Subgroup Membership in GL(2,Z)

Markus Lohrey
Universität Siegen, Germany

Abstract

It is shown that the subgroup membership problem for a virtually free group can be decided in polynomial time where 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 all matrix entries are given in binary notation.

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

Keywords and phrases free groups, virtually free groups, subgroup membership, matrix groups

Digital Object Identifier 10.4230/LIPIcs.STACS.2021.51

Funding *Markus Lohrey*: Funded by DFG project LO 748/12-1.

1 Introduction

The subgroup membership problem (aka 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 of group elements. 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 [33] has developed a polynomial time algorithm for the uniform variant of the subgroup membership problem, where n is part of the input. In this paper, we always consider non-uniform subgroup membership problems, where we consider a fixed infinite f.g. group G. For a f.g. free group, the subgroup membership problem can be solved using Nielsen reduction (see e.g. [23]); 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 [34]; an almost linear time implementation can be found in [35]. An extension of Stallings's folding for fundamental groups of certain graphs of groups was developed in [15]. The folding procedure from [15] can be used to show that subgroup membership is decidable for right-angled Artin groups with a chordal independence graph. Moreover, Friedl and Wilton [10] used the results of [15] 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 [16, 25, 31]. Using completely different (more algebraic) techniques, the subgroup membership problem has been shown to be decidable for polycyclic groups [2, 24] and f.g. metabelian groups [29, 30].

On the undecidability side, Mihaĭlova [26] 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 [28] constructed hyperbolic groups with an undecidable subgroup membership problem.

Apart from the above mentioned result of Avenhaus and Madlener [1] for free groups, the authors are not aware of other precise complexity results for subgroup membership problems in infinite groups. The P-completeness result for free groups from [1] assumes that group elements are represented by finite words over the generators of the free group. In recent years, group theoretic decision problems have been also 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 so-called straight-line program (a context-free grammar that produces exactly one string) has received a lot of attention; see [3, 20] for a survey. For the subgroup membership problem in free groups, Gurevich and Schupp studied in [12] 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 [12] 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.

The polynomial time algorithm for the succinct membership problem for $\mathbb{Z}/2\mathbb{Z} * \mathbb{Z}/3\mathbb{Z}$ from [12] 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 turnes 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 [21] 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 [21] 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 this paper, 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 12). 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 13). 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. 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 $\mathsf{GL}(2,\mathbb{Z})$ with binary encoded matrix entries can be decided in polynomial time.

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 [5] via an automata saturation procedure that moreover can be implemented in cubic time [6]. 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 [4, 7, 9, 18, 27]. Closest to our work is [4], 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 NP-complete (when matrix entries are given in binary notation). For this, the authors of [4] use the ideas of Gurevich and Schupp [12]. In [7, 9], first steps towards $\mathsf{GL}(2,\mathbb{Q})$ are taken: in [9] the authors prove decidability of membership in so-called flat rational subsets of $\mathsf{GL}(2,\mathbb{Q})$, whereas [7] 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 $|w|$. The word $u \in \Sigma^*$ is a *factor* of the word $w \in \Sigma^*$ if $w = sut$ for some $s, t \in \Sigma^*$.

Groups. 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 only consider *finitely generated (f.g.) groups* G, for which there is a finite set $A \subseteq G$ such that $G = \langle A \rangle$; such a set A is called a *finite generating set* for G. If $A = A^{-1}$ then we say that A is a *finite symmetric generating set* for G. Clearly, G is f.g. if and only if there exists a finite alphabet Γ and a surjective monoid homomorphism $\pi \colon \Gamma^* \to G$. We also say that the word $w \in \Gamma^*$ represents the group element $\pi(w)$. For words $u, v \in \Gamma^*$ we say that $u = v$ in G if $\pi(u) = \pi(v)$. Sometimes, we also identify a word $w \in \Gamma^*$ with the corresponding group element $\pi(w)$.

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 $(a_1 a_2 \cdots a_k)^{-1} = a_k^{-1} \cdots a_2^{-1} a_1^{-1}$ for $a_1, \ldots, a_k \in \Gamma$. A word $w \in \Gamma^*$ is called *freely reduced* or *irreducible* 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 irreducible words. For every word $w \in \Gamma^*$ one obtains a unique irreducible 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 $\mathsf{red}(w)$.

The *free group* generated by Σ, $F(\Sigma)$ for short, can identified with the set $\mathsf{red}(\Gamma^*)$ together with the multiplication defined by $u \cdot v = \mathsf{red}(uv)$ for $u, v \in \mathsf{red}(\Gamma^*)$. A group G that has a free subgroup of finite index in G is called *virtually free*.

3 Stallings's folding for power-compressed words

In this section we present our succinct version of Stallings's folding. 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 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 $p_1, \dots, p_n \in \Sigma^+$ and $n_1, \dots, n_k \in \mathbb{N} \setminus \{0\}$. 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. In the case of an alphabet $\Gamma = \Sigma \cup \Sigma^{-1}$ we may also allow negative exponents in a power word. Of course, p^{-n} stands for $(p^{-1})^n$. 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 $\sum_{i=1}^{k} |p_i| + \log n_i$. A power word should be seen as a succinct representation of the word it represents.

Consider a f.g. group G with the finite generating set Σ. The *power-compressed subgroup membership problem* for G is the following problem:

input: Power words $w_0, w_1, \dots, w_n$ over the alphabet $\Sigma \cup \Sigma^{-1}$.
question: Does g_0 belong to the subgroup $\langle g_1, \dots, g_n \rangle \leq G$, where g_i is the group element represented by w_i?

The concrete choice of the finite generating set Σ has no influence on the complexity of the power-compressed subgroup membership problem: If Θ is another finite generating set, then every generator $a \in \Sigma \cup \Sigma^{-1}$ can be expressed as word $w_a \in (\Theta \cup \Theta^{-1})^*$. Hence, from a power word w over $\Sigma \cup \Sigma^{-1}$ one can compute a power word w' over $\Theta \cup \Theta^{-1}$ such that w and w' represent the same group element. For this, one only has to apply the homomorphism $a \mapsto w_a$ to all periods p of the power word w, which can be done in TC^0 [19].

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 $\Gamma = \Sigma \cup \Sigma^{-1}$.

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 irreducible 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 $a \in \Sigma$). Since $w \in \Omega$ is irreducible and cyclically reduced, also every power w^n is irreducible. The following lemma can be found in [21, Lemma 11].

▶ **Lemma 1.** *Let $p, q \in \Omega$, $x, y \in \mathbb{Z}$ and let u be a factor of p^x and v a factor of q^y. If $uv = 1$ in $F(\Sigma)$ and $|u| = |v| \geq |p| + |q| - 1$, then $p = q$.*

We also need the following statement:

▶ **Lemma 2.** *If $p \in \Omega$, $u, v \in \Gamma^*$, $x \in \{-1, 1\}$ and $up^xv = pp$ then $x = 1$ and $u = \varepsilon$ or $v = \varepsilon$.*

Proof. First 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. [22, Proposition 1.3.2]), which implies that p is not primitive, a contradiction.

Now assume that $up^{-1}v = pp$. If $u = \varepsilon$ or $v = \varepsilon$ then $p = p^{-1}$ which implies $p \notin \mathsf{red}(R)$. If $u \neq \varepsilon$ and $v \neq \varepsilon$ then we obtain a factorization $p = qr$ such that $q \neq \varepsilon$, $r \neq \varepsilon$ and $p^{-1} = rq$. Hence, $qr = p = q^{-1}r^{-1}$, which implies $q = q^{-1}$ and $r = r^{-1}$. But the latter implies $q, r \notin \mathsf{red}(R)$ and hence $p \notin \mathsf{red}(R)$, 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 edges ($V \cap E = \emptyset$), $\iota \colon E \to V$ 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 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 $(e^{-1})^{-1} = e$. 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*. 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 $e_1, \ldots, e_k \in E$, $\iota(e_i) = v_i$ 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 $v_1 = v_{k+1}$ 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 $\mathsf{sign}(z_i) = \mathsf{sign}(z_j)$ for all i, j. The path ρ is *without backtracking* if $e_{i+1} \neq e_i^{-1}$ for all $1 \leq i \leq k-1$.

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 $\mathcal{G} = (V, E, \iota, \tau, \lambda, v_0)$ should be viewed as an 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 $F(\Sigma)$.

3.3 Folding uncompressed graphs

Before we continue with power-compressed graphs let us first explain Stallings's folding procedure [34] for uncompressed graphs, which is one of the most powerful techniques for subgroups of free groups. 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 $\mathsf{red}(w) \in L(\mathcal{G})$.

To a given finite set of words $A = \{w_1, \ldots, w_n\} \subseteq \Gamma^+$ we can associate a so-called bouquet graph $\mathcal{B}(A)$ such that $F(\mathcal{B}(A)) = \langle g_1 \ldots, g_n\rangle \leq F(\Sigma)$, where $g_i = \mathsf{red}(w_i) \in F(\Sigma)$ is the free group element represented by w_i): to a non-empty word $w = a_1 a_2 \cdots a_k$, where $a_i \in \Gamma$, we associate the cycle graph $\mathcal{C}(w) = (\{v_0, \ldots, v_{k-1}\}, \{e_i^{\pm 1} \colon 1 \leq i \leq k\}, \iota, \tau, v_0)$, where $\iota(e_i) = v_{i-1}$, $\lambda(e_i) = a_i$, and $\tau(e_i) = v_{i \bmod k}$ for $1 \leq i \leq k$. Then we define the bouquet graph $\mathcal{B}(A)$ by merging in the disjoint union of the cycle graphs $\mathcal{C}(w_i)$ the base points.

Let $\mathcal{S}(A)$ be the graph obtained by folding $\mathcal{B}(A)$ as long as possible (the outcome 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 [14] for a more detailed discussion):

▶ **Lemma 3.** *Let $g \in \mathsf{red}(\Gamma^*)$ be an irreducible 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 $z \in \mathbb{Z} \setminus \{0\}$. Let us define the following numbers:

- $\alpha := \max\{|p| \colon p \in P\} \geq 1$,
- $\beta := 2\alpha - 1 \geq 1$,
- $\gamma := 2(\alpha + \beta) \geq 4$.

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 [21, 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 holds:

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 $p \in \Omega$ and $z_1, z_2 \in \mathbb{Z} \setminus \{0\}$ with $\mathsf{sign}(z_1) = \mathsf{sign}(z_2)$.

Condition B: There exist a long edge e with $\lambda(e) = p^z$ and a path ρ consisting of short edges such that $\iota(e) = \iota(\rho)$, $\lambda(\rho) = p^x$, $x \in \{-1, 1\}$, and $\mathsf{sign}(x) = \mathsf{sign}(z)$.

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, $\{[e]_{\equiv_E} : e \in V\}$). 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-x}. Note that after this modification we have $V \subseteq V'$ and $E \subseteq E'$. 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 $\lambda'(e) = p^{z-x}$. On all other arguments, ι' (respectively, λ') coincides with ι (respectively, λ). If in the resulting graph $\mathcal{G}'$, e is long and $|z - x| < \gamma$ then we replace the edge e by a simple path (or cycle) of short fresh edges spelling the word p^{z-x}. 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.

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 ($1 \leq i \leq k$) 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

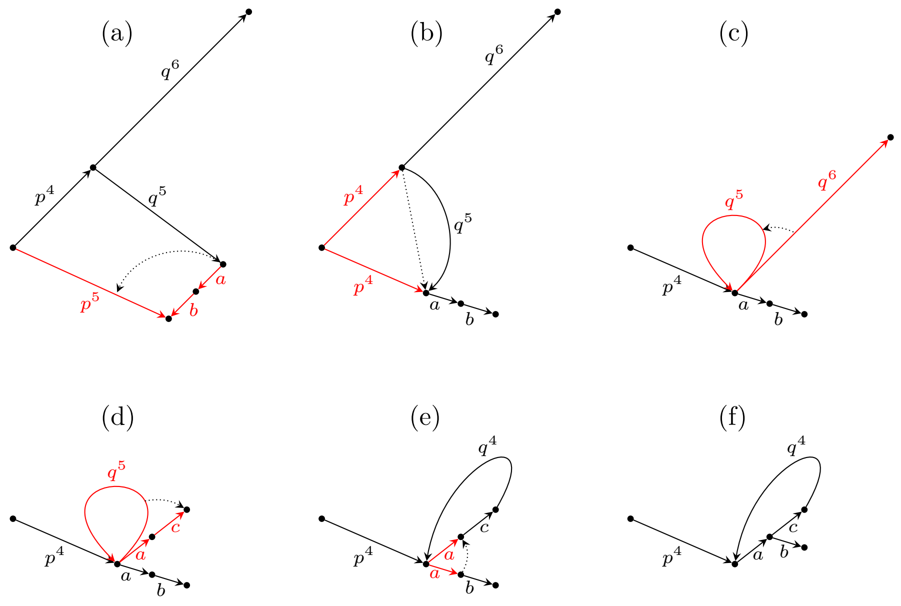

Figure 1 Some folding steps, where $p = ab \in \Omega$ and $q = ac \in \Omega$. We assume that $\gamma = 4$ and that all inverse edges are implicitly present. The edges involved in the folding steps are red; dotted arrows only indicate the direction of foldings and are not part of the graph.

- (a) to (b): nondecreasing p-path fold
- (b) to (c): decreasing p-edge fold
- (c) to (d): nondecreasing q-edge folds (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

The finally graph is weakly folded.

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 preserve the property of being normalized. Clearly, a decreasing fold does not increase $|E_\ell|$ but 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 $|E| + \alpha \cdot \gamma \cdot |E_\ell|$. 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 $P = \{p_1, p_2, \ldots, p_n\}$. We say that the current graph if *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^1-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 $\mathcal{G}$).

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
end
```

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). 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.

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.

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. 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). If there is no such j then we add a new vertex v' to the path as in case 1 and merge v' with v_l. In both cases we get a simple oriented path to which a simple oriented cycle is attached. 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 $\lambda(\rho) = z := \gcd(z_1, z_2)$ (folding the cycles into a single cycle actually corresponds to Euclid's algorithm). 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 $\mathcal{H}_p$ 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 case 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^1 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 either $|E|$ 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 $|E| + \alpha \cdot \gamma \cdot |E_\ell|$ and the number of long edges never increases, the index i in Algorithm 1 can only decrease a polynomial number of times (more precisely: $|E| + (\alpha \cdot \gamma + 1) \cdot |E_\ell|$ times). ◀

It remains to convert a weakly folded power-compressed graph in polynomial time into a strongly folded power-compressed graph. For this, we need several lemmas.

▶ **Lemma 6.** *Let $\mathcal{G}$ be an uncompressed graph and assume that $\mathcal{G}$ is folded into $\mathcal{G}'$ by a sequence of folding steps. If thereby two vertices u and v of $\mathcal{G}$ are merged to a single vertex of $\mathcal{G}'$, then there must exist a path ρ without backtracking in $\mathcal{G}$ from u to v such that $\lambda(\rho) = 1$ in $F(\Sigma)$.*

Proof. The lemma can be shown by a straightforward induction over the number of folding steps from $\mathcal{G}$ to $\mathcal{G}'$. 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 $\rho = [v_1, e_1^{-1}, \iota(e_1), e_2, v_2]$ is without backtracking and satisfies $\lambda(\rho) = a^{-1}a = 1$ in $F(\Sigma)$. ◀

▶ **Lemma 7.** *Consider a word $p^y w q^z \in \Gamma^*$ such that the following hold, where $a = \mathsf{sign}(y)$ and $b = \mathsf{sign}(z)$:*

- $p, q \in P$,
- $w \in \mathsf{red}(\Gamma^*)$,
- $|y| = |z| = \alpha + \beta = \gamma/2 \geq 2$,

- *if $w = \varepsilon$, then $p \neq q$ or $a = b$,*
- *p^{-a} is not a prefix of w and q^{-b} is not a suffix of w.*

Then $\mathsf{red}(p^y w q^z)$ starts with a non-empty prefix of p^a and ends with a non-empty suffix of q^b.

Proof. Since p^y, w and q^z are irreducible, reductions can only occur at the two borders between p^y, w and q^z. Let us start to reduce the word $p^y w q^z$. Since p^{-a} is not a prefix of w and q^{-b} is not a suffix of w, the reductions at the two borders can only consume $|p| - 1 < \alpha$ symbols from the prefix of w and $|q| - 1 < \alpha$ symbols from the suffix of w. If w is not completely cancelled during the reduction, we obtain an irreducible word of the form $p^{y-a} r s t q^{z-b}$, where r is a prefix of p^a, t is a suffix of q^b and s is a non-empty factor of w. The conclusion of the lemma clearly holds in this case.

Let us now assume that w is completely cancelled during the reduction. Since w is irreducible, we obtain factorizations $w = u^{-1} v^{-1}$, $p^a = ru$, and $q^b = vs$. Moreover, $p^y w q^z$ is reduced to $p^{y-a} r s q^{z-b}$. We distinguish several cases:

- $p \neq q$: then the reduction of $p^{y-a} r s q^{z-b}$ can proceed for at most $|p| + |q| - 2 < \beta$ steps (otherwise we obtain a contradiction to Lemma 1).
- $p = q$ and $|r| \neq |s|$: then the reduction of $p^{y-a} r s q^{z-b}$ can proceed for at most $|p| - 1 < \alpha$ steps (otherwise we obtain a contradiction to Lemma 2).
- $p = q$, $|r| = |s|$, and $a = b$: then the reduction of $p^{y-a} r s q^{z-b}$ can proceed for at most $|r| \leq \alpha$ steps (otherwise p would be not cyclically reduced).
- $p = q$, $|r| = |s|$, and $a = -b$: w.l.o.g. assume that $y > 0$ and $z < 0$. We obtain $p = ru$ and $p^{-1} = vs$, i.e., $ru = s^{-1} v^{-1}$. Since $|r| = |s| = |s^{-1}|$ we have $r = s^{-1}$ and $u = v^{-1}$. Therefore $w = u^{-1} v^{-1} = u^{-1} u$. Since $w \in \mathsf{red}(\Gamma^*)$, we must have $w = \varepsilon$. But we have excluded this case in the assumptions of the lemma.

In total, the reduction of $p^y w q^z$ consumes strictly less than $\alpha + \beta = \gamma/2$ symbols from p^y as well as from q^z. Hence, $\mathsf{red}(p^y w q^z)$ starts with a non-empty prefix of p^a and ends with a non-empty suffix of q^b. ◀

▶ **Lemma 8.** *Let $w = s p_1^{z_1} w_1 p_2^{z_2} w_2 \cdots p_{k-1}^{z_{k-1}} w_{k-1} p_k^{z_k} t$ be a word with $k \geq 2$ and let $a_i = \mathsf{sign}(z_i)$. Assume that the following conditions hold:*

- *$p_1, \ldots, p_k \in P$,*
- *$z_1, \ldots, z_k \in \mathbb{Z}$,*
- *$|z_1|, |z_k| \geq \alpha + \beta = \gamma/2$,*
- *$|z_2|, \ldots, |z_{k-1}| \geq \gamma$,*
- *$w_1, \ldots, w_{k-1} \in \mathsf{red}(\Gamma^*)$,*
- *s is a suffix of $p_1^{a_1}$, t is a prefix of $p_k^{a_k}$,*
- *if $w_i = \varepsilon$, then $p_i \neq p_{i+1}$ or $a_i \neq -a_{i+1}$ $(1 \leq i \leq k-1)$,*
- *$p_i^{-a_i}$ is not a prefix of w_i and $p_{i+1}^{-a_{i+1}}$ is not a suffix of w_i $(1 \leq i \leq k-1)$.*

Then $w \neq 1$ in $F(\Sigma)$, i.e., $\mathsf{red}(w) \neq \varepsilon$.

Proof. For $1 \leq i \leq k$ let c_i be such that $|c_i| = \gamma/2$ and $\mathsf{sign}(c_i) = a_i$. Let $u_i = p_i^{c_i} w_i p_{i+1}^{c_{i+1}}$ for $1 \leq i \leq k-1$. We can reduce $w = s p_1^{z_1 - c_1} u_1 p_2^{z_2 - 2c_2} u_2 \cdots p_{k-1}^{z_{k-1} - 2c_{k-1}} u_{k-1} p_k^{z_k - c_k} t$ to

$$w' := s p_1^{z_1 - c_1} \,\mathsf{red}(u_1)\, p_2^{z_2 - 2c_2} \,\mathsf{red}(u_2) \cdots p_{k-1}^{z_{k-1} - 2c_{k-1}} \,\mathsf{red}(u_{k-1})\, p_k^{z_k - c_k} t.$$

By Lemma 7, $\mathsf{red}(u_i)$ starts with a non-empty prefix of $p_i^{a_i}$ and ends with a non-empty suffix of $p_{i+1}^{a_{i+1}}$. This implies that w' is irreducible and non-empty, which shows $w \neq 1$ in $F(\Sigma)$. ◀

We also need the following variant of Lemma 8.

▶ **Lemma 9.** *Let $w = sp_1^{z_1}w_1p_2^{z_2}w_2 \cdots p_k^{z_k}w_k$ be a word with $k \geq 1$ and let $a_i = \mathsf{sign}(z_i)$. Assume that the following conditions hold:*

- $p_1, \dots, p_k \in P$,
- $z_1, \dots, z_k \in \mathbb{Z}$,
- $|z_1| \geq \alpha + \beta = \gamma/2$,
- $|z_2|, \dots, |z_k| \geq \gamma$,
- $w_1, \dots, w_k \in \mathsf{red}(\Gamma^*)$,
- *s is a suffix of $p_1^{a_1}$,*
- *if $w_i = \varepsilon$, then $p_i \neq p_{i+1}$ or $a_i \neq -a_{i+1}$ $(1 \leq i \leq k-1)$,*
- *$p_i^{-a_i}$ is not a prefix of w_i $(1 \leq i \leq k)$ and $p_{i+1}^{-a_{i+1}}$ is not a suffix of w_i $(1 \leq i \leq k-1)$.*

Then $w \neq 1$ in $F(\Sigma)$, i.e., $\mathsf{red}(w) \neq \varepsilon$.

Proof. The proof is almost the same as for Lemma 8. For $1 \leq i \leq k$ let c_i be such that $|c_i| = \gamma/2$ and $\mathsf{sign}(c_i) = a_i$. Let $u_i = p_i^{c_i}w_ip_{i+1}^{c_{i+1}}$ for $1 \leq i \leq k-1$ and $u_k = p_k^{a_k}w_k$. We can reduce $w = sp_1^{z_1-c_1}u_1p_2^{z_2-2c_2}u_2 \cdots p_{k-1}^{z_{k-1}-2c_{k-1}}u_{k-1}p_k^{z_k-c_k-1}u_k$ to

$$w' := sp_1^{z_1-c_1}\,\mathsf{red}(u_1)\,p_2^{z_2-2c_2}\,\mathsf{red}(u_2)\cdots p_{k-1}^{z_{k-1}-2c_{k-1}}\,\mathsf{red}(u_{k-1})\,p_k^{z_k-c_k-1}\,\mathsf{red}(u_k).$$

By Lemma 7, every $\mathsf{red}(u_i)$ with $1 \leq i \leq k-1$ starts with a non-empty prefix of $p_i^{a_i}$ and ends with a non-empty suffix of $p_{i+1}^{a_{i+1}}$. Moreover, $\mathsf{red}(u_k)$ starts with a non-empty prefix of $p_k^{a_k}$ (since $p_k^{-a_k}$ is not a prefix of w_k). This implies that w' is irreducible and non-empty, which shows $w \neq 1$ in $F(\Sigma)$. ◀

▶ **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 $\lambda(e) = p^z$. W.l.o.g. assume that $z > 0$. Since $\mathcal{G}$ is normalized, we have $z \geq \gamma$. 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 present), and
- a simple path ρ_2 of new short edges going from v' to v and such that $\lambda(\rho_2) = p^{\gamma/2} = p^{\alpha+\beta}$.

We then fold $\mathcal{H}$ as long as possible. By Lemmas 6, 8 and 9 we can thereby only fold short edges. In other words: if $\mathcal{H}' = \mathsf{decompress}(\mathcal{H})$ (which is the same as $\mathsf{decompress}(\mathcal{G})$) then a vertex of $\mathcal{H}'$ that arises from decompressing a long edge of $\mathcal{H}$ cannot be merged with another vertex during the folding. To see this, assume the contrary: let u be a vertex of $\mathcal{H}'$ that arises from decompressing a long edge of $\mathcal{H}$ and that is merged with a vertex $v \neq u$ during the folding. By Lemma 6 there must exist a path ρ in $\mathcal{H}'$ from u to v without backtracking such that $\lambda(\rho) = 1$ in $F(\Sigma)$. But since $\mathcal{G}$ is weakly folded the word $\lambda(\rho)$ must be a word w as considered in Lemma 8 (if also v arises from decompressing a long edge of $\mathcal{H}$) or Lemma 9 (if v is already a vertex in $\mathcal{H}$). The w_i in Lemma 8 (resp., Lemma 9) correspond to the maximal subpaths of ρ consisting of short edges and the $p_i^{z_i}$ correspond to the long edges on the path). Hence, $\lambda(\rho) \neq 1$ in $F(\Sigma)$ which is a contradiction.

By the above consideration, if we fold short edges in $\mathcal{H}$ as long as possible we obtain a strongly folded graph $\mathcal{G}'$ which proves the lemma. ◀

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

▶ **Corollary 11.** *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 12.** *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. We construct from $w_1, \ldots, w_n$ 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 $\mathcal{C}(w) = (\{v_0, \ldots, v_{k-1}\}, \{e_i^{\pm 1} \colon 1 \le i \le k\}, \iota, \tau, v_0)$, 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 $\mathcal{B}$ by taking the disjoint union of $\mathcal{C}(w_1), \ldots, \mathcal{C}(w_n)$ and then merging their base points. Using Corollary 11 we can fold $\mathcal{B}$ in polynomial time into a strongly folded power-compressed graph $\mathcal{G}$. Let v_0 be its base point. As explained at the end of Section 3.2 we can view $\mathcal{G}$ 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, $\mathcal{G}$ accepts an irreducible word $g \in \mathsf{red}(\Gamma^*)$ if and only if g represents an element from $\langle g_1, \ldots, g_n \rangle \le F(\Sigma)$ (where w_i represents the group element g_i). Since $\mathcal{G}$ 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 $\mathcal{A}$ that produces exactly one word that is denoted with $\mathrm{val}(\mathcal{A})$. 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 $\mathcal{A}$ over the alphabet $\Gamma = \Sigma \cup \Sigma^{-1}$ one can compute in polynomial time a new straight-line program $\mathcal{A}'$ such that $\mathrm{val}(\mathcal{A}') = \mathsf{red}(\mathrm{val}(\mathcal{A}))$; see [20, Theorem 4.11]. Hence, we can compute in polynomial time a straight-line program $\mathcal{A}'$ for $\mathsf{red}(w_0)$. The transition labels of the automaton $\mathcal{G}$ can be also transformed into equivalent straight-line programs; such automata with straight-line compressed transition labels were investigated in [13]. It remains to check in polynomial time whether the deterministic automaton $\mathcal{G}$ accepts $\mathrm{val}(\mathcal{A}')$. This is possible in polynomial time by [13, Theorem 1]. ◀

4 Power-compressed subgroup membership for virtually free groups

A main advantage of the power-compressed subgroup membership is that its complexity is preserved under finite index group extensions. The proof of the following lemma follows [11], 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 us of the conjugate collection process for power words from [21, Theorem 6].

▶ **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 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 $N \le H$ and N has still finite index in G (the later is a well-known fact). Since $N \le H$, 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 $\overline{r} \in Hr^{-1}$. Recall that G and H are fixed groups, hence $r \cdot r'$ and $\overline{r}$ can be computed in constant time. In [21, Theorem 6] it is shown that the power word problem for G can be reduced in polynomial time (in fact, in NC^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 now take 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 $g_0 \in A := \langle g_1, \ldots, g_n \rangle$. In the following we will not distinguish between g_i and w_i.

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 the set of all representatives $r \in R$ such that $Hr \cap A \neq \emptyset$. Let us denote this closure with V. Clearly, $1 \in V$. If $r_0 \notin V$ then we have $w_0 = w_0' r_0 \notin A$.

Let us now assume that $r_0 \in V$. First assume that $r_0 = 1$, i.e., $w_0 = w_0' \in H$. Hence, $w_0 \in A$ if and only if $w_0 \in H \cap A$. We now compute a finite list of generators for $H \cap A$ written as power words over Θ. For this we follow [11]: we compute a power-compressed graph $\mathcal{G} = (V, E, \iota, \tau, \lambda, 1)$ (in the sense of Section 3.2) by taking V as the set of vertices. We draw an edge from $r \in V$ to $r' \in V$ labelled with the power word w_i (respectively, w_i^{-1}) iff $r \cdot r_i = r'$ (respectively, $r \cdot \overline{r}_i = r'$). Note that every edge has an inverse edge. The label of a path from $1 \in V$ back to $1 \in V$ in the graph $\mathcal{G}$ is a word over $\{w_1, w_1^{-1}, \ldots, w_n, w_n^{-1}\}$ and hence can be viewed as a power word over the alphabet Σ. As such, it represents an element of the group $H \cap A$.

Let T be a spanning tree of $\mathcal{G}$ and let $E \setminus T$ be the set of edges that do not belong to T. 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 [11] for details. Moreover, every x_e can be written as 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 $r_e = 1$. This concludes the case that $r_0 = 1$.

Finally, the case that $r_0 \in V$ but $r_0 \neq 1$ can be easily reduced to the case $r_0 = 1$: we use the same 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 $w_0 x^{-1} = w_0' r_0 r_0^{-1} y^{-1} = w_0' y^{-1}$, where the latter can be written as a power word over Θ. Since the word x represents an element of A we have $w_0 \in A$ if and only if $w_0 x^{-1} \in A$ if and only if $w_0' y^{-1} \in A$. This concludes the proof. ◀

From Theorem 12 and Lemma 13 we immediately obtain the following corollary:

▶ **Corollary 14.** *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 [32].

▶ **Lemma 15.** *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 [12], see also [8, Proposition 15.4]. Now, $\mathsf{SL}(2, \mathbb{Z})$ is a normal subgroup of index two in $\mathsf{GL}(2, \mathbb{Z})$. Fix a 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 [12] 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 16.** *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 14. By Lemma 15 this shows the corollary. ◀

5 Future work

There is not much hope to generalize Corollary 16 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 [17].

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 there exists an NP algorithm. 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 [4].

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.

References

1 Jürgen Avenhaus and Klaus Madlener. The Nielsen reduction and P-complete problems in free groups. *Theoretical Computer Science*, 32(1-2):61–76, 1984.

2 Jürgen Avenhaus and Dieter Wißmann. 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*, pages 322–337. ACM Press, 1989.

3 Frédérique Bassino, Ilya Kapovich, Markus Lohrey, Alexei Miasnikov, Cyril Nicaud, Andrey Nikolaev, Igor Rivin, Vladimir Shpilrain, Alexander Ushakov, and Pascal Weil. Compression techniques in group theory. In *Complexity and Randomness in Group Theory*, chapter 4. De Gruyter, 2020.

4 Paul C. Bell, Mika Hirvensalo, and Igor Potapov. The identity problem for matrix semigroups in $SL_2(\mathbb{Z})$ is NP-complete. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 187–206. SIAM, 2017.

5 Michèle Benois. Parties rationnelles du groupe libre. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, Séries A, 269:1188–1190, 1969.

6 Michèle Benois and Jacques Sakarovitch. On the complexity of some extended word problems defined by cancellation rules. *Information Processing Letters*, 23(6):281–287, 1986.

7 Michaël Cadilhac, Dmitry Chistikov, and Georg Zetzsche. Rational subsets of Baumslag-Solitar groups. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, volume 168 of *LIPIcs*, pages 116:1–116:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

8 Volker Diekert and Murray Elder. Solutions of twisted word equations, EDT0L languages, and context-free groups. *CoRR*, abs/1701.03297, 2017. `arXiv:1701.03297`.

9 Volker Diekert, Igor Potapov, and Pavel Semukhin. 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*, pages 122–129. ACM, 2020.

10 Stefan Friedl and Henry Wilton. The membership problem for 3-manifold groups is solvable. *Algebraic & Geometric Topology*, 16(4):1827–1850, 2016.

11 Zeph Grunschlag. *Algorithms in Geometric Group Theory.* PhD thesis, University of California at Berkley, 1999.

12 Yuri Gurevich and Paul E. Schupp. Membership problem for the modular group. *SIAM Journal on Computing*, 37(2):425–459, 2007.

13 Artur Jeż. The complexity of compressed membership problems for finite automata. *Theory of Computing Systems*, 55(4):685–718, 2014.

14 Ilya Kapovich and Alexei Myasnikov. Stallings foldings and subgroups of free groups. *Journal of Algebra*, 248(2):608–668, 2002.

15 Ilya Kapovich, Richard Weidmann, and Alexei Myasnikov. Foldings, graphs of groups and the membership problem. *International Journal of Algebra and Computation*, 15(1):95–128, 2005.

16 Olga G. Kharlampovich, Alexei G. Myasnikov, Vladimir N. Remeslennikov, and Denis E. Serbin. Subgroups of fully residually free groups: algorithmic problems. In *Group theory, statistics, and cryptography*, volume 360 of *Contemporary Mathematics*, pages 63–101. AMS, Providence, RI, 2004.

17 Evgeny I. Khukhro and Victor D. Mazurov. Unsolved problems in group theory. the Kourovka notebook. *CoRR*, arXiv:1401.0300v19, 2020. Problem 12.50. `arXiv:1401.0300v19`.

18 Sang-Ki Ko, Reino Niskanen, and Igor Potapov. 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*, pages 132:1–132:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

19 Klaus-Jörn Lange and Pierre McKenzie. On the complexity of free monoid morphisms. In *Proceedings of the 9th International Symposium on Algorithms and Computation, ISAAC 1998*, number 1533 in Lecture Notes in Computer Science, pages 247–256. Springer, 1998.

20 Markus Lohrey. *The Compressed Word Problem for Groups.* SpringerBriefs in Mathematics. Springer, 2014.

21 Markus Lohrey and Armin Weiß. The power word problem. In *Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019*, volume 138 of *LIPIcs*, pages 43:1–43:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

22 M. Lothaire. *Combinatorics on Words.* Cambridge University Press, 1997.

23 Roger C. Lyndon and Paul E. Schupp. *Combinatorial Group Theory.* Springer, 1977.

24 Anatolij I. Mal'cev. On homomorphisms onto finite groups. *American Mathematical Society Translations, Series 2*, 119:67–79, 1983. Translation from Ivanov. Gos. Ped. Inst. Ucen. Zap. 18 (1958) 49–60.

25 Luda Markus-Epstein. Stallings foldings and subgroups of amalgams of finite groups. *International Journal of Algebra and Compution*, 17(8):1493–1535, 2007.

26 K. A. Mihaĭlova. The occurrence problem for direct products of groups. *Math. USSR Sbornik*, 70:241–251, 1966. English translation.

27 Igor Potapov and Pavel Semukhin. Decidability of the membership problem for 2 × 2 integer matrices. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 170–186. SIAM, 2017.

28 Eliyahu Rips. Subgroups of small cancellation groups. *Bulletin of the London Mathematical Society*, 14:45–47, 1982.

29 Nikolai S. Romanovskiĭ. Some algorithmic problems for solvable groups. *Algebra i Logika*, 13(1):26–34, 1974.

30 Nikolai S. Romanovskiĭ. The occurrence problem for extensions of abelian groups by nilpotent groups. *Sibirskii Matematicheskii Zhurnal*, 21:170–174, 1980.

31 Paul E. Schupp. Coxeter groups, 2-completion, perimeter reduction and subgroup separability. *Geometriae Dedicata*, 96:179–198, 2003.

32 Jean-Pierre Serre. *Trees*. Springer, 1980.

33 Charles C. Sims. Computation with permutation groups. In *Proceedings of SYMSAC 1971*, pages 23–28. Association for Computing Machinery, 1971.

34 John R. Stallings. Topology of finite graphs. *Inventiones Mathematicae*, 71(3):551–565, 1983.

35 Nicholas W. M. Touikan. A fast algorithm for Stallings' folding process. *International Journal of Algebra and Computation*, 16(6):1031–1045, 2006.