iota(\rho)$, $\lambda(\rho) = p$, $p \in P \cup P^{-1}$, and $z \in \mathbb{N} \setminus \{0\}$.
We say that $\mathcal{G}$ is *strongly folded* if the graph $\mathsf{decompress}(\mathcal{G})$ is folded in the sense of Section 3.3. Clearly, if $\mathcal{G}$ is strongly folded then $\mathcal{G}$ is also weakly folded.

Lemma 5 *A given normalized power-compressed graph $\mathcal{G} = (V, E, \iota, \tau, \lambda, v_0)$ can be folded in polynomial time into a normalized and weakly folded power-compressed graph $\mathcal{G}'$. We have $F(\mathcal{G}) = F(\mathcal{G}')$.*

Proof In order to estimate the complexity of our algorithm, we use two termination parameters: the number $|E_\ell|$ of long edges and the total number of edges $|E|$. The algorithm performs a sequence of folding steps that are explained below. In each step, the value $|E_\ell|$ will not increase. If $|E_\ell|$ does not change then $|E|$ will not increase, but if $|E_\ell|$ decreases then $|E|$ may increase by at most $\gamma - 1$. The situation becomes difficult because it may happen that in a folding step neither $|E_\ell|$ nor $|E|$ changes. We distinguish the following three types of folding steps, where $\mathcal{G} = (V, E, \iota, \tau, \lambda, v_0)$ is the power-compressed graph before the folding step and $\mathcal{G}' = (V', E', \iota', \tau', \lambda', v_0')$ is the power-compressed graph after the folding step.
decreasing (p-edge) fold: If condition A holds with $z_1 = z_2$ then we can merge $\tau(e_1)$ and $\tau(e_2)$ into a single vertex (let us call it v) and replace the two edges e_1 and e_2 by a single edge from $\iota(e_1) = \iota(e_2)$ to v with label p^{z_1}.

More formally: If we define $\equiv_V$ to be the smallest (with respect to inclusion) equivalence relation on V with $\tau(e_1) \equiv_V \tau(e_2)$ and $\equiv_E$ to be the smallest equivalence relation on E with $e_1 \equiv_E e_2$ then we can identify V' (respectively, E') with the set

of equivalence classes $\{[v]_{\equiv_V} : v \in V\}$ (respectively, $\{[e]_{\equiv_E} : e \in E\}$). Moreover $\iota'([e]_{\equiv_E}) = [\iota(e)]_{\equiv_V}$, $\tau'([e]_{\equiv_E}) = [\tau(e)]_{\equiv_V}$, $\lambda'([e]_{\equiv_E}) = \lambda(e)$ (all these mappings are well-defined). The surjective mapping μ with $\mu(v) = [v]_{\equiv_V}$ is called the *merging function* associated with the merging step. Note that some of (or all) the vertices $\iota(e_1)$, $\tau(e_1)$, $\tau(e_2)$ can be equal.

nondecreasing (p-edge) fold: If condition A holds with (w.l.o.g.) $z_1 < z_2$ then we can fold the two edges e_1 and e_2 by first setting $V' = V$, $E' = E$, $\tau' = \tau$, $\iota'(e_2) = \tau(e_1)$ and $\lambda'(e_2) = p^{z_2 - z_1}$. On all other arguments, ι' (respectively, λ') coincides with ι (respectively, λ). The resulting graph $\mathcal{G}'$ may be not normalized, namely if e_2 is long (in $\mathcal{G}'$) and $z_2 - z_1 < \gamma$. In this case we replace e_2 by a simple path (or cycle, in case $\iota'(e_2) = \tau'(e_2)$) of fresh short edges from $\iota'(e_2)$ to $\tau'(e_2)$ spelling the word $p^{z_2 - z_1}$. Note that we have $V \subseteq V'$. We define the merging function $\mu : V \to V'$ as the canonical inclusion mapping.

nondecreasing (p-path) fold: If the situation in condition B occurs, then we first set $V' = V$, $E' = E$, $\tau' = \tau$, $\iota'(e) = \tau(\rho)$ and $\lambda'(e) = p^{z-1}$. On all other arguments, ι' (respectively, λ') coincides with ι (respectively, λ). If $z - 1 < \gamma$ then we replace in $\mathcal{G}'$ the edge e by a simple path (or cycle) of short fresh edges spelling the word p^{z-1}. Again we define the merging function $\mu : V \to V'$ as the canonical inclusion mapping.

Note that each of the above folding steps simulates several folding steps in the corresponding uncompressed graph. Figure 1 shows some folding steps:

- (a) to (b): nondecreasing p-path fold (where ρ is the path that is inverse to the red path labelled with ab)
- (b) to (c): decreasing p-edge fold
- (c) to (d): nondecreasing q-edge fold (the q^6-labelled edge coils once around the q^5-labelled loop and the remaining q-labelled edge is replaced by the two short edges labelled with a and c).
- (d) to (e): nondecreasing q-path fold
- (e) to (f): decreasing a-edge fold

Assume we make a sequence of k folding steps, where $\mathcal{G}$ is the initial graph, $\mathcal{G}'$ is the final graph and μ_i ($1 \le i \le k$) is the merging function for the i-th folding step. Then we can define the composition $\mu = \mu_1 \circ \mu_2 \circ \cdots \circ \mu_k$ (where μ_1 is applied first); it maps every vertex v of $\mathcal{G}$ to a vertex $\mu(v)$ of $\mathcal{G}'$. We then say that *vertex v is mapped to vertex $\mu(v)$ during the folding*. For two vertices u, v of $\mathcal{G}$ with $\mu(u) = \mu(v)$ we say that u and v *are merged during the folding*.

Note that every folding step preserves the property of being normalized and that $|E_\ell|$ never increases. Clearly, a decreasing fold decreases $|E|$ (and possibly $|E_\ell|$ in case e_1 and e_2 are long edges). Therefore, we can always perform decreasing folds if possible. A nondecreasing fold can reduce the number of long edges in which case the number of short edges increases by at most $\alpha \cdot (\gamma - 1)$. If a nondecreasing fold does not reduce the number of long edges then both $|E|$ and $|E_\ell|$ stay the same. Hence, the total number of decreasing folds is bounded by $|E| + \alpha \cdot (\gamma - 1) \cdot |E_\ell|$. Bounding the number of nondecreasing folds is not so easy. If we just iteratively fold then we may obtain an exponential running time. In order to ensure termination in polynomial time, we arrange the folding steps as follows: Assume that $P = \{p_1, p_2, \ldots, p_n\}$.

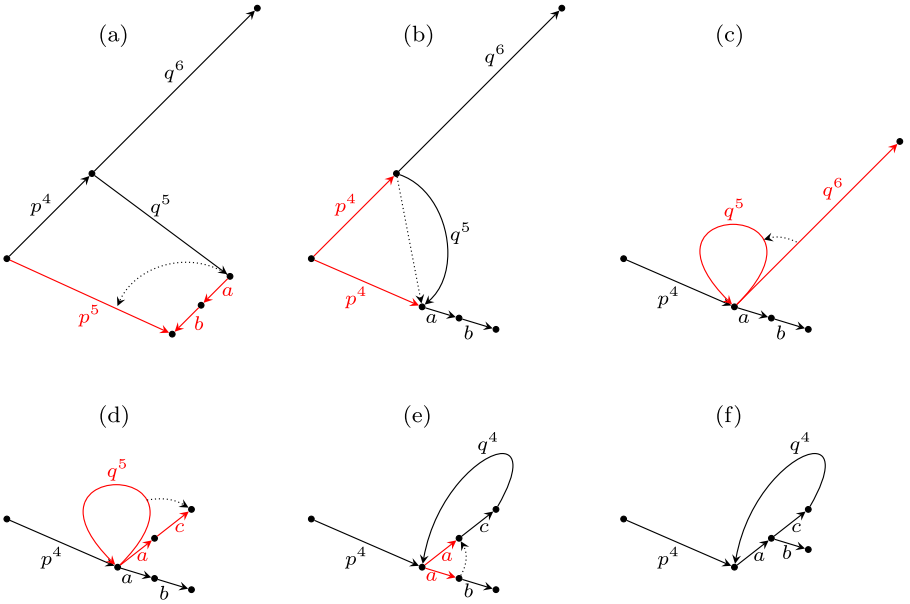

Fig. 1 Some folding steps, where $p = ab \in \Omega$ and $q = ac \in \Omega$. We assume that $\gamma = 4$ and that all inverse edges are implicitly present. The edges involved in the folding steps are red; dotted arrows only indicate the direction of foldings and are not part of the graph. The final graph is weakly folded; in fact it is also strongly folded

Algorithm 1: (The main folding algorithm).

Data: normalized power-compressed graph $\mathcal{G}$
$i := 1$
while *true* **do**
  fold $\mathcal{G}$ with respect to p_i `/* this is explained in the main text */`
  if $\mathcal{G}$ *is weakly folded* **then**
    return $\mathcal{G}$
  else
    $i :=$ smallest j such that $\mathcal{G}$ is not folded with respect to p_j
  end if
end while

We say that the current graph is *folded with respect to* p_j if neither condition A nor condition B holds with $p = p_j$. For the following algorithm it is useful to consider the graph $\mathcal{G}_p$ where the edge set of $\mathcal{G}_p$ contains all long edges from E that are labelled with a power of p. In addition, $\mathcal{G}_p$ contains a p-labelled edge from u to v if $\mathcal{G}$ contains a path ρ of short edges from u to v and such that $\lambda(\rho) = p$ (note that $\mathcal{G}_p$ is in general not normalized). Such an edge should be only viewed as an abbreviation of the corresponding path ρ (which is unique if no decreasing folds are possible in $\mathcal{G}$).

The main structure of the folding algorithm is shown in Algorithm 1. In the following, we always perform decreasing folds when possible without mentioning this explicitly.

We now explain how to fold the current graph $\mathcal{G}$ with respect to some $p = p_i$ (line 3 of Algorithm 1). We consider each connected component of the graph $\mathcal{G}_p$ separately. For the following consideration, we can assume that $\mathcal{G}_p$ is connected. We claim that $\mathcal{G}_p$ can be folded either into a simple oriented path or a simple oriented cycle. Moreover, if $\mathcal{G}_p$ is a tree then it is folded into a simple oriented path. The case that $\mathcal{G}_p$ consists of a single edge is clear. If $\mathcal{G}_p$ has more than one edge then we consider the following cases.

Case 1. $\mathcal{G}_p$ is a tree: Choose an edge e with $\iota(e) = u$ and $\tau(e) = v$ where v is a leaf. Let $\mathcal{G}'$ be the connected graph obtained from $\mathcal{G}_p$ by removing e, e^{-1} and v. By induction, $\mathcal{G}'$ can be folded into a simple oriented path $\rho = [v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1}]$, where w.l.o.g. $\lambda(e_i) = p^{a_i}$ with $a_i > 0$ for all i. Let v_i be the vertex to which $u = \iota(e)$ is mapped during the folding. Assume that $\lambda(e) = p^b$ with $b > 0$ (the case $b < 0$ is analogous). If there exists $j \geq i$ such that $b = a_i + \cdots + a_j$ then nothing has to be done (the vertex v is mapped to v_{j+1} during the folding and the edges e and e^{-1} are removed). If there is no such j then we have to add a vertex to the path: if there is $j \geq i$ such that $a_i + \cdots + a_{j-1} < b < a_i + \cdots + a_j$ then we replace the edge e_j by an edge from v_j to a fresh vertex v' and an edge from v' to v_{j+1}. The label of the first edge is $p^{b-(a_i+\cdots+a_{j-1})}$ and the label of the second edge is $p^{a_i+\cdots+a_j-b}$. If $a_i + \cdots + a_k < b$ then we add an edge from v_{k+1} to the new vertex v' with label $p^{b-(a_i+\cdots+a_k)}$. In both cases the vertex $v = \tau(e)$ is mapped to the new vertex v' during the folding. The resulting graph is an oriented path.

Case 2. $\mathcal{G}_p$ is not a tree. Then we choose an edge e such that $\mathcal{G}' := \mathcal{G}_p \setminus e$ (the graph obtained from $\mathcal{G}_p$ by removing the edges e and e^{-1}) is still connected. By induction, we obtain the following two cases.

Case 2.1. $\mathcal{G}'$ is folded into a simple oriented path

$$\rho = [v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1}],$$

where w.l.o.g. $\lambda(e_i) = p^{a_i}$ with $a_i > 0$ for all i. Let v_i (respectively, v_l) be the vertex to which $\iota(e)$ (respectively, $\tau(e)$) is mapped during the folding and let $\lambda(e) = p^b$ with $b > 0$. We proceed as in case 1. In case there exists $j \geq i$ with $b = a_i + \cdots + a_j$ then we additionally merge v_{j+1} and v_l. We may have already $v_{j+1} = v_l$ in which case we end up with a simple oriented path. Otherwise we obtain a simple oriented path with a simple oriented cycle attached to it. If there is no $j \geq i$ with $b = a_i + \cdots + a_j$ then we add a new vertex v' to the path as in case 1 and merge v' with v_l. This yields again a simple oriented path with a simple oriented cycle attached to it. We then fold the two ends of the simple path onto the cycle (by coiling them around the cycle) and obtain a simple oriented cycle.

Case 2.2. $\mathcal{G}'$ is folded into a simple oriented cycle $\mathcal{C}$. We proceed analogously to case 2.1. We either obtain a single simple oriented cycle or two simple oriented cycles ρ_1 and ρ_2 that are glued together in a single vertex v (to see this, one can first remove an arbitrary edge from the cycle $\mathcal{C}$, which yields a simple oriented path, then carries out the construction from case 2.1 and finally adds the removed edge again). Such a pair of cycles can be replaced by a single cycle as follows: Let $\lambda(\rho_1) = p^{z_1}$ and $\lambda(\rho_2) = p^{z_2}$ with $z_1, z_2 > 0$. Then one can replace the two cycles by a single cycle ρ with $\lambda(\rho) = p^z$, where $z = \gcd(z_1, z_2)$. Folding the cycles into a single cycle actually

corresponds to Euclid's algorithm.[3] Of course, we also have to map the vertices of ρ_1 and ρ_2 into the cycle ρ. For this we start with a p^z-labelled loop at vertex v. If $v' \neq v$ is a vertex belonging to say ρ_1 and the simple path from v to v' on the cycle ρ_1 is labelled with p^y, $y > 0$, then we compute $r := y \bmod z$ and subdivide the loop into an edge from v to v' with label p^r and an edge from v' back to v with label p^{z-r}. We continue in this way with the other vertices on ρ_1 and ρ_2.

Let the power-compressed graph $\mathcal{H}_p$ be the outcome of the above procedure. It is a disjoint union of simple oriented paths and simple oriented cycles and hence folded with respect to p. The running time of the computations in cases 1 and 2 is polynomial in $\|\mathcal{G}_p\|$ and due to the recursion this running time has to be charged for every edge of $\mathcal{G}_p$. Recall that edges labelled with p in $\mathcal{H}_p$ actually correspond to paths of short edges in the original graph $\mathcal{G}$. This concludes the description of line 3 in Algorithm 1.

It remains to argue that we make only polynomially many iterations of the while-loop in Algorithm 1. For this assume that the current graph (call it $\mathcal{G}'$) is folded with respect to p_i and that we fold the graph with respect to some p_j with $j > i$. Let us denote the sequence of folding steps with respect to p_j with $\mathcal{F}_j$ and let $\mathcal{G}''$ be the graph after the execution of $\mathcal{F}_j$. Moreover, assume that $\mathcal{G}''$ is no longer folded with respect to p_i. We argue that this implies that during the execution of $\mathcal{F}_j$ we made progress in the sense that $|E|$ or $|E_\ell|$ decreases. Since $\mathcal{G}'$ is folded with respect to p_i but $\mathcal{G}''$ is not, we must have $\mathcal{G}'_{p_i} \neq \mathcal{G}''_{p_i}$. But this implies that $|E|$ or $|E_\ell|$ must decrease during $\mathcal{F}_j$. Otherwise we only make non-decreasing p_j-edge and p_j-path folds that do not eliminate long edges. Such folds only change the source and target vertices of p_j^z-labelled long edges, which does not modify the graph $\mathcal{G}'_{p_i}$.

Since we have already bounded the number of decreasing folds by $|E| + \alpha \cdot (\gamma - 1) \cdot |E_\ell|$ and the number of long edges never increases, the index i in Algorithm 1 can only decrease a polynomial number of times (more precisely: $|E| + \alpha \cdot \gamma \cdot |E_\ell|$ times). This shows that Algorithm 1 works in polynomial time and concludes the proof of Lemma 5.

It remains to convert a weakly folded power-compressed graph in polynomial time into a strongly folded power-compressed graph. The general idea is the following. Let $\mathcal{G}$ be a normalized and weakly folded power-compressed graph. Recall that $\mathsf{decompress}(\mathcal{G})$ is obtained from $\mathcal{G}$ by replacing every long edge e with label p^z by a simple path (or simple cycle) ρ with $\iota(e) = \iota(\rho)$, $\tau(e) = \tau(\rho)$ and $\lambda(\rho) = p^z$. We show that any sequence of folding steps in $\mathsf{decompress}(\mathcal{G})$ can only affect a short initial and final part of this path ρ. Hence, it suffices to partially decompress $\mathcal{G}$ and then fold short edges as long as possible.

Let us be a bit more precise: We will show that in the above situation, vertices in $\mathsf{decompress}(\mathcal{G})$ that neither belong to the prefix of ρ labelled with $p^{\gamma/2}$ nor to the suffix of ρ labelled with $p^{\gamma/2}$ – later such vertices will be called *protected* – cannot be merged with other vertices during a sequence of folding steps starting in

[3] It is not surprising that at some point we use Euclid's algorithm. For the special case of the free group of rank one, which is isomorphic to $\mathbb{Z}$, the power-compressed subgroup membership problem corresponds to solving a single linear equation with binary encoded integers. The solvability of such an equation can be checked using Euclid's algorithm.

decompress($\mathcal{G}$) (recall the definition of β and $\gamma = 2(\alpha + \beta)$ from the beginning of Section 3.4). For this we need the following simple lemma:

Lemma 6 *Let $\mathcal{H}$ be an uncompressed graph and assume that $\mathcal{H}$ is folded into $\mathcal{H}'$ by a sequence of folding steps. If thereby two vertices u and v of $\mathcal{H}$ are merged to a single vertex of $\mathcal{H}'$, then there must exist a path ρ without backtracking in $\mathcal{H}$ from u to v such that $\lambda(\rho) = \varepsilon$ in $F(\Sigma)$.*[4]

Proof It suffices to find a path ρ from u to v such that $\lambda(\rho) = \varepsilon$ in $F(\Sigma)$. By removing subpaths $[u', e, v', e^{-1}, u']$ from ρ we obtain a path ρ' without backtracking and such that $\lambda(\rho') = \varepsilon$ still holds in $F(\Sigma)$. The existence of such a path can be shown by a straightforward induction over the number of folding steps from $\mathcal{H}$ to $\mathcal{H}'$. Note that if two different vertices v_1 and v_2 of an uncompressed graph are merged in a single folding step, then there exist two different edges $e_1 \neq e_2$ such that $\iota(e_1) = \iota(e_2)$, $\tau(e_1) = v_1$, $\tau(e_2) = v_2$, and $\lambda(e_1) = \lambda(e_2) = a$ for some $a \in \Gamma$. Hence, the path $\rho' = [v_1, e_1^{-1}, \iota(e_1), e_2, v_2]$ satisfies $\lambda(\rho') = a^{-1}a = \varepsilon$ in $F(\Sigma)$. □

Due to Lemma 6 it will suffice to show that a non-empty path without backtracking in decompress($\mathcal{G}$) that starts in a protected vertex is labelled with a word w such that $w \neq \varepsilon$ in $F(\Sigma)$. Since $\mathcal{G}$ is normalized and weakly folded, it will turn out that this word w must be a prefix of an ω-word from the following set $\mathcal{L} \subseteq \Gamma^\omega$: The set $\mathcal{L}$ consists of all ω-words of the form

$$s p_1^{z_1} w_1 p_2^{z_2} w_2 p_3^{z_3} w_3 p_4^{z_4} w_4 \cdots \tag{1}$$

such that the following properties hold for all $i \geq 1$:

- $p_i \in \Omega \cup \Omega^{-1}$,
- s is a suffix of p_1,
- $w_i \in \mathsf{red}(\Gamma^*) \setminus (p_i^{-1}\Gamma^* \cup \Gamma^* p_{i+1}^{-1})$,
- $z_1 \geq \alpha + \beta = \gamma/2$ and $z_i \geq \gamma$ if $i \geq 2$,
- if $w_i = \varepsilon$, then $p_i \neq p_{i+1}^{-1}$.

By our previous discussion, the following lemma is crucial:

Lemma 7 *Every non-empty prefix w of an ω-word from $\mathcal{L}$ satisfies $w \neq \varepsilon$ in $F(\Sigma)$, i.e.,* $\mathsf{red}(w) \neq \varepsilon$.

In order to prove Lemma 7, the following technical lemma turns out be useful. It ensures that in a factor $p_i^{\alpha+\beta} w_i p_{i+1}^{\alpha+\beta}$ in (1) not too much cancellation happens. More precisely, it allows to show that $\mathsf{red}(p_i^{\alpha+\beta} w_i p_{i+1}^{\alpha+\beta})$ starts with p_i and ends with p_{i+1}.

Lemma 8 *Let $p \in \Omega \cup \Omega^{-1}$ and assume that $v \in \Gamma^*$ satisfies one of the following two conditions:*

(i) $v \in \mathsf{red}(\Gamma^*) \setminus p^{-1}\Gamma^*$

[4] Recall that ε is the identity element of $F(\Sigma)$.

(ii) $v = wq'$, *where* $q' \neq \varepsilon$ *is a prefix of* $q^{\alpha+\beta}$ *for some* $q \in \Omega \cup \Omega^{-1}$, $w \in \mathsf{red}(\Gamma^*) \setminus (p^{-1}\Gamma^* \cup \Gamma^* q^{-1})$, *and if* $w = \varepsilon$ *then* $p \neq q^{-1}$.

Then p *is a prefix of* $\mathsf{red}(p^{\alpha+\beta}v)$. *In addition, if case (ii) holds and* $q' = q^{\alpha+\beta}$ *then* q *is a suffix of* $\mathsf{red}(p^{\alpha+\beta}v)$.

Proof If $v \in \mathsf{red}(\Gamma^*) \setminus p^{-1}\Gamma^*$ then p is a prefix of $\mathsf{red}(p^{\alpha+\beta}v)$ (note that $\alpha+\beta \geq 2$).

Now assume that (ii) holds, i.e., $v = wq'$ where $q' \neq \varepsilon$ is a prefix of $q^{\alpha+\beta}$ for some $q \in \Omega \cup \Omega^{-1}$, $w \in \mathsf{red}(\Gamma^*) \setminus (p^{-1}\Gamma^* \cup \Gamma^* q^{-1})$, and if $w = \varepsilon$, then $p \neq q^{-1}$. It suffices to show that p is a prefix of $\mathsf{red}(p^{\alpha+\beta}wq')$. Then by symmetry, q is a suffix of $\mathsf{red}(p^{\alpha+\beta}wq^{\alpha+\beta})$.

Since $p^{\alpha+\beta}$, w and q' are freely reduced, cancellations can only occur at the two borders between $p^{\alpha+\beta}$, w and q'. Let us start to reduce the word $p^{\alpha+\beta}wq'$. Since p^{-1} is not a prefix of w and q^{-1} is not a suffix of w, the reductions at the two borders can only consume $|p|-1 \leq \alpha-1$ symbols from the prefix of w and $|q|-1 \leq \alpha-1$ symbols from the suffix of w. If w is not completely cancelled during the reduction, we obtain a freely reduced word of the form $p^{\alpha+\beta-1}rsq''$, where r is a non-empty prefix of p, s is a non-empty factor of w, and q'' is a possibly empty factor of $q^{\alpha+\beta}$. Thus, p is indeed a prefix of $\mathsf{red}(p^{\alpha+\beta}wq') = p^{\alpha+\beta-1}rsq''$.

Let us now assume that w is completely cancelled during the reduction. Since w is freely reduced, we obtain factorizations $w = u^{-1}t^{-1}$, $p = ru$, and $q = ts$. Moreover, $q' = tq''$ and $p^{\alpha+\beta}wq'$ is reduced to $p^{\alpha+\beta-1}rq''$. Now the word $p^{\alpha+\beta-1}rq''$ can be further reduced at the border between the freely reduced words $p^{\alpha+\beta-1}r$ and q''. If $|q''| < \alpha$ then the reduction can continue for at most $\alpha-1$ steps. Then, the free reduction of $p^{\alpha+\beta}wq'$ consumes from $p^{\alpha+\beta}$ only a suffix of length at most $2(\alpha-1) < \alpha+\beta$. Hence, the first copy of p survives.

We can therefore assume that $|q''| \geq \alpha$. This allows us to write $q'' = sq^k s'$ (recall that $q = ts$ and that the prefix t of q' was cancelled), where $k \geq 0$ and s' is a prefix of q. We distinguish several cases:

- $p \neq q^{-1}$: then by Lemma 1 the reduction of $p^{\alpha+\beta-1}rsq^k s'$ can proceed for at most $|p|+|q|-2 < \beta$ steps.
- $p = q^{-1}$ and $|r| \neq |s|$: then by Lemma 2 the reduction of $p^{\alpha+\beta-1}rsq^k s'$ can proceed for at most $|p|-1 < \alpha \leq \beta$ steps.
- $p = q^{-1}$ and $|r| = |s|$: we obtain $p = ru$ and $p^{-1} = ts$, i.e., $ru = s^{-1}t^{-1}$. Since $|r| = |s| = |s^{-1}|$ we have $r = s^{-1}$ and $u = t^{-1}$. Therefore $w = u^{-1}t^{-1} = u^{-1}u$. Since $w \in \mathsf{red}(\Gamma^*)$, we must have $w = \varepsilon$. Together with $p = q^{-1}$ this yields a contradiction to the assumptions of the lemma.

In total, during the free reduction of $p^{\alpha+\beta}wq'$ only a suffix of $p^{\alpha+\beta}$ of length $< \alpha+\beta$ is cancelled. Hence, the first copy of p is not cancelled. This concludes the proof of the lemma. □

We can now prove Lemma 7.

Proof *(Proof of Lemma 7)* Let w be a non-empty prefix of an ω-word from $\mathcal{L}$. We can write w as

$$w = s\, p_1^{n_1} \prod_{i=1}^{k} (p_i^{\alpha+\beta} w_i p_{i+1}^{\alpha+\beta}\, p_{i+1}^{n_{i+1}})\, t$$

such that $k \geq 0$ and for all i in the proper range we have

- $n_i \geq 0$,
- $p_i \in \Omega \cup \Omega^{-1}$,
- s is a suffix of p_1,
- $w_i \in \mathsf{red}(\Gamma^*) \setminus (p_i^{-1}\Gamma^* \cup \Gamma^* p_{i+1}^{-1})$,
- if $w_i = \varepsilon$, then $p_i \neq p_{i+1}^{-1}$.

Moreover, for the word t one of the following cases must hold:

- t is a prefix of p_{k+1},
- $t = p_{k+1}^{\alpha+\beta} w_{k+1}$ with $w_{k+1} \in \mathsf{red}(\Gamma^*) \setminus p_{k+1}^{-1}\Gamma^*$,
- $t = p_{k+1}^{\alpha+\beta} w_{k+1} v$ with v a non-empty proper prefix of a word $q^{\alpha+\beta}$ for some $q \in \Omega \cup \Omega^{-1}$, $w_{k+1} \in \mathsf{red}(\Gamma^*) \setminus (p_{k+1}^{-1}\Gamma^* \cup \Gamma^* q^{-1})$, and if $w_{k+1} = \varepsilon$ then $p_{k+1} \neq q^{-1}$.

By Lemma 8 every word $\mathsf{red}(p_i^{\alpha+\beta} w_i p_{i+1}^{\alpha+\beta})$ starts with p_i and ends with p_{i+1}. Moreover, $\mathsf{red}(t)$ is a prefix of p_{k+1} or, by Lemma 8, starts with p_{k+1}. This implies that

$$\mathsf{red}(w) = s\, p_1^{n_1} \prod_{i=1}^{k} (\mathsf{red}(p_i^{\alpha+\beta} w_i p_{i+1}^{\alpha+\beta})\, p_{i+1}^{n_{i+1}})\, \mathsf{red}(t) \neq \varepsilon.$$

This concludes the proof of the lemma. □

Consider now a normalized and weakly folded power-compressed graph $\mathcal{G}$. Recall that $\mathsf{decompress}(\mathcal{G})$ is obtained from $\mathcal{G}$ by replacing every long edge e with label p^z by a simple path (or simple cycle) ρ with $\iota(e) = \iota(\rho)$, $\tau(e) = \tau(\rho)$ and $\lambda(e) = \lambda(\rho)$. The vertices of $\mathsf{decompress}(\mathcal{G})$ that are not already in $\mathcal{G}$ (i.e., the inner vertices of the paths that replace the long edges) are also called the *fresh* vertices of $\mathsf{decompress}(\mathcal{G})$. We say that a fresh vertex v of $\mathsf{decompress}(\mathcal{G})$ is *protected* if the following hold: let e be the long edge of $\mathcal{G}$ such that v is an inner vertex of the path ρ that replaces e. Let $\lambda(e) = \lambda(\rho) = p^z$, where $p \in \Omega \cup \Omega^{-1}$ and $z \geq \gamma$. Then the path ρ can be split into two subpaths ρ_1 and ρ_2 such that ρ_1 is a simple path from $\iota(e)$ to v and ρ_2 is a simple path from v to $\tau(e)$. Then v is protected if $p^{\alpha+\beta} = p^{\gamma/2}$ is a prefix of $\lambda(\rho_1)$ and a suffix of $\lambda(\rho_2)$. Intuitively, v is not too close to the two end points $\iota(e)$ and $\tau(e)$.

Lemma 9 *Let $\mathcal{G}$ be a normalized and weakly folded power-compressed graph and let v be a fresh and protected vertex of* $\mathsf{decompress}(\mathcal{G})$. *Let ρ be a non-empty path without backtracking in* $\mathsf{decompress}(\mathcal{G})$ *that starts in v, i.e., $\iota(\rho) = v$. Then $\lambda(\rho) \neq \varepsilon$ in $F(\Sigma)$.*

Proof Let e be the edge in $\mathcal{G}$ such that decompressing e produces v and let ρ' be the simple path/cycle that replaces e. Let $\lambda(e) = \lambda(\rho') = p^z$ with $p \in \Omega \cup \Omega^{-1}$ and $z \geq \gamma$. If ρ is a simple subpath of ρ' then $\lambda(\rho)$ is a non-empty factor of p^z and therefore freely reduced.

Now assume that ρ is not a simple subpath of ρ'. By Lemma 7 it suffices to show that $\lambda(\rho)$ is a non-empty prefix of an ω-word from $\mathcal{L}$. The path ρ has to leave the

path ρ' via $\iota(e)$ or $\tau(e)$. In both cases we can factorize $\lambda(\rho)$ as $\lambda(\rho) = sp_1^{z_1}w$, where $p_1 \in \{p, p^{-1}\}$, $z_1 \in \mathbb{N}$, and $sp_1^{z_1}$ is a suffix of p^z or a suffix of p^{-z}. Moreover, since the vertex v is protected we must have $z_1 \geq \gamma/2$.

The remaining word w can be factorized as

$$w = w_1 p_2^{z_2} w_2 p_3^{z_3} \cdots w_{k-1} p_k^{z_k} w_k t$$

where every $p_i^{z_i}$ is the label of a long edge of $\mathcal{G}$ (hence, $p_i \in \Omega \cup \Omega^{-1}$ and $z_i \geq \gamma$) and every w_i is the label of a path consisting of short edges in $\mathcal{G}$. For the word t, there are two cases:

- $t = \varepsilon$ or
- $t \neq \varepsilon$ arises from long edge e' of $\mathcal{G}$, in which case t is a non-empty prefix of $\lambda(e')$. Hence, t is a non-empty prefix of a word $p_{k+1}^{z_{k+1}}$ for some $p_{k+1} \in \Omega \cup \Omega^{-1}$.

Since $\mathcal{G}$ is weakly folded, the following conditions hold:

- $w_i \in \mathsf{red}(\Gamma^*)$ (since ρ is without backtracking and the situation from condition A on page11 does not occur in $\mathcal{G}$),
- $w_i \notin p_i^{-1}\Gamma^*$ and $w_i \notin \Gamma^* p_{i+1}^{-1}$ if p_{i+1} exists (since the situation from condition B on page 11 does not occur in $\mathcal{G}$),
- if $w_i = \varepsilon$ and p_{i+1} exists, then $p_i \neq p_{i+1}^{-1}$ (since ρ is without backtracking and the situation from condition A on page 11 does not occur in $\mathcal{G}$).

This shows that $\lambda(\rho)$ is a prefix of an ω-word from $\mathcal{L}$. □

Lemma 10 *A given normalized and weakly folded power-compressed graph $\mathcal{G}$ can be folded in polynomial time into a strongly folded power-compressed graph $\mathcal{G}'$. We have $F(\mathcal{G}) = F(\mathcal{G}')$.*

Proof We first construct a power-compressed graph $\mathcal{H}$ by partially decompressing $\mathcal{G}$. Consider a long edge e in $\mathcal{G}$. Let $\iota(e) = u$, $\tau(e) = v$ and $\lambda(e) = p^z$ with $p \in \Omega \cup \Omega^{-1}$ and $z \geq \gamma$. We then replace e by

- a simple path ρ_1 of new short edges going from u to a new vertex u' and such that $\lambda(\rho_1) = p^{\gamma/2} = p^{\alpha+\beta}$,
- a new edge from u' to another new vertex v' with label $p^{z-\gamma}$ (if $z = \gamma$ then $u' = v'$ and the new edge is not needed), and
- a simple path ρ_2 of new short edges going from v' to v and such that $\lambda(\rho_2) = p^{\gamma/2} = p^{\alpha+\beta}$.

The power-compressed graph $\mathcal{H}$ is not necessarily normalized (this is not needed).

We next fold short edges in $\mathcal{H}$ as long as possible. Thereby, the number of edges decreases in each step (folding two short edges is a special case of a decreasing fold). Hence, the process stops after polynomially many folding steps. Let $\mathcal{H}'$ be the resulting power-compressed graph. We show that $\mathcal{H}'$ is strongly folded, which proves the lemma.

Assume the contrary. Then there exist two edges $e_1 \neq e_2$ in $\mathsf{decompress}(\mathcal{H}')$ such that $\iota(e_1) = \iota(e_2)$ and $\lambda(e_1) = \lambda(e_2)$. If e_1 and e_2 are already edges of $\mathcal{H}'$, then e_1 and e_2 are two short edges of $\mathcal{H}'$ that can be folded, which is a contradiction.

Therefore, w.l.o.g. e_1 and $\tau(e_1)$ must arise from decompressing a long edge of $\mathcal{H}'$, i.e., from replacing a long edge in $\mathcal{H}'$ by a simple path of new short edges. Hence $\tau(e_1)$ is a fresh vertex of $\mathsf{decompress}(\mathcal{H}')$ and $\tau(e_1) \neq \tau(e_2)$. We clearly can also fold $\mathsf{decompress}(\mathcal{H})$ (which is the same as $\mathsf{decompress}(\mathcal{G})$) into $\mathsf{decompress}(\mathcal{H}')$. There are vertices $u_1 \neq u_2$ in $\mathsf{decompress}(\mathcal{H})$ such that u_i is mapped to $\tau(e_i)$ while folding $\mathsf{decompress}(\mathcal{H})$ into $\mathsf{decompress}(\mathcal{H}')$. Moreover, also u_1 must be a fresh vertex of $\mathsf{decompress}(\mathcal{H})$. Due to the partial decompression of $\mathcal{G}$ into $\mathcal{H}$, u_1 is a fresh and protected vertex of $\mathsf{decompress}(\mathcal{G})$. By Lemma 6 there must exist a non-empty path ρ in $\mathsf{decompress}(\mathcal{G})$ from u_1 to u_2 without backtracking such that $\lambda(\rho) = \varepsilon$ in $F(\Sigma)$. But this contradicts Lemma 9. □

Lemmas 4, 5 and 10 finally yield the main technical result of Section 3.4:

Corollary 1 *A given power-compressed graph $\mathcal{G}$ can be folded in polynomial time into a strongly folded power-compressed graph $\mathcal{G}'$. We have $F(\mathcal{G}) = F(\mathcal{G}')$.*

3.5 Power-Compressed Subgroup Membership Problem for Free Groups

We can now show the main result of Section 3:

Theorem 1 *The power-compressed subgroup membership problem for a f.g. free group can be solved in polynomial time.*

Proof Let $w_0, w_1, \ldots, w_n$ be the input power words and let $A = \{w_1, \ldots, w_n\}$. We construct from A a power-compressed bouquet graph in the same way as in Section 3.3 for uncompressed graphs: to a non-empty power word $w = p_1^{z_1} p_2^{z_2} \cdots p_k^{z_k}$ we associate the power-compressed cycle graph

$$\mathcal{C}(w) = (\{v_0, \ldots, v_{k-1}\}, \{e_i^{\pm 1} : 1 \leq i \leq k\}, \iota, \tau, v_0),$$

where $\iota(e_i) = v_{i-1}$, $\lambda(e_i) = p_i^{z_i}$, and $\tau(e_i) = v_{i \bmod k}$. We then construct the power-compressed bouquet graph $\mathcal{B}(A)$ by taking the disjoint union of $\mathcal{C}(w_1), \ldots, \mathcal{C}(w_n)$ and then merging their base points. Using Corollary 1 we can fold $\mathcal{B}(A)$ in polynomial time into a strongly folded power-compressed graph $\mathcal{S}(A)$. Let v_0 be its base point. As explained at the end of Section 3.2 we can view $\mathcal{S}(A)$ as a finite automaton, where transitions are labelled with succinct words of the form p^z with z given in binary notation. By Lemma 3, $\mathcal{S}(A)$ accepts a freely reduced word $g \in \mathsf{red}(\Gamma^*) = F(\Sigma)$ if and only if g belongs to the subgroup $\langle \mathsf{red}(w_1), \ldots, \mathsf{red}(w_n) \rangle \leq F(\Sigma)$. Since $\mathcal{S}(A)$ is strongly folded, it is a deterministic automaton in the sense that the labels of two outgoing transitions of a state do not have a non-empty common prefix.

For the rest of the proof it is convenient to switch from power words to straight-line programs. A straight-line program is a context-free grammar $\mathcal{P}$ that produces exactly one word that is denoted with $\mathrm{val}(\mathcal{P})$. By repeated squaring, our given power word w_0 can be easily transformed in polynomial time into an equivalent straight-line program. Moreover, from a given straight-line program $\mathcal{P}$ over the alphabet $\Gamma = \Sigma \cup \Sigma^{-1}$ one can compute in polynomial time a new straight-line program $\mathcal{Q}$ such that $\mathrm{val}(\mathcal{Q}) = \mathsf{red}(\mathrm{val}(\mathcal{P}))$; see [22, Theorem 4.11]. Hence, we can compute in polynomial time

a straight-line program $\mathcal{Q}$ for $\mathsf{red}(w_0)$. The transition labels of the automaton $\mathcal{S}(A)$ can be also transformed into equivalent straight-line programs; such automata with straight-line compressed transition labels were investigated in [15]. It remains to check in polynomial time whether the deterministic automaton $\mathcal{S}(A)$ accepts $\mathrm{val}(\mathcal{Q})$. This is possible in polynomial time by [15, Theorem 1]. □

4 Power-Compressed Subgroup Membership for Virtually Free Groups

A main advantage of the power-compressed subgroup membership problem is that its complexity is preserved under finite index group extensions. The proof of the following lemma follows [13], where it is shown that the complexity of the (ordinary) subgroup membership problem is preserved under finite index group extensions. In order to extend this result to the power-compressed setting, we make use of the conjugate collection process for power words from [23, Theorem 6].

Lemma 11 *Let G be a fixed f.g. group and H a fixed subgroup of finite index in G.*[5] *The power-compressed subgroup membership problem for G is polynomial time reducible to the power-compressed subgroup membership problem for H.*

Proof Using the following standard trick we can assume that H is a normal subgroup of finite index in G: Let N be the intersection of all conjugate subgroups $g^{-1}Hg$. Then N is a normal subgroup of G and has still finite index in G (the latter is a well-known fact). Since $N \leq H$, the power-compressed subgroup membership problem for N is polynomial time reducible to the power-compressed subgroup membership problem for H. Hence, it suffices to show that the power-compressed subgroup membership problem for G is polynomial time reducible to the power-compressed subgroup membership problem for N.

By the above consideration, we can assume that H is a normal subgroup of finite index in G. Let us fix a symmetric generating Θ for H and let $R \subseteq G$ be a (finite) set of coset representatives for H with $1 \in R$. Then $\Sigma := \Theta \cup (R \setminus \{1\})$ generates G. On R we can define the structure of the quotient group G/H by defining $r \cdot r' \in R$ and $\overline{r} \in R$ for $r, r' \in R$ such that $rr' \in H(r \cdot r')$ and $r^{-1} \in H\overline{r}$. Recall that G and H are fixed groups, hence $r \cdot r'$ and $\overline{r}$ can be computed in constant time. In [23, Theorem 6] it is shown that the power word problem for G can be reduced in polynomial time (in fact, in $\mathbb{N}C^1$) to the power word problem for H. The proof shows the following fact:
Fact 1. Given a power word w over the alphabet Σ we can compute in polynomial time a power word w' over the alphabet Θ and $r \in R$ such that $w = w'r$ in G.
Let us now take a finite list of power words $w_0, w_1, \ldots, w_n$ over the alphabet Σ and let $g_i \in G$ be the group element represented by w_i. We want to check whether $g_0 \in A := \langle g_1, \ldots, g_n \rangle$.

First we use Fact 1 and rewrite in polynomial time each power word w_i as $w_i' r_i$ with $w_i' \in \Theta^*$ a power word and $r_i \in R$. Let w_i' represent $g_i' \in H$. By computing the closure of $\{r_1, \overline{r}_1, \ldots, r_n, \overline{r}_n\}$ with respect to the multiplication $\cdot$ on R we obtain

[5] It is well-known that in this situation H must be f.g. as well; see e.g. [32, 1.6.11].

in constant time the set of all coset representatives $r \in R$ such that $Hr \cap A \neq \emptyset$. Let us denote this closure with $V \subseteq R$. Clearly, $1 \in V$. If $r_0 \notin V$ then we have $g_0 = g_0' r_0 \notin A$ and we are done.

Claim 1. In polynomial time we can compute a finite list of generators for $H \cap A$ written as power words over Θ.

For the proof of Claim 1 we follow [13]: we compute a power-compressed graph $\mathcal{G}$ (in the sense of Section 3.2) as follows. All coset representatives from V are vertices of $\mathcal{G}$. Moreover, we add a simple path from $r \in V$ to $r' \in V$ labelled with the power word w_i iff $r \cdot r_i = r'$ $(1 \leq i \leq n)$. The corresponding inverse path (that consists of the inverse edges) is of course labelled with w_i^{-1} and we have $r' \cdot \bar{r}_i = r$. The label of a path from $1 \in V$ back to $1 \in V$ in the graph $\mathcal{G}$ belongs to $\{w_1, w_1^{-1}, \ldots, w_n, w_n^{-1}\}^*$ and hence can be viewed as a power word over the alphabet Σ. As such, it represents an element of the group $H \cap A$.

Fix a spanning tree of $\mathcal{G}$, let E be the set of edges of $\mathcal{G}$ and let $T \subseteq E$ be those edges that belong to the fixed spanning tree. We then obtain a set of generators for $H \cap A$ by taking for every edge $e \in E \setminus T$ the circuit in $\mathcal{G}$ obtained by following the unique simple path in T from 1 to $\iota(e)$, followed by the edge e, followed by the unique simple path in T from $\tau(e)$ back to 1. Let $x_e \in \{w_1, w_1^{-1}, \ldots, w_n, w_n^{-1}\}^*$ be the label of this circuit. Every x_e represents an element of $H \cap A$ and the set of all these elements (for $e \in E \setminus T$) is a generating set of $H \cap A$; see [13] for details. Moreover, every x_e can be written as a power word over the alphabet Σ of polynomial length. Using Fact 1 we can rewrite this power word in polynomial time into $x_e' r_e$ where x_e' is a power word over the alphabet Θ and $r_e \in R$. But since x_e represents an element of H, we must have $r_e = 1$. Hence the power words x_e' represent a generating set of $H \cap A$.

Now we can finish the proof of the lemma. We use the graph $\mathcal{G}$ defined above. Since $r_0 \in V$, there is a path from 1 to r_0. Let $x \in \{w_1, w_1^{-1}, \ldots, w_n, w_n^{-1}\}^*$ be the label of this path. It is a power word over Σ and by Fact 1, x can be rewritten into the form yr for a power word y over Θ and $r \in R$. Clearly, we must have $r = r_0$. In the group G we have $g_0 x^{-1} = g_0' r_0 r_0^{-1} y^{-1} = g_0' y^{-1}$ (here, the words x and y are identified with the corresponding elements of G). Note that $g_0' y^{-1}$ is represented by the power word $w_0' y^{-1}$ over the alphabet Θ. Since the word x represents an element of A we have $g_0 \in A$ if and only if $g_0 x^{-1} \in A$ if and only if $g_0' y^{-1} \in A$ if and only if $g_0' y^{-1} \in H \cap A$. The latter is an instance of the power-compressed subgroup membership problem for H since we have power-compressed generators for $H \cap A$. This concludes the proof. □

From Theorem 1 and Lemma 11 we immediately obtain the following corollary:

Corollary 2 *The power-compressed subgroup membership problem for a fixed f.g. virtually free group can be solved in polynomial time.*

The group $\mathsf{GL}(2, \mathbb{Z})$ consists of all (2×2)-matrices over the integers with determinant -1 or 1. It is a well-known example of a f.g. virtually free group [36]. We are interested in the situation where group elements of $\mathsf{GL}(2, \mathbb{Z})$ are represented by 4-tuples of binary encoded integers. Testing whether such a 4-tuple belongs to $\mathsf{GL}(2, \mathbb{Z})$ is of course possible in polynomial time.

Lemma 12 *From a given matrix $A \in \mathsf{GL}(2,\mathbb{Z})$ with binary encoded entries one can compute in polynomial time a power word over a fixed finite generating set of $\mathsf{GL}(2,\mathbb{Z})$, which evaluates to the matrix A.*

Proof For the group $\mathsf{SL}(2,\mathbb{Z})$ of all (2×2)-matrices over the integers with determinant 1 the result is shown in [14], see also [10, Proposition 15.4]. Now, $\mathsf{SL}(2,\mathbb{Z})$ is a normal subgroup of index two in $\mathsf{GL}(2,\mathbb{Z})$. Fix an arbitrary matrix $B \in \mathsf{GL}(2,\mathbb{Z})$ with determinant -1. Given a matrix $A \in \mathsf{GL}(2,\mathbb{Z})$ with binary encoded entries and determinant -1 we first compute the matrix $AB^{-1} \in \mathsf{SL}(2,\mathbb{Z})$. Using [14] we can compute in polynomial time a power word w for AB^{-1}. Hence, wB (where B is taken as an additional generator) is a power word for A. □

Corollary 3 *The subgroup membership problem for $\mathsf{GL}(2,\mathbb{Z})$ can be solved in polynomial time when matrix entries are given in binary encoding.*

Proof Since $\mathsf{GL}(2,\mathbb{Z})$ is f.g. virtually free, the power-compressed subgroup membership problem for $\mathsf{GL}(2,\mathbb{Z})$ can be solved in polynomial time by Corollary 2. By Lemma 12 this shows Corollary 3. □

5 The Finite Index Problem

For a f.g. group G with the finite generating set Σ we define the *finite index problem* as follows, where as usual $\Gamma = \Sigma \cup \Sigma^{-1}$.
input: words $w_1, \ldots, w_n$ over the alphabet Γ.
output: the index (an element of $\mathbb{N} \cup \{\infty\}$) of the subgroup $\langle g_1, \ldots, g_n\rangle \leq G$, where g_i is the group element represented by w_i.
If the words $w_1, \ldots, w_n$ are represented as power words then we speak of the *power-compressed finite index problem*.

Theorem 2 *The power-compressed finite index problem for a f.g. free group can be solved in polynomial time.*

Proof We use the following criterion from [16]. Consider a f.g. subgroup $\langle A\rangle \leq F(\Sigma)$ with $A \subseteq \Gamma^+$ finite. We compute the folded (uncompressed) graph $\mathcal{S}(A)$ as described before Lemma 3. Let v_0 be the base point of $\mathcal{S}(A)$. We define the *core* of $\mathcal{S}(A)$, denoted with $\mathsf{core}(\mathcal{S}(A))$, by removing from $\mathcal{S}(A)$ all vertices $v \neq v_0$ and edges e that do not belong to a cycle without backtracking that contains the base point v_0; see also [16, Definitions 3.5 and 5.3]. This means that we delete as long as possible vertices $v \neq v_0$ for which there is a unique edge e with $\tau(e) = v$ together with the edges e and e^{-1}.[6] For instance, the core of the graph in Figure 1(f), where the origin of the p^4-labelled edge is the base point, is obtained by removing the b-labelled edge and its target vertex. On the other hand, if the base point is the target of the p^4-labelled edge, then the core is obtained by removing the b-labelled edge and its target vertex as well as the p^4-labelled edge and its source vertex.

[6] If $A \subseteq \mathsf{red}(\Gamma^*)$ then $\mathcal{S}(A)$ is its own core; see the proof of [16, Proposition 3.8].

Assume that $\mathsf{core}(\mathcal{S}(A)) = (V, E, \iota, \tau, \lambda, v_0)$. It is shown in [16, Proposition 8.3] that $\langle A \rangle$ has finite index in $F(\Sigma)$ if and only if $\mathsf{core}(\mathcal{S}(A))$ is a *Γ-regular* graph in the sense that for every $v \in V$ and every $a \in \Gamma$ there is a (necessarily unique) edge $e \in E$ with $\iota(e) = v$ and $\lambda(e) = a$.

Let us now consider the case where the words in A are power words. We then compute in polynomial time the power-compressed and strongly folded graph $\mathcal{S}(A)$ as described in the proof of Theorem 1. We compute the core of this graph in the same way as above by removing vertices $v \neq v_0$ of degree one together with the adjacent edges; let us denote this core with $\mathcal{C}(A)$. It remains to check whether $\mathsf{decompress}(\mathcal{C}(A))$ is Γ-regular. The case $|\Sigma| = 1$ is trivial. So, let us assume that $|\Sigma| \geq 2$ (and hence $|\Gamma| \geq 4$). But then $\mathsf{decompress}(\mathcal{C}(A))$ has vertices of degree two if $\mathcal{C}(A)$ contains long edges. To see this note that since $\mathcal{C}(A)$ is strongly folded, every edge label p^z of a long edge must be a freely reduced word. Hence, if $\mathcal{C}(A)$ contains long edges then $\langle A \rangle$ has infinite index in $F(\Sigma)$. On the other hand, if $\mathcal{C}(A)$ only contains short edges, then we can directly apply the above criterion from [16] in order to compute the index $[F(\Sigma) : \langle A \rangle]$. □

Lemma 13 *Let G be a fixed f.g. group and H a fixed subgroup of finite index in G (thus, H must be f.g. as well). The power-compressed finite index problem for G is polynomial time reducible to the power-compressed finite index problem for H.*

Proof As in the proof of Lemma 13 we can restrict to the case where H is a normal subgroup of G. Otherwise we define N as the intersection of all conjugate subgroups $g^{-1}Hg$. Then N is a normal subgroup of finite index in G. Let $d = [H : N]$ be the index of N in H, which is a fixed constant. Assume that we can reduce in polynomial time the power-compressed finite index problem for G to the power-compressed finite index problem for N. The power-compressed finite index problem for N is polynomial time reducible to the power-compressed finite index problem for H (for a f.g. subgroup $A \leq N$ we have $[N : A] = [H : A]/d$). Hence, the power-compressed finite index problem for G is polynomial time reducible to the power-compressed finite index problem for H.

Let us now assume that H is normal subgroup of G and assume that we can solve the power-compressed index problem for H in polynomial time. We take over all notations from the proof of Lemma 13. Hence, $A \leq G$ is the f.g. subgroup whose index $[G : A]$ we want to compute and the generators of A are given as power words.

Recall that $V \subseteq R$ is the set of coset representatives of H such that $r \in V$ if and only if $Hr \cap A \neq \emptyset$. We claim that $|V| = [A : H \cap A]$. To see this choose for every $r \in V$ an arbitrary element $g_r \in Hr \cap A$. We then have $Hr \cap A = (H \cap A)g_r \neq \emptyset$. Moreover, the sets $Hr \cap A$ $(r \in V)$ are pairwise disjoint; hence also the sets $(H \cap A)g_r$ $(r \in V)$ are pairwise disjoint. Since $A = \bigcup_{r \in V} Hr \cap A = \bigcup_{r \in V} (H \cap A)g_r$, it follows that the sets $(H \cap A)g_r$ $(r \in V)$ are the right cosets of $H \cap A$ in A. This shows that $|V| = [A : H \cap A] < \infty$. In particular, we can compute $[A : H \cap A] < \infty$ in constant time.

By Claim 1 from the proof of Lemma 13 we can compute in polynomial time a finite list of generators for $H \cap A$ written as power words. Hence, we can compute the

index $[H : H \cap A]$ in polynomial time. We now have

$$[G : H \cap A] = [G : H] \cdot [H : H \cap A] = [G : A] \cdot [A : H \cap A]$$

and thus

$$[G : A] = \frac{[G : H] \cdot [H : H \cap A]}{[A : H \cap A]}.$$

Here, $[G : H]$ is a fixed constant, and $[H : H \cap A]$ and $[A : H \cap A]$ can be computed in polynomial time. Hence, $[G : A]$ can be computed in polynomial time. □

Theorem 2 and Lemma 13 yield:

Corollary 4 *The power-compressed finite index problem for a fixed f.g. virtually free group can be solved in polynomial time.*

With Lemma 12 and Corollary 4 we finally obtain:

Corollary 5 *The finite index problem for* $\mathsf{GL}(2, \mathbb{Z})$ *can be solved in polynomial time when matrix entries are given in binary encoding.*

6 Future Work

There is not much hope to generalize Corollary 3 to higher dimensions. For $\mathsf{SL}(4, \mathbb{Z})$ the subgroup membership problem is undecidable and decidability of the subgroup membership problem for $\mathsf{SL}(3, \mathbb{Z})$ is a long standing open problem [20].

A more feasible problem concerns the rational subset membership problem for free groups when transitions are labelled with power words. It is easy to see that this problem is NP-hard (reduction from subset sum) and we conjecture that it belongs to NP. As a consequence this would show that the rational subset membership problem for $\mathsf{GL}(2, \mathbb{Z})$ is NP-complete when the transitions of the automaton are labelled with binary encoded matrices. The corresponding statement for $\mathsf{PSL}(2, \mathbb{Z})$ was shown in [5].

Another interesting problem is whether the subgroup membership problem for a free group can be solved in polynomial time, when all group elements are represented by straight-line programs (which can be more succinct than power words). One might try to show this using an adaptation of Stallings's folding, but controlling the size of the graph during the folding seems to be more difficult when the transition labels are represented by straight-line programs instead of power words.

Acknowledgements This work was funded by the DFG project LO 748/12-1.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Avenhaus, J., Madlener, K.: The Nielsen reduction and P-complete problems in free groups. Theo Comput Sci **32**(1–2), 61–76 (1984). https://doi.org/10.1016/0304-3975(84)90024-0
2. Avenhaus J, Wißmann D (1989) Using rewriting techniques to solve the generalized word problem in polycyclic groups. In : Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation, ISSAC 1989, pp. 322–337. ACM Press. https://doi.org/10.1145/74540.74579
3. Babai L, Luks EM, Seress V Á (1987) Permutation groups in NC. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC 1987, pp. 409–420. ACM. https://doi.org/10.1145/28395.28439
4. Bassino F, Kapovich I, Lohrey M, Miasnikov A, Nicaud C, Nikolaev A, Rivin I, Shpilrain V, Ushakov A , Weil P (2020) Compression techniques in group theory. In: Complexity and Randomness in Group Theory. De Gruyter, chapter 4. https://doi.org/10.1515/9783110667028
5. Bell PC, Hirvensalo M, Potapov I (2017) The identity problem for matrix semigroups in $\mathsf{SL}_2(\mathbb{Z})$ is NP-complete. In: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017. SIAM, pp. 187–206. https://doi.org/10.1137/1.9781611974782.13
6. Benois, M.: Parties rationnelles du groupe libre. Comptes rendus hebdomadaires des séances de l'Académie des sciences, Séries A **269**, 1188–1190 (1969)
7. Benois, M., Sakarovitch, J.: On the complexity of some extended word problems defined by cancellation rules. Inf Process Lett **23**(6), 281–287 (1986). https://doi.org/10.1016/0020-0190(86)90087-6
8. Cadilhac M, Chistikov D, Zetzsche G (2020) Rational subsets of Baumslag-Solitar groups. In: Proceedings of the 47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, vol 168 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 116:1–116:16. https://doi.org/10.4230/LIPIcs.ICALP.2020.116
9. Delgado, J., Ventura, E.: Algorithmic problems for free-abelian times free groups. J Algebra **391**, 256–283 (2013). https://doi.org/10.1016/j.jalgebra.2013.04.033
10. Diekert V, Elder M (2017) Solutions of twisted word equations, EDT0L languages, and context-free groups. Proceedings of the 44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, vol 80 of LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 96:1–96:14. https://doi.org/10.4230/LIPIcs.ICALP.2017.96
11. Diekert V, Potapov I, Semukhin P (2020) Decidability of membership problems for flat rational subsets of GL(2,Q) and singular matrices. In: Proceedings of the 45th international symposium on symbolic and algebraic computation, ISSAC 2020. ACM, pp. 122–129. https://doi.org/10.1145/3373207.3404038
12. Friedl, S., Wilton, H.: The membership problem for 3-manifold groups is solvable. Algebraic Geometric Topology **16**(4), 1827–1850 (2016). https://doi.org/10.2140/agt.2016.16.1827
13. Grunschlag Z (1999) Algorithms in Geometric Group Theory. PhD thesis, University of California at Berkley
14. Gurevich, Y., Schupp, P.E.: Membership problem for the modular group. SIAM J Comput **37**(2), 425–459 (2007). https://doi.org/10.1137/050643295
15. Jeż, A.: The complexity of compressed membership problems for finite automata. Theory of Computing Systems **55**(4), 685–718 (2014). https://doi.org/10.1007/s00224-013-9443-6
16. Kapovich, I., Myasnikov, A.: Stallings foldings and subgroups of free groups. J Algebra **248**(2), 608–668 (2002). https://doi.org/10.1006/jabr.2001.9033
17. Kapovich, I., Weidmann, R., Myasnikov, A.: Foldings, graphs of groups and the membership problem. Int J Algebra Comput **15**(1), 95–128 (2005). https://doi.org/10.1142/S021819670500213X
18. Kharlampovich, O., Miasnikov, A., Weil, P.: Stallings graphs for quasi-convex subgroups. J Algebra **488**, 442–483 (2017). https://doi.org/10.1016/j.jalgebra.2017.05.037
19. Kharlampovich OG, Myasnikov AG, Remeslennikov VN, Serbin DE (2004) Subgroups of fully residually free groups: algorithmic problems. In: group theory, statistics, and cryptography, vol 360 of contemporary mathematics. AMS, Providence, RI, pp. 63–101. https://doi.org/10.1090/conm/360

20. Khukhro EI, Mazurov VD (2020) Unsolved problems in group theory. The Kourovka notebook. CoRR, arXiv:1401.0300v19, Problem 12.50
21. Ko S-K, Niskanen R, Potapov I (2018) On the identity problem for the special linear group and the Heisenberg group. In: Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, volume 107 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 132:1–132:15. https://doi.org/10.4230/LIPIcs.ICALP.2018.132
22. Lohrey M (2014) The Compressed Word Problem for Groups. SpringerBriefs in Mathematics. Springer. https://doi.org/10.1007/978-1-4939-0748-9
23. Lohrey M, Weiß A (2019) The power word problem. In: Proceedings of the 44th international symposium on mathematical foundations of computer science, MFCS 2019, volume 138 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pages 43:1–43:15. https://doi.org/10.4230/LIPIcs.MFCS.2019.43
24. Lothaire, M.: Combinatorics on Words. Cambridge University Press (1997). https://doi.org/10.1017/CBO9780511566097
25. Lyndon, R.C., Schupp, P.E.: Combinatorial Group Theory. Springer (1977). https://doi.org/10.1007/978-3-642-61896-3
26. Mal'cev AI (1983) On homomorphisms onto finite groups. American Math Soc Trans, Series 2 119: 67–79. Translation from Ivanov. Gos. Ped. Inst. Ucen. Zap. 18 (1958) 49–60. https://doi.org/10.1090/trans2/119
27. Markus-Epstein, L.: Stallings foldings and subgroups of amalgams of finite groups. Int J Algebra Comput **17**(8), 1493–1535 (2007). https://doi.org/10.1142/S0218196707003846
28. Mihaĭlova KA (1966) The occurrence problem for direct products of groups. Math. USSR Sbornik 70:241–251, English translation
29. AlG, Myasnikov, Weiß, A.: Parallel complexity for nilpotent groups. Int J Algebra Comput **32**(5), 895–928 (2022). https://doi.org/10.1142/S0218196722500382
30. Potapov I, Semukhin P (2017) Decidability of the membership problem for 2×2 integer matrices. In: Proceedings of the 28th annual ACM-SIAM symposium on discrete algorithms, SODA 2017. SIAM, pp. 170–186. https://doi.org/10.1137/1.9781611974782.12
31. Rips, E.: Subgroups of small cancellation groups. Bulletin London Math Soc **4**, 45–47 (1982). https://doi.org/10.1112/blms/14.1.45
32. Robinson DJS (1996) A Course in the Theory of Groups, 2nd edn. Springer. https://doi.org/10.1007/978-1-4419-8594-1
33. Romanovskiĭ NS (1974) Some algorithmic problems for solvable groups. Algebra Logic 13:13–16, English translation. https://doi.org/10.1007/BF01462922
34. Romanovskiĭ NS (1980) The occurrence problem for extensions of Abelian groups by nilpotent groups. Siberian Math J 21:273–276, English translation. https://doi.org/10.1007/BF00968275
35. Schupp, P.E.: Coxeter groups, 2-completion, perimeter reduction and subgroup separability. Geomet Dedicata **96**, 179–198 (2003). https://doi.org/10.1023/A:1022155823425
36. Serre, J.-P.: Trees. Springer (1980). https://doi.org/10.1007/978-3-642-61856-7
37. Silva, P.V., Soler-Escrivà, X., Ventura, E.: Finite automata for Schreier graphs of virtually free groups. J Group Theory **19**(1), 25–54 (2016). https://doi.org/10.1515/jgth-2015-0028
38. Sims CC (1971) Computation with permutation groups. In: Proceedings of the 2nd ACM symposium on symbolic and algebraic manipulation, SYMSAC 1971. Association for Computing Machinery, p 23–28. https://doi.org/10.1145/800204.806264
39. Stallings, J.R.: Topology of finite graphs. Invent Math **71**(3), 551–565 (1983). https://doi.org/10.1007/BF02095993
40. Touikan, Nicholas W. M.: A fast algorithm for Stallings' folding process. Int J Algebra Comput **16**(6), 1031–1045 (2006). https://doi.org/10.1142/S0218196706003396