Efficient computation in groups via compression

Markus Lohrey[1] and Saul Schleimer[2]

[1] Universität Stuttgart, FMI, Germany
[2] School of Mathematics and Statistics, Rutgers University, Mathematics Department, New Brunswick, New Jersey, USA
lohrey@informatik.uni-stuttgart.de, saulsch@math.rutgers.edu

Abstract. A compressed variant of the word problem for finitely generated groups, where the input word is given by a context-free grammar that generates exactly one string (also called a straight-line program), is studied. It is shown that finite extensions and free products preserve the complexity of the compressed word problem and that the compressed word problem for a graph group can be solved in polynomial time. Using these results together with connections between the compressed word problem and the (classical) word problem allows to obtain new upper complexity bounds for certain automorphism groups and group extensions.

1 Introduction

The *word problem for finitely generated groups* is a fundamental computational problem linking group theory, topology, mathematical logic, and computer science. For a group G, finitely generated by Σ, it is asked whether a word over Σ and the inverses of Σ represents the 1 of G. The word problem was introduced in the pioneering work of Dehn from 1910 [10] in relation with topological questions. It took about 45 years until Novikov [32] and later independently Boone [5] proved the existence of a finitely presented group with an undecidable word problem. Despite this negative result, many natural classes of groups with decidable word problems were found. Prominent examples are for instance finitely generated linear groups, automatic groups [17], and one-relator groups. With the advent of computational complexity theory, also the complexity of word problems became an active research area. For instance, it was shown that for a finitely generated linear group the word problem can be solved in logarithmic space [27, 40], that automatic groups have polynomial time solvable (in fact quadratic) word problems [17], and that the word problem for a one-relator group is primitive recursive [7].

Group theoretic operations, which preserve (or moderately increase) the complexity of the word problem, are useful in order to get a building set for constructing groups with efficiently solvable word problems. An example of such a construction is the free product: it is not hard to see that the word problem for a free product $G * H$ can be reduced in polynomial time to the word problem for G and H. In this paper, we will increase the building set of such group operations by introducing a new technique for obtaining upper complexity bounds for word problems. This technique is based on data compression. More precisely, we use a compressed representation of strings — so called *straight-line programs*, briefly SLPs — which is able to achieve exponential compression rates for strings with repeated subpatterns. Formally, an SLP G is a context-free grammar, which generates exactly one string $\mathrm{eval}(G)$. Recently, SLPs turned out to be a very flexible compressed representation of strings, which is well-suited for studying algorithms on compressed data. For instance, several polynomial time algorithms for the pattern matching problem on SLP-compressed input strings were developed [18, 26, 31, 37]. In [28], the first author started to investigate the *compressed word problem* for a finitely generated group G with finite generating set Σ. For a given SLP G that generates a string over Σ and the inverses of Σ it is asked whether $\mathrm{eval}(G)$ represents the 1 of G (actually, in [28] the compressed word problem for finitely generated monoids was studied). This problem is equivalent to the well-known circuit evaluation problem, where we ask whether a circuit over a finitely generated group G (i.e., an acyclic directed graph with leafs labeled by generators of G and internal nodes labeled by the group multiplication) evaluates to the 1 of G. In [3], this problem was investigated for finite groups, and it was shown that there exist finite groups, for which the circuit evaluation problem is complete for P (deterministic polynomial time).

In [3, 28], the main motivation for studying the compressed word problem came from computational complexity theory. Since the input in the compressed word problem is given in a more compact form than in the ordinary word problem, it can be expected that in general the compressed word problem for a group G is more difficult than the ordinary word problem. For instance, whereas the word problem for a finitely generated free group belongs to the class L (deterministic logspace) [27], the compressed word problem for a finitely generated free group of rank at least two is P-complete [28].[1]

In [38], the second author used the polynomial time algorithm for the compressed word problem for a free group in order to present a polynomial time algorithm for the ordinary word problem for the automorphism group of a free group, which answered a question from [24]. Hence, the compressed word problem is used in order to obtain better algorithms for the ordinary word problem. In this paper, we will continue this idea and thereby obtain efficient algorithms for a variety of word problems. In order to achieve this goal, we proceed in two steps:

In the first step (Section 3) we show connections between the compressed word problem for a group G and the word problem for some group derived from G. We prove three results of this kind:

- If H is a finitely generated subgroup of the automorphism group of a group G, then the word problem for H is logspace reducible to the compressed word problem for G (Proposition 2). This result is a straight-forward extension of Theorem 5.2 from [38].
- The word problem for the semidirect product $K \rtimes_\varphi Q$ of two finitely generated groups K and Q is logspace reducible to (i) the word problem for Q and (ii) the compressed word problem for K (Proposition 3).
- If K is a finitely generated normal subgroup of G such that the quotient G/K is an automatic group, then the word problem for G is polynomial time reducible to the compressed word problem for K (Proposition 4).

[1] It is believed, although not proven, that L is a proper subclass of P.

In the second step (Section 4) we concentrate on the compressed word problem. We prove the following results:

- If K is a finitely generated subgroup of G such that the index $[G : K]$ is finite, then the compressed word problem for G is polynomial time reducible to the compressed word problem for K (Theorem 1).
- The compressed word problem for a free product $G_1 * G_2$ is polynomial time reducible (under Turing reductions) to the compressed word problem for G_1 and G_2 (Theorem 2). This result even holds for the more general graph product construction [20] (Theorem 4).
- The compressed word problem for a graph group [15] can be solved in polynomial time (Theorem 3). In a graph group, every defining relation is of the form $ab = ba$ for generators a and b.
- The compressed word problem for a finitely generated linear group belongs to the complexity class coRP (Theorem 5), which is the complementary class of randomized polynomial time, see Section 4.4 for the definition.

We end this paper with a few direct applications of the above results. Let us mention one of them concerning topology, see [41] for definitions: Crisp and Wiest [9] have shown shown that the fundamental group of any orientable surface (and of most non-orientable surfaces) embeds in a graph group. It follows that for most fundamental groups of surfaces, the word problem for the corresponding automorphism group can be solved in polynomial time. These automorphism groups play a very important role in algebraic topology.

2 Preliminaries

Let Σ be a finite alphabet. With $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ we denote the set of non-empty words over Σ. We use $\Sigma^{-1} = \{a^{-1} \mid a \in \Sigma\}$ to denote a disjoint copy of Σ. Let $\Sigma^{\pm 1} = \Sigma \cup \Sigma^{-1}$. Define $(a^{-1})^{-1} = a$; this defines an involution ${}^{-1} : \Sigma^{\pm 1} \to \Sigma^{\pm 1}$, which can be extended to an involution on $(\Sigma^{\pm 1})^*$ by setting $(a_1 \cdots a_n)^{-1} = a_n^{-1} \cdots a_1^{-1}$. For $\Gamma \subseteq \Sigma$, we denote by $\pi_\Gamma(w)$ the projection of the word w to the alphabet Γ, i.e., we erase in w all symbols from $\Sigma \setminus \Gamma$.

For a word $s = a_1 \cdots a_m$ ($a_i \in \Sigma$) let

- $|s| = m$, $\mathrm{alph}(s) = \{a_1, \ldots, a_m\}$,
- $s[i] = a_i$ for $1 \leq i \leq m$
- $s[i : j] = a_i \cdots a_j$ for $1 \leq i \leq j \leq m$ and $s[i : j] = \varepsilon$ for $i > j$,
- $s[: i] = s[1 : i] = a_1 \cdots a_i$ for $0 \leq i \leq m$, and
- $s[i :] = s[i : m] = a_i \cdots a_m$ for $1 \leq i \leq m + 1$.

For $c \in \mathbb{N}$ let $\Sigma^{\leq c} = \{w \in \Sigma^* \mid |w| \leq c\}$ denote the set of all words of length at most c.

For background in complexity theory see [33]. For languages K, L we write $K \leq_m^P L$ (resp. $K \leq_m^{\log} L$) if there exists a polynomial time (resp. logspace) many-one reduction from K to L. We write $K \leq_T^P L$ if there exists a polynomial time Turing reduction from K to L, which means that K can be solved in deterministic polynomial time on a Turing machine with oracle access to the language L. Let $\preceq \in \{\leq_m^P, \leq_m^{\log}, \leq_T^P\}$. In case $K \preceq L_1 \times \cdots \times L_n$ we write $K \preceq (L_1, \ldots, L_n)$. Clearly, if $L_1, \ldots, L_n$ belong to the class P (deterministic polynomial time) and $K \leq_T^P (L_1, \ldots, L_n)$, then K belongs to P as well.

2.1 Groups

For background in combinatorial group theory see [30]. Let G be a *finitely generated group* and let Σ be a finite *group generating set* for G. Hence, $\Sigma^{\pm 1}$ is a finite *monoid generating set* for G and there exists a canonical monoid homomorphism $h : (\Sigma^{\pm 1})^* \to G$, which maps a word $w \in (\Sigma^{\pm 1})^*$ to the group element represented by w. For $u, v \in (\Sigma^{\pm 1})^*$ we will also say that $u = v$ in G in case $h(u) = h(v)$.

The *word problem* for G with respect to Σ is the following decision problem:

INPUT: A word $w \in (\Sigma^{\pm 1})^*$.
QUESTION: $w = 1$ in G, i.e., $h(w) = 1$?

It is well known and easy to see that if Γ is another finite generating set for G, then the word problem for G with respect to Σ is logspace many-one reducible to the word problem for G with respect to Γ. This justifies to speak just of the word problem for the group G. The word problem for G is also denoted by $\mathrm{WP}(G)$. The *automorphism group* of a group G is denoted by $\mathrm{Aut}(G)$.

The *free group* $F(\Sigma)$ generated by Σ can be defined as the quotient monoid

$$F(\Sigma) = (\Sigma^{\pm 1})^* / \{aa^{-1} = \varepsilon \mid a \in \Sigma^{\pm 1}\}.$$

As usual, the *free product* of two groups G_1 and G_2 is denoted by $G_1 * G_2$. Assume that Σ_i is a finite generating set for G_i ($i \in \{1, 2\}$), where $\Sigma_1 \cap \Sigma_2 = \emptyset$. Then, every element of the free product $G_1 * G_2$ can be represented by a word $u \in (\Sigma_1^{\pm 1} \cup \Sigma_2^{\pm 2})^*$, where

- $u = u_1 \cdots u_n$ for some $n \geq 0$ and $u_1, \ldots, u_n \in (\Sigma_1^{\pm 1})^+ \cup (\Sigma_2^{\pm 1})^+$,
- for all $1 \leq i < n$: $u_i \in (\Sigma_1^{\pm 1})^+ \Leftrightarrow u_{i+1} \in (\Sigma_2^{\pm 1})^+$, and
- for all $1 \leq i \leq n$: $u_i \neq 1$ in G_1 if $u_i \in (\Sigma_1^{\pm 1})^+$ and $u_i \neq 1$ in G_2 if $u_i \in (\Sigma_2^{\pm 1})^+$.

We call such a word *irreducible in* $G_1 * G_2$. If $v = v_1 \cdots v_m$ is another word, irreducible in $G_1 * G_2$ (with $v_1, \ldots, v_m \in (\Sigma_1^{\pm 1})^+ \cup (\Sigma_2^{\pm 1})^+$ and $v_i \in (\Sigma_1^{\pm 1})^+ \Leftrightarrow v_{i+1} \in (\Sigma_2^{\pm 1})^+$ for all $1 \leq i < m$), then u and v represent the same group element of $G_1 * G_2$ if and only if $n = m$ and for all $1 \leq i \leq n$, u_i and v_i represent the same group element of G_1 (if $u_i, v_i \in (\Sigma_1^{\pm 1})^+$) or of G_2 (if $u_i, v_i \in (\Sigma_2^{\pm 1})^+$).

For the standard definition of *automatic groups*, see [17]. Every automatic group G is finitely presented and its word problem can be solved in time $O(n^2)$. We will need the following important properties of automatic groups, see [17]. Let G be automatic and let Σ be a finite generating set for G. Then there exists a normal form mapping $\mathrm{NF} : (\Sigma^{\pm 1})^* \to (\Sigma^{\pm 1})^*$ and constants $\alpha, \beta \in \mathbb{N}$ with the following properties, where $u, v \in (\Sigma^{\pm 1})^*$, and $a \in \Sigma \cup \Sigma^{-1}$:

- $\mathrm{NF}(u) = \mathrm{NF}(v)$ if and only if $u = v$ in G, $\mathrm{NF}(u) = u$ in G, and $\mathrm{NF}(\varepsilon) = \varepsilon$

– The set of normal forms $\mathrm{NF}((\Sigma^{\pm 1})^*)$ is regular.
– The normal form $\mathrm{NF}(u)$ can be computed in time $O(|u|^2)$ (hence the word problem of G can be solved in quadratic time).
– $||\mathrm{NF}(u)| - |\mathrm{NF}(ua)|| \leq \alpha$, i.e., the length of the normal form only changes by a constant amount when appending a generator to a word.
– If $\mathrm{NF}(u) = a_1 \cdots a_m$ and $\mathrm{NF}(ua) = b_1 \cdots b_n$ (with $a_1, \dots, a_m, b_1, \dots b_n \in \Sigma \cup \Sigma^{-1}$), then there exists words $r_0, r_1, \dots, r_{\max(m,n)} \in (\Sigma \cup \Sigma^{-1})^{\leq \beta}$ such that $r_0 = \varepsilon$, $r_{\max(m,n)} = a$, and $(a_1 \cdots a_i) r_i = (b_1 \cdots b_i)$ in G for all $1 \leq i \leq \max(m,n)$ (here we set $a_i = \varepsilon$ for $m < i \leq \max(m,n)$ and $b_i = \varepsilon$ for $n < i \leq \max(m,n)$).

The last property is also called the *synchronous fellow traveller property*.

2.2 Trace monoids and graph groups

In the following we introduce some notions from trace theory, see [11, 14] for more details. This material will be only needed in Section 4.3. An *independence alphabet* is just a finite undirected graph (Σ, I) without loops. Hence, $I \subseteq \Sigma \times \Sigma$ is an irreflexive and symmetric relation. The complementary graph (Σ, D) with $D = (\Sigma \times \Sigma) \setminus I$ is called a *dependence alphabet*. The *trace monoid* $\mathbb{M}(\Sigma, I)$ is defined as the quotient

$$\mathbb{M}(\Sigma, I) = \Sigma^* / \{ab = ba \mid (a,b) \in I\}.$$

It is a cancellative monoid. Elements of $\mathbb{M}(\Sigma, I)$ are called *traces*. The trace represented by the word $s \in \Sigma^*$ is also denoted by $[s]_I$. For $a \in \Sigma$ let $I(a) = \{b \in \Sigma \mid (a,b) \in I\}$ and $D(a) = \{b \in \Sigma \mid (a,b) \in D\}$.

Traces can be represented conveniently by *dependence graphs*, which are node-labelled directed acyclic graphs. Let $s = a_1 \cdots a_n$ be a word, where $a_i \in \Sigma$. The vertex set of the dependence graph D_s of s is $\{1, \dots, n\}$ and vertex i is labeled with $a_i \in \Sigma$. There is an edge from vertex i to j in D_s if and only if $i < j$ and $(a_i, a_j) \notin I$. Then, for two words $s, t \in \Sigma^*$ we have $[s]_I = [t]_I$ if and only if D_s and D_t are isomorphic. In particular, we can speak of the dependence graph of a trace u. Clearly, by taking the transitive and reflexive closure of the edge relation of a dependence graph D_s, one obtains a partial order.

A trace u is a prefix of a trace v if there exists a trace w such that $v = uw$ in $\mathbb{M}(\Sigma, I)$. The prefixes of a trace v correspond to the downward-closed node sets of the dependence graph of u. For two traces $u, v \in \mathbb{M}(\Sigma, I)$, the infimum with respect to the prefix order is denoted by $u \sqcap v$. That is, $u \sqcap v$ is a prefix of u and v and every other common prefix of u and v is a prefix of $u \sqcap v$. With $u \setminus v$ we denote the unique trace t such that $u = (u \sqcap v)t$; uniqueness follows from the fact that $\mathbb{M}(\Sigma, I)$ is cancellative. Note that $u \setminus v = u \setminus (u \sqcap v)$. For words $s, t \in \Sigma^*$ we write $s \preceq_I t$ if the trace $[s]_I$ is a prefix of the trace $[t]_I$.

Example 1. Let (Σ, I) be the following independence alphabet:

$$b — d — a — c$$

The corresponding dependence alphabet looks as follows, where the self loop at every node is omitted:

$$\begin{tikzcd} a \arrow[r, dash] & b \arrow[r, dash] & c \arrow[r, dash] & d \end{tikzcd}$$

Let $s = dabcd$ and $t = abaadcdb$. The (Hasse diagram of the reflexive and transitive closure of the) dependence graph of s and t, respectively, looks as follows. For a node $i \in \{1, \ldots, 8\}$ we only show its label from $\{a, b, c, d\}$.

$$D_s \quad \begin{tikzcd} a \arrow[r] & b \arrow[dr] & & \\ d \arrow[rr] & & c \arrow[r] & d \end{tikzcd} \qquad D_t \quad \begin{tikzcd} a \arrow[r] & b \arrow[r] \arrow[dr] & a \arrow[r] & a \arrow[r] & b \\ d \arrow[rr] & & c \arrow[urr] \arrow[rr] & & d \end{tikzcd}$$

Since we only show Hasse diagrams, we omit for instance the edge from the first d to the second d in D_s.

We have $s \preceq_I t$. For $u = dabcbdc$ the dependence graph is

$$D_u \quad \begin{tikzcd} a \arrow[r] & b \arrow[dr] & & b \arrow[dr] & \\ d \arrow[rr] & & c \arrow[ur] \arrow[r] & d \arrow[r] & c \end{tikzcd}$$

We see that $[t]_I \sqcap [u]_I = [s]_I$, $[t]_I \setminus [s]_I = [t]_I \setminus [u]_I = [aab]_I$, and $[u]_I \setminus [s]_I = [u]_I \setminus [t]_I = [bc]_I$.

A clique covering of the dependence alphabet (Σ, D) is a tuple of subsets $(\Sigma_i)_{1 \leq i \leq n}$ such that $\Sigma = \bigcup_{1 \leq i \leq n} \Sigma_i$ and $D = \bigcup_{1 \leq i \leq n} (\Sigma_i \times \Sigma_i)$. It is well-known that for a clique covering $(\Sigma_i)_{1 \leq i \leq n}$ and two words $s, t \in \Sigma^*$ on has $[s]_I = [t]_I$ if and only if $\pi_{\Sigma_i}(s) = \pi_{\Sigma_i}(t)$ for all $1 \leq i \leq n$. This fact is also known as the projection lemma [14]. We also need the following simple fact:

Lemma 1. *Let $(\Sigma_i)_{1 \leq i \leq n}$ be a clique covering of the dependence alphabet (Σ, D) and let $s, t \in \Sigma^*$. Then $s \preceq_I t$ if and only if $\pi_{\Sigma_i}(s)$ is a prefix of $\pi_{\Sigma_i}(t)$ for all $1 \leq i \leq n$.*

Proof. The "only if"-direction is trivial. For the "if"-direction assume that there exist words $u_i \in \Sigma_i^*$ ($1 \leq i \leq n$) such that $\pi_{\Sigma_i}(t) = \pi_{\Sigma_i}(s) u_i$. From [16, Prop. 1.6] it follows that there exists a word $u \in \Sigma^*$ such that $\pi_{\Sigma_i}(u) = u_i$ for all $1 \leq i \leq n$. Hence $\pi_{\Sigma_i}(t) = \pi_{\Sigma_i}(s) \pi_{\Sigma_i}(u) = \pi_{\Sigma_i}(su)$. By the projection lemma we have $[t]_I = [su]_I$, i.e., $s \preceq_I t$. □

Example 2. A clique covering of the dependence alphabet from Example 1 is

$$(\{a, b\}, \{b, c\}, \{c, d\}).$$

The tuple of projections for the word s (resp. t) from Example 1 is (ab, bc, dcd) (resp. $(abaab, bcb, dcd)$). Every component of the tuple (ab, bc, dcd) is a prefix of the corresponding component of the tuple $(abaab, bcb, dcd)$. Hence, we have indeed $s \preceq_I t$.

A *trace rewriting system* R over $\mathbb{M}(\Sigma, I)$ is just a finite subset of $\mathbb{M}(\Sigma, I) \times \mathbb{M}(\Sigma, I)$ [11]. We can define the *one-step rewrite relation* $\to_R \subseteq \mathbb{M}(\Sigma, I) \times \mathbb{M}(\Sigma, I)$ by: $x \to_R y$ if and only if there are $u, v \in \mathbb{M}(\Sigma, I)$ and $(\ell, r) \in R$ such that $x = u\ell v$ and $y = urv$. The notion of a *confluent* and *terminating* trace rewriting system is defined as for other types of rewriting systems [4]. A trace t is *irreducible* with respect to R if there does not exist a trace u with $t \to_R u$. The set of all traces that are irreducible with respect to R is denoted with $\mathrm{IRR}(R)$. If R is terminating and confluent, then for every trace u, there exists a unique *normal form* $\mathrm{NF}_R(u) \in \mathrm{IRR}(R)$ such that $u \xrightarrow{*}_R \mathrm{NF}_R(u)$.

The graph group $\mathbb{G}(\varSigma, I)$ is defined as the quotient group

$$\mathbb{G}(\varSigma, I) = F(\varSigma)/\{ab = ba \mid (a,b) \in I\}.$$

Note that $(a,b) \in I$ implies $a^{-1}b = ba^{-1}$ in $\mathbb{G}(\varSigma, I)$. Thus, the graph group $\mathbb{G}(\varSigma, I)$ can be also defined as the quotient

$$\mathbb{G}(\varSigma, I) = \mathbb{M}(\varSigma^{\pm 1}, I)/\{aa^{-1} = \varepsilon \mid a \in \varSigma^{\pm 1}\}.$$

Here, we implicitly extend $I \subseteq \varSigma \times \varSigma$ to $I \subseteq \varSigma^{\pm 1} \times \varSigma^{\pm 1}$ by setting $(a^\alpha, b^\beta) \in I$ if and only if $(a,b) \in I$ for $a, b \in \varSigma$ and $\alpha, \beta \in \{1, -1\}$. We can also lift the involution ${}^{-1} : (\varSigma^{\pm 1})^* \to (\varSigma^{\pm 1})^*$ to an involution ${}^{-1} : \mathbb{M}(\varSigma^{\pm 1}, I) \to \mathbb{M}(\varSigma^{\pm 1}, I)$ by setting $[s]_I^{-1} = [s^{-1}]_I$ (well-definedness is easily seen).

Free groups and free abelian groups arise as special cases of graph groups; note that $\mathbb{G}(\varSigma, \emptyset) = F(\varSigma)$ and $\mathbb{G}(\varSigma, (\varSigma \times \varSigma) \setminus \mathrm{id}_\varSigma) = \mathbb{Z}^{|\varSigma|}$. Graph groups were studied e.g. in [15]; they are also known as *free partially commutative groups* [12, 42], *right-angled Artin groups* [6, 9], and *semifree groups* [1].

2.3 Grammar based compression

In this section we introduce straight-line programs, which are used as a compressed representation of strings with reoccuring subpatterns. Following [35], a *straight-line program (SLP) over the alphabet $\varGamma$* is a context-free grammar $\mathbb{A} = (V, \varGamma, S, P)$, where V is the set of *nonterminals*, $\varGamma$ is the set of *terminals*, $S \in V$ is the *initial nonterminal*, and $P \subseteq V \times (V \cup \varGamma)^*$ is the set of *productions*, such that (i) for every $X \in V$ there is exactly one $\alpha \in (V \cup \varGamma)^*$ with $(X, \alpha) \in P$ and (ii) there is no cycle in the relation $\{(X, Y) \in V \times V \mid \exists \alpha : (X, \alpha) \in P, Y \in \mathrm{alph}(\alpha)\}$. A production (X, α) is also written as $X \to \alpha$. The language generated by the SLP $\mathbb{A}$ contains exactly one word that is denoted by $\mathrm{eval}(\mathbb{A})$. More generally, every nonterminal $X \in V$ produces exactly one word that is denoted by $\mathrm{eval}_{\mathbb{A}}(X)$. We omit the index $\mathbb{A}$ if the underlying SLP is clear from the context. The size of $\mathbb{A}$ is $|\mathbb{A}| = \sum_{(X,\alpha) \in P} |\alpha|$. Every SLP can be transformed in polynomial time into an equivalent SLP that is in *Chomsky normal form* (as a context-free grammar). This means that all productions have the form $A \to BC$ or $A \to a$ for nonterminals A, B, and C and a terminal a. The following tasks can be solved in polynomial time (the first two problems can be reduced to simple arithmetic, whereas the third problem requires more subtle techniques):

– Given an SLP $\mathbb{A}$, calculate $|\mathrm{eval}(\mathbb{A})|$.
– Given an SLP $\mathbb{A}$ and a number $i \in \{0, \ldots, |\mathrm{eval}(\mathbb{A})| - 1\}$, calculate $\mathrm{eval}(G)[i]$.
– Given SLPs $\mathbb{A}$ and $\mathbb{B}$ decide whether $\mathrm{eval}(\mathbb{A}) = \mathrm{eval}(\mathbb{B})$ [34].

Let G be a finitely generated group and let $\varSigma$ be a finite generating set for G. The *compressed word problem* for G with respect to $\varSigma$ is the following decision problem:

INPUT: An SLP $\mathbb{A}$ over the terminal alphabet $\varSigma^{\pm 1}$.
QUESTION: Does $\mathrm{eval}(\mathbb{A}) = 1$ hold in G?

Here, the input size is $|\mathbb{A}|$. It is easy to see that also for the compressed word problem the complexity does not depend on the chosen generating set, which allows to speak of the compressed word problem for the group G. The compressed word problem for G is also denoted by $\mathrm{CWP}(G)$. The following fact is trivial:

Proposition 1. *Assume that H is a finitely generated subgroup of the finitely generated group G. Then $\mathrm{CWP}(H) \leq_m^{\log} \mathrm{CWP}(G)$.*

The next lemma is crucial for our applications of compressed word problems.

Lemma 2. *For a given sequence $\varphi_1, \dots, \varphi_n$ of homomorphisms $\varphi_i : \Gamma^* \to \Gamma^*$ ($1 \leq i \leq n$) and a symbol $a \in \Gamma$ we can compute in logarithmic space an SLP $\mathbb{A}$ such that $\mathrm{eval}(\mathbb{A}) = \varphi_1 \cdots \varphi_n(a)$. Moreover, $|\mathbb{A}| = O(\sum_{a \in \Gamma} \sum_{i=1}^{n} |\varphi_i(a)|)$. In particular, if Γ is fixed and every φ_i is taken from some fixed finite set of homomorphisms, then $|\mathbb{A}| = O(n)$.*

Proof. Let us take nonterminals $A_{i,b}$, where $0 \leq i \leq n$ and $b \in \Gamma$, and define the productions as follows:

$$\begin{aligned} &A_{0,b} \to b \\ &A_{i,b} \to A_{i-1,a_1} \cdots A_{i-1,a_m}, \text{ where } \varphi_i(b) = a_1 \cdots a_m \end{aligned}$$

By induction on i one can easily show that $\mathrm{eval}(\mathbb{A}_{i,b}) = \varphi_1 \cdots \varphi_i(b)$. $\square$

A *composition system* $\mathbb{A} = (V, \Gamma, S, P)$ is defined analogously to an SLP, but in addition to productions of the form $A \to \alpha$ ($A \in V$, $\alpha \in (V \cup \Gamma)^*$) it may also contain productions of the form $A \to B[i : j]$ for $B \in V$ and $i, j \in \mathbb{N}$ [18]. For such a production we define $\mathrm{eval}_{\mathbb{A}}(A) = \mathrm{eval}_{\mathbb{A}}(B)[i : j]$.[2] As for SLPs we define $\mathrm{eval}(\mathbb{A}) = \mathrm{eval}_{\mathbb{A}}(S)$. In [21], Hagenah presented a polynomial time algorithm, which transforms a given composition system $\mathbb{A}$ into an SLP $\mathbb{B}$ such that $\mathrm{eval}(\mathbb{A}) = \mathrm{eval}(\mathbb{B})$. We will also allow more general kinds of productions, where right-hand sides are arbitrary words, built up from terminals, nonterminals and symbols $B[: i], B[i :], B[i : j]$ for a nonterminal B and $i, j \in \mathbb{N}$. The semantics of such productions is the obvious one.

In Section 4.3 we will need the following generalization of composition systems: An *extended composition system* over the terminal alphabet Γ may contain in addition to productions of the form $A \to \alpha$ ($A \in V$, $\alpha \in (V \cup \Gamma)^*$) and $A \to B[i : j]$ ($B \in V$ and $i, j \in \mathbb{N}$) also productions of the form $A \to \pi_\Sigma(B)$ for $B \in V$ and $\Sigma \subseteq \Gamma$. For such a production we define $\mathrm{eval}_{\mathbb{A}}(A) = \pi_\Sigma(\mathrm{eval}_{\mathbb{A}}(B))$.

Lemma 3. *Let Γ be a fixed terminal alphabet. There is a polynomial time algorithm, which transforms a given extended composition system $\mathbb{A}$ over the terminal alphabet Γ into an SLP $\mathbb{B}$ such that $\mathrm{eval}(\mathbb{A}) = \mathrm{eval}(\mathbb{B})$.*

[2] In [18], a slightly more restricted formalism, where all productions have the form $A \to a \in \Gamma$ or $A \to B[j :]C[: i]$, was introduced. But this definition is easily seen to be equivalent to our formalism.

Proof. Let $\mathbb{A}$ be a given extended composition system. By the Hagenah's result from [21] it suffices to construct in polynomial time an equivalent composition system. For this, we construct in polynomial time a composition system $\mathbb{B}$, which contains for every nonterminal X of $\mathbb{A}$ and every subset Σ of the fixed terminal alphabet Γ a nonterminal X_Σ such that $\mathrm{eval}(X_\Sigma) = \pi_\Sigma(\mathrm{eval}(X))$. For this we introduce in a bottom-up way new productions: For a production $X \to a$ with $a \in \Gamma$ we introduce the productions $X_\Sigma \to \pi_\Sigma(a)$. For a production $X \to YZ$, we introduce the productions $X_\Sigma \to Y_\Sigma Z_\Sigma$. For a production $X \to \pi_\Theta(Y)$ let $X_\Sigma \to Y_{\Sigma\cap\Theta}$. Finally, consider a production $X \to Y[i : j]$. We introduce the productions $X_\Sigma \to Y_\Sigma[k : \ell]$, where $k = |\pi_\Sigma(\mathrm{eval}(Y)[: i-1])|+1$ and $\ell = |\pi_\Sigma(\mathrm{eval}(Y)[: j])|$. These lengths can be computed in polynomial time as follows: Implicitly, when processing the production $X \to Y[i : j]$ we have already constructed a composition system which generates the string $\mathrm{eval}(Y) = \mathrm{eval}(Y_\Gamma)$. Hence, by adding a single production, we can write down a composition system for the string $\mathrm{eval}(Y)[: i-1]$. Using Hagenah's algorithm [21] we can transform this composition system in polynomial time into an equivalent SLP. From this SLP, the length $|\pi_\Sigma(\mathrm{eval}(Y)[: i-1])|$ can be easily computed bottom-up (the SLP for the string $\mathrm{eval}(Y)[: i-1]$ is then not used anymore). ☐

It should be remarked that in the previous proof it is crucial that the alphabet Γ is fixed, i.e., not part of the input. Otherwise the construction would lead to an exponential blow-up. It is not clear whether Lemma 3 remains true, when the terminal alphabet Γ is part of the input.

3 Connections between the word problem and the compressed word problem

Our main motivation for studying the compressed word problem for a group are the following results:

Proposition 2 (cf [38]). *Let G be a finitely generated group and let H be a finitely generated subgroup of $\mathrm{Aut}(G)$. Then $\mathrm{WP}(H) \leq_m^{\log} \mathrm{CWP}(G)$.*

Proposition 3. *Let K and Q be finitely generated groups and let $\varphi : Q \to \mathrm{Aut}(K)$ be a homomorphism. Then, for the semidirect product $K \rtimes_\varphi Q$ we have $\mathrm{WP}(K \rtimes_\varphi Q) \leq_m^{\log} (\mathrm{WP}(Q), \mathrm{CWP}(K))$.*

Proof. Elements of the semidirect product $K \rtimes_\varphi Q$ can be written as pairs $(k, q) \in K \times Q$ and the multiplication is defined as $(k, q)(\ell, p) = (k \circ \varphi(q)(\ell), qp)$ (here $\circ$ is the multiplication in K; note that $\varphi(q) \in \mathrm{Aut}(K)$). Let us consider a word $(k_1, q_1)(k_2, q_2)\cdots(k_n, q_n)$, where k_i (resp. q_i) is a generator of K (resp. Q). In $K \rtimes_\varphi Q$, this word equals $(\theta_1(k_1) \circ \theta_2(k_2) \circ \cdots \circ \theta_n(k_n), q_1q_2\cdots q_n)$, where $\theta_i \in \mathrm{Aut}(K)$ is the automorphism defined by $\theta_i = \varphi(q_1 \cdots q_{i-1}) = \varphi(q_1)\cdots\varphi(q_{i-1})$ for $1 \leq i \leq n$ (note that $\theta_1 = \mathrm{id}_K$). By Lemma 2, we can compute in logarithmic space an SLP $\mathbb{A}$ over the generators of K, which produces the string $\theta_1(k_1)\theta_2(k_2)\cdots\theta_n(k_n)$. We have $(k_1, q_1)(k_2, q_2)\cdots(k_n, q_n) = 1$ in $K \rtimes_\varphi Q$ if and only if $q_1q_2\cdots q_n = 1$ in Q and $\mathrm{eval}(\mathbb{A}) = 1$ in K. This proves the proposition. ☐

The semidirect product $G = K \rtimes_\varphi Q$ is a an extension of K by Q, i.e., K is a normal subgroup of G with quotient $G/K \simeq Q$. A reasonable generalization of Proposition 3 would be $\mathrm{WP}(G) \leq_m^{\log} (\mathrm{WP}(G/K), \mathrm{CWP}(K))$. But this cannot be true: there exist finitely generated groups G, Q, K such that (i) $Q = G/K$, (ii) Q and K have decidable word problems, and (iii) G has an undecidable word problem [2]. On the other hand, if we require additionally, that Q is finitely presented (in fact, Q recursively presented suffices), then G must have a decidable word problem [8]. For the special case that the quotient $Q = G/K$ is automatic (and hence finitely presented), we can prove the following:

Proposition 4. *Let K be a finitely generated normal subgroup of G such that the quotient $Q = G/K$ is an automatic group. Then $\mathrm{WP}(G) \leq_m^P \mathrm{CWP}(K)$.*

Proof. Let Σ be a finite generating set for K and let Γ be a finite generating set of the automatic group $Q = G/K$ (recall that automatic groups are finitely presented). Let $\varphi : G \to Q$ be the canonical morphism and choose a mapping $h : Q \to G$ with $h(1) = 1$ and $\varphi(h(a)) = a$ for $a \in Q$. The set $\Sigma \cup h(\Gamma)$ generates G and there exists a so called factor set $f : Q \times Q \to K$ such that $h(a)h(b) = f(a,b)h(ab)$ for $a, b \in Q$.

Let us first prove the following claim (recall from Section 2.2 the existence of normal form mappings for automatic groups):

Claim 1. For a given word $w = a_1 \cdots a_n$ ($a_i \in \Gamma^{\pm 1}$) with $\mathrm{NF}(w) = b_1 \cdots b_m$ ($b_j \in \Gamma^{\pm 1}$) we can compute in polynomial time an SLP $\mathbb{A}(w)$ over the terminal alphabet $\Sigma^{\pm 1}$ such that $|\mathbb{A}(w)| \in O(n^3)$ and in G we have

$$h(a_1)h(a_2)\cdots h(a_n) = \mathrm{eval}(\mathbb{A}(w))\, h(b_1)\cdots h(b_m).$$

Proof of Claim 1. Let us take a word $w = a_1 \cdots a_n$ ($a_i \in \Gamma^{\pm 1}$). If $n = 0$, then we take for $\mathbb{A}(w)$ an SLP generating the empty string. Now assume that $n > 0$ and let

$$\begin{aligned} v &= a_1 \cdots a_{n-1}, \\ \mathrm{NF}(v) &= c_1 \cdots c_k, \text{ and} \\ \mathrm{NF}(w) &= b_1 \cdots b_m. \end{aligned}$$

There is a constant α (only depending on Q) such that $k \leq \alpha \cdot (n-1)$ and $m \leq k + \alpha \leq \alpha \cdot n$.

By induction, we can assume that we have already calculated an SLP $\mathbb{A}(v)$ over the terminal alphabet $\Sigma^{\pm 1}$ such that

$$h(a_1)h(a_2)\cdots h(a_{n-1}) = \mathrm{eval}(\mathbb{A}(v))h(c_1)\cdots h(c_k)$$

in G and $|\mathbb{A}(v)| \leq \delta \cdot (n-1)^3$, where δ is a constant, which can be fixed later. Hence

$$h(a_1)h(a_2)\cdots h(a_n) = \mathrm{eval}(\mathbb{A}(v))h(c_1)\cdots h(c_k)h(a_n)$$

in G. For the rest of the proof, we have to distinguish the cases $k \leq m$ and $k > m$. We only consider the case $k \leq m$, the case $k > m$ can be dealt similarly. So, assume that $k \leq m$. Since the automatic group Q satisfies the synchronous fellow traveller property, there exist a constant $\beta \in \mathbb{N}$ (depending only on Q) and words $r_0, \ldots, r_m \in (\Gamma^{\pm 1})^{\leq \beta}$ such that:

(1) $r_0 = \varepsilon$, $r_m = a_n$,
(2) $r_{i-1}b_i = c_i r_i$, i.e., $c_i = r_{i-1}b_i r_i^{-1}$ in Q for $1 \le i \le k$, and
(3) $r_{i-1}b_i = r_i$, i.e., $1 = r_{i-1}b_i r_i^{-1}$ in Q for $k < i \le m$.

In the group K the identities in (2) and (3) correspond to identities of the following form (when writing $h(r_i)$, we identify r_i with the element of Q it represents):

(1') $h(c_i) = p_i h(r_{i-1})h(b_i)h(r_i)^{-1}$ ($1 \le i \le k$)
(2') $1 = p_i h(r_{i-1})h(b_i)h(r_i)^{-1}$ ($k < i \le m$).

where $p_1, \dots, p_k \in (\Sigma^{\pm 1})^*$. Since in (1') and (2') there is only a finite number (depending only on Q) of different possibilities for $b_i, c_i \in \Gamma^{\pm 1}$, and $r_i \in (\Gamma^{\pm 1})^{\le \beta}$, we can write down a finite list of all possible candidates for the words p_i. In particular, there is a constant γ (depending only on Q and K) such that $p_1, \dots, p_k \in (\Sigma^{\pm 1})^{\le \gamma}$. Since $h(r_0) = h(1) = 1$ and $h(r_m) = h(a_n)$, we obtain

$$\begin{aligned} h(c_1)\cdots h(c_k)h(a_n) &= p_1 h(b_1)h(r_1)^{-1} p_2 h(r_1)h(b_2)h(r_2)^{-1}\cdots \\ &\qquad p_m h(r_{m-1})h(b_m)h(r_m)^{-1}h(a_n) \\ &= p_1 h(b_1)h(r_1)^{-1} p_2 h(r_1)h(b_2)h(r_2)^{-1}\cdots p_m h(r_{m-1})h(b_m) \end{aligned}$$

in G. We now shift the elements $h(b_i)$ and $h(r_i)^{-1}$ to the right (thereby, the $h(r_i)$ and $h(r_i)^{-1}$ cancel out each other) by applying the automorphisms of K defined by conjugation with these elements to the $p_i \in K$. Note that since the length of any word r_i is bounded by the fixed constant β, all applied automorphisms are taken from some fixed finite subset of $\mathrm{Aut}(K)$. By Lemma 2, we can compute an SLP $\mathbb{B}$ of size $O(m^2) \le O(n^2)$ such that

$$\begin{aligned} p_1 h(b_1)h(r_1)^{-1} p_2 h(r_1)h(b_2)h(r_2)^{-1}\cdots p_m h(r_{m-1})h(b_m) = \\ \mathrm{eval}(\mathbb{B})h(b_1)h(b_2)\cdots h(b_m) \text{ in } G. \end{aligned}$$

The size bound for $\mathbb{B}$ holds, since $p_1 \cdots p_m$ has length $O(m)$ and to each symbol in $p_1 \cdots p_m$ we apply $O(m)$ many automorphisms. We obtain

$$h(a_1)\cdots h(a_n) = \mathrm{eval}(\mathbb{A}(v))\mathrm{eval}(\mathbb{B})h(b_1)\cdots h(b_m) \text{ in } G.$$

Hence, we can take for $\mathbb{A}(w)$ an SLP which computes the concatenation of $\mathrm{eval}(\mathbb{A}(v))$ and $\mathrm{eval}(\mathbb{B})$. It follows that

$$|\mathbb{A}(w)| = |\mathbb{A}(v)| + |\mathbb{B}| + 1 \le \delta \cdot (n-1)^3 + O(n^2).$$

By choosing the constant δ large enough (depending on the constant hidden in the $O(n^2)$ term), we obtain $|\mathbb{A}(w)| \le \delta \cdot n^3$. This completes the proof of Claim 1.

Let us continue the proof of Proposition 4. Assume that w is a word over the generating set $\Sigma^{\pm 1} \cup h(\Gamma^{\pm 1})$ of G. Let

$$w = w_0 h(a_1) w_1 h(a_2) \cdots w_{n-1} h(a_n) w_n$$

with $w_i \in (\Sigma^{\pm 1})^*$ $(0 \leq i \leq n)$ and $a_i \in \Gamma^{\pm 1}$ $(1 \leq i \leq n)$. Let $\psi_i \in \mathrm{Aut}(K)$ $(1 \leq i \leq n)$ be the automorphism of K defined by $\psi_i(k) = h(a_i)kh(a_i)^{-1}$ and let $\chi_i = \psi_1 \cdots \psi_i$ for $0 \leq i \leq n$. Thus, $\chi_0 = \mathrm{id}_K$ and we have

$$\begin{aligned} w &= w_0 h(a_1) w_1 h(a_2) \cdots w_{n-1} h(a_n) w_n \\ &= \chi_0(w_0)\chi_1(w_1) \cdots \chi_n(w_n) h(a_1) h(a_2) \cdots h(a_n) \end{aligned}$$

in G. In the automatic quotient Q we have $\varphi(w) = a_1 \cdots a_n$. Hence, we first calculate in polynomial time the normal form $v = \mathrm{NF}(a_1 \cdots a_n)$. If $v \neq \varepsilon$, then we know that $w \neq 1$ in G. Hence assume that $v = \varepsilon$. By Claim 1, we can compute in polynomial time an SLP $\mathbb{A}$ over the terminal alphabet $\Sigma^{\pm 1}$ such that $h(a_1)h(a_2) \cdots h(a_n) = \mathrm{eval}(\mathbb{A})$ in G. Hence, $w = 1$ in G if and only if $\chi_0(w_0)\chi_1(w_1) \cdots \chi_n(w_n)\mathrm{eval}(\mathbb{A}) = 1$ in G, which (by Lemma 2) can be transfered in polynomial time into an instance of the compressed word problem for K. ☐

4 Upper bounds for compressed word problems

4.1 Finite extensions

Since every finite group is automatic, Proposition 4 applies to the case that the quotient Q is finite. In this situation, we even obtain a polynomial time reduction from the *compressed* word problem of G to the compressed word problem of K:

Theorem 1. *Assume that K is a finitely generated subgroup of the group G such that the index $[G : K]$ is finite. Then* $\mathrm{CWP}(G) \leq_m^P \mathrm{CWP}(K)$.

Proof. Let Γ be a finite generating set for K and let Σ be a finite generating set for G. Let $h : (\Sigma^{\pm 1})^* \to G$ be the canonical morphism. Let $Kg_1, \ldots, Kg_n$ be a list of the cosets of K, where w.l.o.g. $g_1 = 1$. Let $\mathcal{A}$ be the coset automaton of K. This is a finite automaton over the alphabet $\Sigma^{\pm 1}$ and with state set $\{Kg_1, \ldots, Kg_n\}$. The initial and final state is $K = Kg_1$ and there is a transition $Kg_i \xrightarrow{a} Kg_j$ $(a \in \Sigma^{\pm 1})$ if and only if $Kg_i a = Kg_j$. Note that this automaton accepts a word $w \in (\Sigma^{\pm 1})^*$ if and only if $h(w) \in K$. Since it can be checked in polynomial time whether the word generated by a given SLP is accepted by a given finite automaton (here, we even have a fixed automaton $\mathcal{A}$), we can check in polynomial time whether $h(\mathrm{eval}(\mathbb{A})) \in K$ for a given SLP $\mathbb{A}$.

Now let $\mathbb{A}$ be an SLP in Chomsky normal form over the alphabet $\Sigma^{\pm 1}$. We want to check whether $\mathrm{eval}(\mathbb{A}) = 1$ in G. First, we check in polynomial time, whether $h(\mathrm{eval}(\mathbb{A})) \in K$. If not, we reject immediately. Otherwise, we will construct an SLP $\mathbb{B}$ over the generating set $\Gamma^{\pm 1}$ of K, which computes the same group element as $\mathbb{A}$. Then we can apply an algorithm for $\mathrm{CWP}(K)$.

Let V be the set of nonterminals of $\mathbb{A}$ and let S be the start nonterminal of $\mathbb{A}$. The set of nonterminals of $\mathbb{B}$ is the set of triples

$$W = \{[g_i, A, g_j^{-1}] \mid A \in V, 1 \leq i, j \leq n, g_i h(\mathrm{eval}(A)) g_j^{-1} \in K\}.$$

By the above observation, this set can be computed in polynomial time. Now, let us introduce the production for the nonterminal $[g_i, A, g_j^{-1}] \in W$. First, assume that the production for A is $A \to a$, where $a \in \Sigma^{\pm 1}$. Hence, $g_i a g_j^{-1} \in K$, and we introduce the production $[g_i, A, g_j^{-1}] \to w$, where $w \in (\Gamma^{\pm 1})^*$ is such that $h(w) = g_i a g_j^{-1}$. Now assume that the production for A is of the form $A \to BC$. Assume that $g_i h(\mathrm{eval}(B))$ belongs to the coset Kg_k. Thus, $g_i h(\mathrm{eval}(B)) g_k^{-1} \in K$, i.e., $[g_i, B, g_k^{-1}] \in W$. We introduce the production

$$[g_i, A, g_j^{-1}] \to [g_i, B, g_k^{-1}][g_k, C, g_j^{-1}].$$

Note that

$$g_i h(\mathrm{eval}(A)) g_j^{-1} = g_i h(\mathrm{eval}(B)) g_k^{-1} g_k h(\mathrm{eval}(C)) g_j^{-1}.$$

Hence, since $g_i h(\mathrm{eval}(A)) g_j^{-1}$ and $g_i h(\mathrm{eval}(B)) g_k^{-1}$ both belong to the subgroup K, we also have $g_k h(\mathrm{eval}(C)) g_j^{-1} \in K$, i.e., $[g_k, C, g_j^{-1}] \in W$. Finally, let $[g_1, S, g_1^{-1}] = [1, S, 1]$ be the start nonterminal of $\mathbb{B}$. It is easy to prove that for every nonterminal $[g_i, A, g_j^{-1}] \in W$, $\mathrm{eval}_{\mathbb{B}}([g_i, A, g_j^{-1}])$ represents the group element $g_i h(\mathrm{eval}_{\mathbb{A}}(A)) g_j^{-1}$. Thus, $\mathrm{eval}(\mathbb{A}) = 1$ in G if and only if $\mathrm{eval}(\mathbb{B}) = 1$ in K, which is an instance of $\mathrm{CWP}(K)$. This proves the theorem. $\square$

The reducibility relation $\leq_m^P$ in Theorem 1 cannot be replaced by the stronger relation $\leq_m^{\log}$ (unless P = L) because there exists a finite group G with a P-complete compressed word problem [3] (take $K = 1$ in Theorem 1).

4.2 Free products

The aim of this section is to prove the following theorem:

Theorem 2. *Assume that* $G = G_1 * G_2$. *Then* $\mathrm{CWP}(G) \leq_T^P (\mathrm{CWP}(G_1), \mathrm{CWP}(G_2))$.

Proof. For the proof of the theorem, it is useful, to introduce a special kind of composition systems, which we call *2-level composition systems*. A 2-level composition system over the terminal alphabet Γ is a tuple $\mathbb{A} = (\mathbb{B}, V_u, V_\ell)$, where $\mathbb{B} = (V, \Gamma, S, P)$ is a composition system and $V = V_u \cup V_\ell$ ($V_u \cap V_\ell = \emptyset$) is a partition of the set of nonterminals into the set of upper-level nonterminals V_u and the set of lower-level nonterminals V_ℓ such that: $S \in V_u$ and for every production $(A \to w) \in P$ we have either (i) $A \in V_\ell$ and $w \in (V_\ell \cup \Gamma)^*$ or (ii) $A \in V_u$ and ($w \in V^*$ or $w = B[i:j]$ for some $B \in V_u$ and $i, j \in \mathbb{N}$). For $A \in V$ we set $\mathrm{eval}_{\mathbb{A}}(A) = \mathrm{eval}_{\mathbb{B}}(A)$ and $\mathrm{eval}(\mathbb{A}) = \mathrm{eval}(\mathbb{B})$. Define the composition system $\mathbb{A}_u = (V_u, V_\ell, \{(A \to w) \in P \mid A \in V_u\}, S)$ and let $\mathrm{ueval}(\mathbb{A}) = \mathrm{eval}(\mathbb{A}_u) \in V_\ell^*$. Using Hagenah's result [21], every 2-level composition system can be transformed in polynomial time into an equivalent SLP.

Let Σ_i be a finite generating set for G_i ($i \in \{1, 2\}$), where $\Sigma_1 \cap \Sigma_2 = \emptyset$. Let $\mathbb{A}$ be an SLP over the terminal alphabet $\Sigma_1^{\pm 1} \cup \Sigma_2^{\pm 1}$. Our goal is to construct in polynomial time a 2-level composition system $\mathbb{A}'$ such that $\mathrm{eval}(\mathbb{A})$ and $\mathrm{eval}(\mathbb{A}')$ represent the same group element of $G_1 * G_2$ but $\mathrm{eval}(\mathbb{A}')$ is irreducible in $G_1 * G_2$. Then, $\mathrm{eval}(\mathbb{A}) = 1$ in $G_1 * G_2$ if and only if $\mathrm{eval}(\mathbb{A}') = \varepsilon$.

The construction of $\mathbb{A}'$ follows the strategy for free groups from [28]. In a first step we check for every nonterminal A of $\mathbb{A}$ whether either $\mathrm{eval}(A) \in (\Sigma_1^{\pm 1})^*$ and $\mathrm{eval}(A) = 1$ in G_1 or $\mathrm{eval}(A) \in (\Sigma_2^{\pm 1})^*$ and $\mathrm{eval}(A) = 1$ in G_2 (for this, we have to solve instances of $\mathrm{CWP}(G_1)$ and $\mathrm{CWP}(G_2)$). If this is true, then we eliminate the nonterminal A from $\mathbb{A}$ by replacing A in all right-hand sides by ε. We iterate this process as long as we can eliminate nonterminals. Let us denote the resulting SLP again by $\mathbb{A}$. W.l.o.g. we can assume that $\mathbb{A}$ is in Chomsky normal form.

Let V be the set of nonterminals of $\mathbb{A}$. We define a partition $V = V_u \cup V_\ell$ of V as follows: $V_\ell = \{A \in V \mid \mathrm{eval}(A) \in (\Sigma_1^{\pm 1})^* \cup (\Sigma_2^{\pm 1})^*\}$ and $V_u = V \setminus V_\ell$. This defines a 2-level composition system $(\mathbb{A}, V_u, V_\ell)$. In the rest of the proof we will manipulate this 2-level composition system such that at the end we obtain a 2-level composition system $\mathbb{A}'$ with the property that (i) it generates the same group element of $G_1 * G_2$ and (ii) every nonterminal of $\mathbb{A}'$ generates a word which is irreducible in $G_1 * G_2$. In the following, the notions eval, ueval, V_u, V_ℓ will refer to the current 2-level composition system. Note that initially, for $A \in V_\ell$ we have: $\mathrm{eval}(A) \neq 1$ in G_1 in case $\mathrm{eval}(A) \in (\Sigma_1^{\pm 1})^*$ and $\mathrm{eval}(A) \neq 1$ in G_2 in case $\mathrm{eval}(A) \in (\Sigma_2^{\pm 1})^*$. In particular, $\mathrm{eval}(A) \in (\Sigma_1^{\pm 1})^+ \cup (\Sigma_2^{\pm 1})^+$. This property will be preserved during the construction of $\mathbb{A}'$.

A word $u = A_1 \cdots A_n$ ($A_i \in V_\ell$ for $1 \leq i \leq n$) is called *irreducible*, if for all $1 \leq i < n$: $\mathrm{eval}(A_i) \in (\Sigma_1^{\pm 1})^+ \Leftrightarrow \mathrm{eval}(A_{i+1}) \in (\Sigma_2^{\pm 1})^+$. Since every word $\mathrm{eval}(A_i)$ ($1 \leq i \leq n$) neither represents the 1 of G_1 (if $\mathrm{eval}(A_1) \in (\Sigma_1^{\pm 1})^+$) nor of G_2 (if $\mathrm{eval}(A_1) \in (\Sigma_1^{\pm 1})^+$), this means that $\mathrm{eval}(A_1) \cdots \mathrm{eval}(A_n)$ is irreducible in $G_1 * G_2$.

For words $u, v \in V_\ell^*$ we write $\mathrm{cancel}(u, v)$ if $u = A_n \cdots A_1$, $v = B_1 \cdots B_n$ for some $n \geq 0$ and $A_1, \ldots, A_n, B_1, \ldots, B_n \in V_\ell$ and for every $1 \leq i \leq n$ there is $j \in \{1, 2\}$ such that $\mathrm{eval}(A_i), \mathrm{eval}(B_i) \in (\Sigma_j^{\pm 1})^+$ and $\mathrm{eval}(A_i)\mathrm{eval}(B_i) = 1$ in G_j.

Claim: On a Turing machine with oracle access to $\mathrm{CWP}(G_1)$ and $\mathrm{CWP}(G_2)$ we can check in polynomial time whether $\mathrm{cancel}(\mathrm{eval}(\mathbb{C}), \mathrm{eval}(\mathbb{D}))$ for given composition systems $\mathbb{C}$ and $\mathbb{D}$ over the terminal alphabet V_ℓ.

Proof of the Claim: By Hagenah's result [21] we may assume that $\mathbb{C}$ and $\mathbb{D}$ are SLPs. From $\mathbb{D}$ we can easily compute an SLP $\mathbb{D}'$ such that $\mathrm{eval}(\mathbb{D}')$ is the string that results from reversing $\mathrm{eval}(\mathbb{D})$. Define a mapping $f_1 : V_\ell \to V_\ell$ as follows: Fix an order $\preceq$ on V_ℓ. Then, for $A \in V_\ell$, $f_1(A)$ is the smallest $B \in V_\ell$ such that $\mathrm{eval}(A) = \mathrm{eval}(B)$ in either G_1 or G_2. Moreover, define a second mapping $f_2 : V_\ell \to V_\ell$ as follows: For $A \in V_\ell$, $f_2(A)$ is the smallest $B \in V_\ell$ such that $\mathrm{eval}(A) = \mathrm{eval}(B)^{-1}$ in either G_1 or G_2. Note that the mappings f_1 and f_2 can be computed in polynomial time on a Turing machine with oracle access to $\mathrm{CWP}(G_1)$ and $\mathrm{CWP}(G_2)$. We extend f_1 and f_2 to morphisms on V_ℓ^*. Now we can easily compute SLPs $\mathbb{C}'$ and $\mathbb{D}''$ such that $\mathrm{eval}(\mathbb{C}') = f_1(\mathrm{eval}(\mathbb{C}))$ and $\mathrm{eval}(\mathbb{D}'') = f_2(\mathrm{eval}(\mathbb{D}'))$. Then, $\mathrm{cancel}(\mathrm{eval}(\mathbb{C}), \mathrm{eval}(\mathbb{D}))$ if and only if $\mathbb{C}'$ and $\mathbb{D}''$ generate the same strings over the alphabet V_ℓ. This can be checked in polynomial time [34].

Let us now construct the 2-level composition system $\mathbb{A}'$. In a bottom-up process, similarly to [28], we will process every upper-level nonterminal from V_u. Thereby, we will enforce that for every $A \in V_u$, $\mathrm{ueval}(A) \in V_\ell^*$ is irreducible and hence $\mathrm{eval}(A)$ is irreducible in $G_1 * G_2$. So, assume that $A \to BC$ is a production of $\mathbb{A}$ with $A \in V_u$ and that B and C are either from V_ℓ or were already processed. There are four possible cases: (i) $B, C \in V_\ell$, (ii) $B, C \in V_u$, (iii) $B \in V_u$, $C \in V_\ell$, and (iv) $B \in V_\ell$, $C \in V_u$. In case (i), we must have either $\mathrm{eval}(B) \in (\Sigma_1^{\pm 1})^+$ and $\mathrm{eval}(C) \in (\Sigma_2^{\pm 1})^+$

or $\mathrm{eval}(B) \in (\Sigma_2^{\pm 1})^+$ and $\mathrm{eval}(C) \in (\Sigma_1^{\pm 1})^+$, because otherwise A would belong to V_ℓ. Hence, $\mathrm{ueval}(A) = BC$ is irreducible and we do not have to modify the production $A \to BC$ for A. From the cases (ii)–(iv) we will only consider case (ii), the other two cases are simpler to deal with.

The words $u = \mathrm{ueval}(B) \in V_\ell^*$ and $v = \mathrm{ueval}(C) \in V_\ell^*$ are already irreducible. We now determine the maximal amount of cancellation between u and v, when interpreting the symbols in these words as elements from $G_1 \cup G_2$. Assume that $u = B_n \cdots B_1$ and $v = C_1 \cdots C_m$ for $B_1, \ldots, B_n, C_1, \ldots, C_m \in V_\ell$. We first determine in polynomial time the symbols $B_1, C_1 \in V_\ell$. If either $\mathrm{eval}(B_1) \in (\Sigma_1^{\pm 1})^+$ and $\mathrm{eval}(C_1) \in (\Sigma_2^{\pm 1})^+$ or $\mathrm{eval}(B_1) \in (\Sigma_2^{\pm 1})^+$ and $\mathrm{eval}(C_1) \in (\Sigma_1^{\pm 1})^+$ then $\mathrm{ueval}(A)$ is already irreducible and we do not have to modify the production $A \to BC$. Otherwise, using binary search over the range $\{1, \ldots, \min(n, m)\}$ we find the largest number i such that $\mathrm{cancel}(B_i \cdots B_1, C_1 \cdots C_i)$. Note that we can write down composition systems of polynomial size generating the words $B_i \cdots B_1$ and $C_1 \cdots C_i$. Hence, by the above claim, the number i can be found in polynomial time on a Turing machine with oracle access to $\mathrm{CWP}(G_1)$ and $\mathrm{CWP}(G_2)$. Next, we have to distinguish the following cases:

– $i = n = m$: We replace the production $A \to BC$ by $A \to \varepsilon$.
– $i = n < m$: We replace the production $A \to BC$ by $A \to C[i + 1 :]$.
– $i = m < n$: symmetric to the previous case.
– $i < n, i < m$: we add a new lower-level nonterminal D to V_ℓ together with the production $D \to B_{i+1}C_{i+1}$. Note that $\mathrm{eval}(B_{i+1})\mathrm{eval}(C_{i+1}) \neq 1$ (either in G_1 or in G_2) because otherwise i would not be the largest number such that $\mathrm{cancel}(B_i \cdots B_1, C_1 \cdots C_i)$. Moreover, the production $A \to BC$ is replaced by $A \to B[: n - i - 1]DC[i + 2 :]$.

This concludes the construction of the 2-level composition system $\mathbb{A}'$. □

Again, the reducibility relation $\leq_T^P$ in Theorem 2 cannot be replaced by the stronger relation $\leq_m^{\log}$ (unless P = NC)[3] because the compressed word problem for $\mathbb{Z} * \mathbb{Z}$ is P-complete [28], whereas the compressed word problem for $\mathbb{Z}$ is easily seen to be in NC.

4.3 Graph groups and graph products

For this section, we need the material from Section 2.2. Let us fix an independence alphabet (Σ, I). Define a trace rewriting system R over $\mathbb{M}(\Sigma^{\pm 1}, I)$ as follows:

$$R = \{([aa^{-1}]_I, [\varepsilon]_I) \mid a \in \Sigma^{\pm 1}\}. \tag{1}$$

One can show that R is terminating and confluent and that for all $u \in \mathbb{M}(\Sigma^{\pm 1}, I)$: $u = 1$ in $\mathbb{G}(\Sigma, I)$ if and only if $\mathrm{NF}_R(u) = [\varepsilon]_I$, i.e., $u \xrightarrow{*}_R [\varepsilon]_I$ [12]. This leads to a linear time solution for the word problem of $\mathbb{G}(\Sigma, I)$ [12, 42].

Example 3. Let (Σ, I) be the following independence alphabet:

[3] NC denotes Nick's class — the class of all problems that can be solved with polynomially many processors in polylogarithmic time.

$$\begin{tikzcd} b \arrow[r, dash] & d \arrow[r, dash] & a \arrow[r, dash] & c \end{tikzcd}$$

An example for a derivation using the trace rewriting system R is:

$$\begin{aligned} &[b^{-1}adc^{-1}a^{-1}cbd^{-1}]_I = \\ &[b^{-1}ada^{-1}c^{-1}cbd^{-1}]_I \to_R \\ &[b^{-1}ada^{-1}bd^{-1}]_I = \\ &[b^{-1}aa^{-1}dbd^{-1}]_I \to_R \\ &[b^{-1}dbd^{-1}]_I = \\ &[b^{-1}bdd^{-1}]_I \to_R^2 [\varepsilon]_I \end{aligned}$$

The fact that $[b^{-1}adc^{-1}a^{-1}cbd^{-1}]_I \xrightarrow{*}_R [\varepsilon]_I$ becomes quite obvious, when looking at the dependence graph of the trace $[b^{-1}adc^{-1}a^{-1}cbd^{-1}]_I$:

$$\begin{tikzcd} b^{-1} \arrow[r] \arrow[dr] & a \arrow[r] & a^{-1} \arrow[r] & b \\ d \arrow[r] & c^{-1} \arrow[r] & c \arrow[r] \arrow[ur] & d^{-1} \end{tikzcd}$$

This graph can be reduced to the empty graph by successively canceling nodes with inverse labels, which are moreover connected by an edge.

In this section, we will show that the compressed word problem for $\mathbb{G}(\Sigma, I)$ can be solved in polynomial time. We follow our strategy for free groups [28]. For a given SLP $\mathbb{A}$ over the terminal alphabet $\Sigma^{\pm 1}$ we construct an extended composition system (see Section 2.3) $\mathbb{B}$ such that $[\text{eval}(\mathbb{B})]_I = \text{NF}_R([\text{eval}(\mathbb{A})]_I)$. For this we will accumulate the productions of $\mathbb{B}$ in the same way as in the free group case. We start with all productions from $\mathbb{A}$ of the form $A \to a$. Now assume that $\mathbb{A}$ contains a production $A \to BC$ and that $\mathbb{B}$ already contains enough productions such that $[\text{eval}_{\mathbb{B}}(B)]_I = \text{NF}_R([\text{eval}_{\mathbb{A}}(B)]_I)$ and $[\text{eval}_{\mathbb{B}}(C)]_I = \text{NF}_R([\text{eval}_{\mathbb{A}}(C)]_I)$. We have to add a rule for the nonterminal A such that

$$[\text{eval}_{\mathbb{B}}(A)]_I = \text{NF}_R([\text{eval}_{\mathbb{B}}(B)\text{eval}_{\mathbb{B}}(C)]_I). \tag{2}$$

Intuitively, R-reduction steps in the trace $[\text{eval}_{\mathbb{B}}(B)\text{eval}_{\mathbb{B}}(C)]_I$ can only occur at the border between the prefix $[\text{eval}_{\mathbb{B}}(B)]_I$ and the suffix $[\text{eval}_{\mathbb{B}}(C)]_I$, because both these traces are irreducible w.r.t. R. In other words, some suffix of $[\text{eval}_{\mathbb{B}}(B)]_I$ will cancel against some prefix of $[\text{eval}_{\mathbb{B}}(C)]_I$. We have to determine and cut away on the level of extended composition systems this suffix and prefix, respectively. At this point, the construction becomes more involved than for the free groups case. We need two lemmas:

Lemma 4. *For two given extended composition systems $\mathbb{A}$ and $\mathbb{B}$ it can be checked in polynomial time whether* $\text{eval}(\mathbb{A}) \preceq_I \text{eval}(\mathbb{B})$.

Proof. By Lemma 3 we can assume that $\mathbb{A}$ and $\mathbb{B}$ are SLPs. Let $(\Sigma_i)_{1 \le i \le n}$ be a clique covering for the dependence alphabet (Σ, D). We compute in polynomial time SLPs $\mathbb{A}_i$ and $\mathbb{B}_i$ ($1 \le i \le n$) such that $\text{eval}(\mathbb{A}_i) = \pi_{\Sigma_i}(\text{eval}(\mathbb{A}))$ and $\text{eval}(\mathbb{B}_i) = \pi_{\Sigma_i}(\text{eval}(\mathbb{B}))$. By Lemma 1 it suffices to check whether $\text{eval}(\mathbb{A}_i)$ is a prefix of $\text{eval}(\mathbb{B}_i)$ for all $1 \le i \le n$. But this can be easily reduced to an equivalence check: Compute $n_i = |\text{eval}(\mathbb{A}_i)|$ and an SLP $\mathbb{C}_i$ with $\text{eval}(\mathbb{C}_i) = \text{eval}(\mathbb{B}_i)[:n_i]$. Finally check whether $\text{eval}(\mathbb{C}_i) = \text{eval}(\mathbb{A}_i)$ for all $1 \le i \le n$. ☐

Lemma 5. *Let* $p, q \in \mathbb{M}(\Sigma^{\pm 1}, I)$. *If* $p, q \in \mathrm{IRR}(R)$, *then*

$$\mathrm{NF}_R(pq) = (p^{-1} \setminus q)^{-1}(q \setminus p^{-1}).$$

Proof. Let $p, q \in \mathrm{IRR}(R)$. By [13, Lemma 13], in the trace monoid $\mathbb{M}(\Sigma^{\pm 1}, I)$ there exist factorizations $p = xr$ and $q = r^{-1}y$ such that $\mathrm{NF}_R(pq) = xy$. Moreover, r is the trace with maximal length such that p and q can be written as $p = xr$ and $q = r^{-1}y$. It follows that $r = (p^{-1} \sqcap q)^{-1}$, $x = (p^{-1} \setminus q)^{-1}$, and $y = q \setminus p^{-1}$. $\square$

Example 4. Let (Σ, I) be the independence alphabet from Example 3. Let

$$p = [cbdcd^{-1}b^{-1}a]_I \in \mathrm{IRR}(R) \text{ and}$$
$$q = [da^{-1}baac^{-1}bd^{-1}]_I = [da^{-1}bc^{-1}d^{-1}aab]_I \in \mathrm{IRR}(R).$$

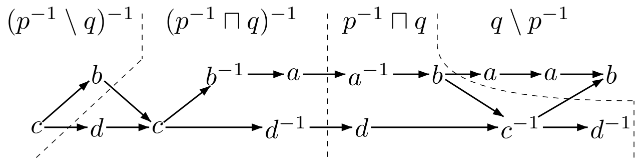
Hence,

$$p^{-1} = [a^{-1}bdc^{-1}d^{-1}b^{-1}c^{-1}]_I = [da^{-1}bc^{-1}d^{-1}b^{-1}c^{-1}]_I$$

and we see that

$$p^{-1} \sqcap q = [da^{-1}bc^{-1}d^{-1}]_I,$$
$$p^{-1} \setminus q = [b^{-1}c^{-1}]_I, \ (p^{-1} \setminus q)^{-1} = [cb]_I, \text{ and}$$
$$q \setminus p^{-1} = [aab]_I.$$

Hence, $\mathrm{NF}_R(pq) = (p^{-1} \setminus q)^{-1}(q \setminus p^{-1}) = [cb]_I[aab]_I = [cbaab]_I$. This fact can be also visualized in the dependence graph of the trace pq, which looks as follows:



From Lemma 5, it follows that in order to compute a production for the nonterminal A such that (2) holds, we basically have to solve the following problem: For a given extended composition system $\mathbb{B}$ with nonterminals B and C, construct a production for a new nonterminal A such that

$$[\mathrm{eval}_{\mathbb{B}}(A)]_I = [\mathrm{eval}_{\mathbb{B}}(B)]_I \setminus [\mathrm{eval}_{\mathbb{B}}(C)]_I.$$

Then, productions for $([\mathrm{eval}_{\mathbb{B}}(B)]_I^{-1} \setminus [\mathrm{eval}_{\mathbb{B}}(C)]_I)^{-1}$ and $[\mathrm{eval}_{\mathbb{B}}(C)]_I \setminus [\mathrm{eval}_{\mathbb{B}}(B)]_I^{-1}$ can be calculated in the same way.

We first need some concepts concerning dependence graphs. In the following, it is not relevant that the alphabet is of the form $\Sigma^{\pm 1}$. Hence, let us fix a dependence alphabet (Σ, D) for the further consideration. For $a \in \Sigma$ let $\mathcal{P}_a$ be the set all simple paths in the dependence alphabet (Σ, D) (viewed as an undirected graph) that start in the node a (a path is simple, if it does not visit a node twice). The path, which only consists of the node a belongs to $\mathcal{P}_a$. Note that $\mathcal{P}_a$ is finite and its size only depends on the dependence alphabet (Σ, D).

Take a string $s \in \Sigma^*$ and a set of position $J \subseteq \{1, \dots, |s|\}$ in s. We are looking for a compact representation of the set of all positions k such that there exists a directed path in the dependence graph D_s from some position $j \in J$ to position k. Let us define the set $\mathrm{CP}(J) \subseteq \{1, \dots, |s|\}$ of *critical positions* as follows: Let $j \in J$ and $a = s[j]$ For each path $p \in \mathcal{P}_a$, define the position $\mathrm{pos}(j, p)$ inductively as follows: If $p = a$, then $\mathrm{pos}(j, p) = j$. If $p = a, q$, where q is a simple path starting in some node $b \in D(a)$, then $\mathrm{pos}(j, p) = \mathrm{pos}(j', q)$, where j' is the smallest position $j' > j$ such that $s[j'] = b$. If such a j' does not exist, then $\mathrm{pos}(j, p)$ is undefined. Finally, let us set

$$\mathrm{CP}(J) = \{\mathrm{pos}(j, p) \mid j \in J, p \in \mathcal{P}_{s[j]}, \mathrm{pos}(j, p) \text{ is defined}\}$$

and for every $k \in \mathrm{CP}(J)$ set

$$\Gamma_{J,k} = \{s[\ell] \mid \ell \in \mathrm{CP}(J), \ell \leq k\}.$$

The set $\mathrm{CP}(J)$ contains positions, which are redundant in order to fulfill the crucial Lemma 6 below. We therefore define a reduced set of critical positions:

$$\mathrm{RCP}(J) = \{k \in \mathrm{CP}(J) \mid \neg\exists \ell \in \mathrm{CP}(J) : \ell < k, \Gamma_{J,\ell} = \Gamma_{J,k}\}.$$

Note that $\Gamma_{J,\ell} \subsetneq \Gamma_{J,k}$ for all $\ell, k \in \mathrm{RCP}(J)$ with $\ell < k$. This implies $|\mathrm{RCP}(J)| \leq |\Sigma|$.

In case J contains only a single position j, we write $\mathrm{CP}(j)$ and $\Gamma_{j,\ell}$ instead of $\mathrm{CP}(J)$ and $\Gamma_{J,\ell}$, respectively.

Lemma 6. *Let $s \in \Sigma^*$ and $J \subseteq \{1, \dots, |s|\}$. Then for every position k the following two properties are equivalent:*

(1) There exists a directed path in the dependence graph D_s from some position $j \in J$ to position k.
(2) There exists $\ell \in \mathrm{RCP}(J)$ such that $\ell \leq k$ and $s[k] \in \Gamma_{J,\ell}$.

Proof. First assume that there exists $\ell \in \mathrm{RCP}(J)$ such that $\ell \leq k$ and $s[k] \in \Gamma_{J,\ell}$. The latter implies that $s[k] = s[\ell']$ for some $\ell' \in \mathrm{CP}(J)$ with $\ell' \leq \ell \leq k$. Since $\ell' \in \mathrm{CP}(J)$, we have $\ell' = \mathrm{pos}(j, p)$ for some $j \in J$ and some path p in (Σ, D). But this implies that there exists a directed path in D_s from position j to position ℓ' in D_s. Finally, since $s[k] = s[\ell']$ and $\ell' \leq k$, either $k = \ell'$ or there exists an edge from ℓ' to k in D_s. Thus, there exists a directed path in the dependence graph D_s from j to k.

Now assume that there exists a directed path in the dependence graph D_s from some position $j \in J$ to position k. If $s[j] \neq s[k]$, then we can assume that this directed path corresponds to a simple path p in (Σ, D). It follows that $k \geq \mathrm{pos}(j, p)$ and $s[k] = s[\mathrm{pos}(j, p)]$ (if $s[j] = s[k]$, then we choose for p the path, which only consists of $s[j]$). Hence, there exists $\ell' \in \mathrm{CP}(J)$ such that $\ell' \leq k$ and $s[k] \in \Gamma_{J,\ell'}$ (take $\ell' = \mathrm{pos}(j, p)$). But then there exists $\ell \in \mathrm{RCP}(J)$ such that $\ell \leq \ell' \leq k$ and $s[k] \in \Gamma_{J,\ell}$. □

Note that condition (2) in Lemma 6 can be replaced by: For $\ell = \max(\mathrm{RCP}(J) \cap \{1, \dots k\})$ it holds $s[k] \in \Gamma_{J,\ell}$. This is true, because $\Gamma_{J,\ell} \subsetneq \Gamma_{J,k}$ for all $\ell, k \in \mathrm{RCP}(J)$ with $\ell < k$.

Let us now come back to the problem of constructing an extended composition system for $[\mathrm{eval}_{\mathbb{B}}(B)]_I \setminus [\mathrm{eval}_{\mathbb{B}}(C)]_I$. Let us first solve this problem for uncompressed

```
$i := 1;$                                                  (stores a position from $\{1,\dots,|s|\}$)
$u := \varepsilon;$                                                            (stores a string)
$v := \varepsilon;$                                                            (stores a string)
$\mathrm{RCP} := \{|s|+1\};$                                     (stores a subset of $\{1,\dots,|s|\}$)
$\mathit{\Gamma} := \emptyset;$                                                (stores a subset of $\mathit{\Sigma}$)
while $i \leq |s|$ do
    $j := \max\{j \leq \min(\mathrm{RCP})-1 \mid u\,\pi_{\mathit{\Sigma}\setminus\mathit{\Gamma}}(s[i,j]) \preceq_I t\};$                              (*)
    $u := u\,\pi_{\mathit{\Sigma}\setminus\mathit{\Gamma}}(s[i,j]);$
    $v := v\,\pi_{\mathit{\Gamma}}(s[i:j])\,s[j+1];$                        (let us set here $s[|s|+1]=\varepsilon$)
    if $j < \min(\mathrm{RCP})-1$ then                           (we obtain new critical points)
        for all $k \in \mathrm{CP}(j+1)$ do $\mathit{\Gamma}(k) := \mathit{\Gamma} \cup \mathit{\Gamma}_{j+1,k}$ endfor
        $\mathrm{RCP} := \mathrm{RCP} \cup \mathrm{CP}(j+1);$
        for all $k \in \mathrm{RCP}$ do $\mathit{\Gamma}(k) := \bigcup\{\mathit{\Gamma}(\ell) \mid \ell \in \mathrm{RCP}, \ell \leq k\}$ endfor
        $\mathrm{RCP} := \{k \in \mathrm{RCP} \mid \neg\exists \ell \in \mathrm{RCP} : \ell < k, \mathit{\Gamma}(\ell) = \mathit{\Gamma}(k)\};$
    endif
    $\mathit{\Gamma} := \mathit{\Gamma}(j+1);$                                                        (**)
    $i := j+2;$
    $\mathrm{RCP} := \mathrm{RCP} \setminus \{j+1\};$
endwhile
```

Fig. 1. An algorithm for computing $[s]_I \sqcap [t]_I$ and $[s]_I \setminus [t]_I$

strings. Then we will argue that our algorithm leads to a polynomial time algorithm for compressed input strings.

How can we compute for two given words $s, t \in \mathit{\Sigma}^*$ words $u, v \in \mathit{\Sigma}^*$ such that

$$[u]_I = [s]_I \sqcap [t]_I \text{ and } [v]_I = [s]_I \setminus [t]_I?$$

In the algorithm in Figure 1 we accumulate the strings u and v by determining for every position from $\{1,\dots,|s|\}$ (viewed as a node of the dependence graph D_s) whether it belongs to $[u]_I$ or $[v]_I$. For this, we will store a current position i in the string s, which will increase during the computation. Initially, we set $i := 1$ and $u := \varepsilon$, $v := \varepsilon$.

For a set of positions $K \subseteq \{1,\dots,|s|\}$ let us define the string $s{\restriction}K = s[i_1]\cdots s[i_k]$, where $i_1 < i_2 < \cdots < i_k$ and $K = \{i_1,\dots,i_k\}$. Consider a specific iteration of the while-loop in Figure 1 and let i denote the value of the corresponding program variable at the beginning of the iteration. Assume that $J \subseteq \{1,\dots,i-1\}$ is the set all positions from $\{1,\dots,i-1\}$, which belong to the difference $[s]_I \setminus [t]_I$, i.e., they do not belong to the common prefix $[s]_I \sqcap [t]_I$. Thus, $[s{\restriction}(\{1,\dots,i-1\} \setminus J)]_I$ is a trace prefix of $[s]_I \sqcap [t]_I$. If $i, u, v, \mathrm{RCP}, \mathit{\Gamma}$, and $\mathit{\Gamma}(k)$ denote the values of the corresponding program variables at the beginning of the iteration, then the algorithm will maintain the following relationships as invariants (the set J is defined as above, it is not stored by the algorithm):

- $u = s{\restriction}(\{1,\dots,i-1\} \setminus J)$, $v = s{\restriction}J$,
- $\mathrm{RCP} = (\{i,\dots,|s|\} \cap \mathrm{RCP}(J)) \cup \{|s|+1\}$
- $\mathit{\Gamma} = \emptyset$ if $\{0,\dots,i-1\} \cap \mathrm{RCP}(J) = \emptyset$, otherwise $\mathit{\Gamma} = \mathit{\Gamma}_{J,\ell}$, where ℓ is the maximum of $\{0,\dots,i-1\} \cap \mathrm{RCP}(J)$,
- for every $k \in \{i,\dots,|s|\} \cap \mathrm{RCP}(J)$, $\mathit{\Gamma}(k) = \mathit{\Gamma}_{J,k}$.

We put the imaginary position $|s|+1$ in the set RCP In order to save some if-branchings in the algorithm.

In each iteration of the while-loop, we investigate the subword of s from position i to the next critical position from the set RCP, and we determine for each position from some initial segment of this interval, whether it belongs to $[s]_I \sqcap [t]_I$ or $[s]_I \setminus [t]_I$. More precisely, we search for the largest position $j \leq \min(\mathrm{RCP}) - 1$ such that $u\pi_{\Sigma\setminus\Gamma}(s[i,j]) \preceq_I t$. Recall that $u = s\restriction(\{1,\ldots,i-1\} \setminus J)$ is the already collected part of the common trace prefix. We update u and v by $u := u\pi_{\Sigma\setminus\Gamma}(s[i,j])$ and $v := v\pi_\Gamma(s[i:j])s[j+1]$. The correctness of this step is expressed in the following lemma:

Lemma 7. *Assume that the following is given:*

- $i \in \{1,\ldots,|s|\}$
- $J \subseteq \{1,\ldots,i-1\}$ *is the set of all positions from* $\{1,\ldots,i-1\}$*, which belong to the trace difference* $[s]_I \setminus [t]_I$ *(thus,* $\{1,\ldots,i-1\} \setminus J$ *is downward-closed in* D_s *and* $[s\restriction(\{1,\ldots,i-1\} \setminus J)]_I$ *is a trace prefix of* $[s]_I \sqcap [t]_I$*).*
- $\Gamma = \emptyset$ *if* $\{0,\ldots,i-1\} \cap \mathrm{RCP}(J) = \emptyset$ *(this is equivalent to* $J = \emptyset$*), otherwise* $\Gamma = \Gamma_{J,\ell}$*, where* $\ell = \max(\{0,\ldots,i-1\} \cap \mathrm{RCP}(J))$.
- j *is the maximal position such that* $j \leq \min((\{i,\ldots,|s|\} \cap \mathrm{RCP}(J)) \cup \{|s|+1\}) - 1$ *and* $s\restriction(\{1,\ldots,i-1\} \setminus J)\, \pi_{\Sigma\setminus\Gamma}(s[i,j]) \preceq_I t$.

Then a position $p \in \{i,\ldots,j\}$ *belongs to the common trace prefix* $[s]_I \sqcap [t]_I$ *if and only if* $s[p] \notin \Gamma$.

Proof. If $s[p] \in \Gamma$, then by Lemma 6 there exists a path in D_s from some position in J to position p. Since positions in J do not belong to $[s]_I \sqcap [t]_I$, p cannot belong to $[s]_I \sqcap [t]_I$ as well. Now consider the set of positions $A = \{p \in \{i,\ldots,j\} \mid s[p] \notin \Gamma\}$. We claim that $(\{1,\ldots,i-1\} \setminus J) \cup A$ is a downward-closed subset of D_s. Since $s\restriction((\{1,\ldots,i-1\} \setminus J) \cup A) = s\restriction(\{1,\ldots,i-1\} \setminus J)\, \pi_{\Sigma\setminus\Gamma}(s[i,j]) \preceq_I t$, this implies that all positions from A indeed belong to $[s]_I \sqcap [t]_I$. First, recall that $\{1,\ldots,i-1\} \setminus J$ is downward-closed. Moreover, by Lemma 6 there does not exist a path from a node in J to a node from A. But also a path from a node in $\{i,\ldots,j\} \setminus A$ to node of A cannot exist, because by Lemma 6 every node from $\{i,\ldots,j\} \setminus A$ can be reached via a path starting in a node from J. This shows that $(\{1,\ldots,i-1\} \setminus J) \cup A$ is indeed downward-closed in D_s. □

The remaining assignments in Figure 1 update the variables RCP, Γ, and $\Gamma(k)$ (for $k \in \mathrm{RCP}$) in the correct way.

Lemma 8. *The number of iterations of the while-loop in Figure 1 is bounded by* $|\Sigma|$.

Proof. We claim that in each iteration of the while-loop, the set-variable Γ strictly grows, which proves the lemma. Let us consider an iteration of the while-loop. Since $\Gamma(j+1)$ will be the next value for Γ (see line (**)), we have to show $\Gamma \subsetneq \Gamma(j+1)$. There are two cases to distinguish. If $j < \min(\mathrm{RCP}) - 1$, then the symbol $s[j+1] \in \Gamma_{j+1,j+1}$ will belong the set $\Gamma(j+1)$. But $s[j+1]$ cannot belong to Γ, because this contradicts the choice of j in line (*) . If $j = \min(\mathrm{RCP}) - 1$, then $j+1$ is the smallest critical position from the set RCP, hence for the current Γ one has $\Gamma \subsetneq \Gamma(j+1)$. □

```
$i := 1$;
$\alpha := \varepsilon$;
$\beta := \varepsilon$;
$\mathrm{RCP} := \{|s| + 1\}$;
$\Gamma := \emptyset$;
**while** $i \leq |\mathrm{eval}(B)|$ **do**
    $j := \max\{j \leq \min(\mathrm{RCP}) - 1 \mid \mathrm{eval}(\alpha \circ \pi_{\Sigma\setminus\Gamma}(B[i,j])) \preceq_I \mathrm{eval}(C)\}$;                (*)
    $\alpha := \alpha \circ \pi_{\Sigma\setminus\Gamma}(B[i,j])$;
    $\beta := \beta \circ \pi_{\Gamma}(B[i:j]) \circ B[j+1]$;          (let us set here $B[|\mathrm{eval}(B)|+1] = \varepsilon$)
    **if** $j < \min(\mathrm{RCP}) - 1$ **then**                          (we obtain new critical points)
        **for all** $k \in \mathrm{CP}(j+1)$ **do** $\Gamma(k) := \Gamma \cup \Gamma_{j+1,k}$ **endfor**               (**)
        $\mathrm{RCP} := \mathrm{RCP} \cup \mathrm{CP}(j+1)$;
        **for all** $k \in \mathrm{RCP}$ **do** $\Gamma(k) := \bigcup\{\Gamma(\ell) \mid \ell \in \mathrm{RCP}, \ell \leq k\}$ **endfor**
        $\mathrm{RCP} := \{k \in \mathrm{RCP} \mid \neg\exists \ell \in \mathrm{RCP} : \ell < k, \Gamma(\ell) = \Gamma(k)\}$;
    **endif**
    $\Gamma := \Gamma(j+1)$;
    $i := j + 2$;
    $\mathrm{RCP} := \mathrm{RCP} \setminus \{j+1\}$;
**endwhile**
```

Fig. 2. An algorithm for computing $[\mathrm{eval}(B)]_I \sqcap [\mathrm{eval}(C)]_I$ and $[\mathrm{eval}(B)]_I \setminus [\mathrm{eval}(C)]_I$

The above algorithm for computing $[s]_I \setminus [t]_I$ leads to a polynomial time algorithm, which adds to a given extended composition system $\mathbb{B}$ with nonterminals B and C a new production $A \to \alpha$ such that $[\mathrm{eval}_{\mathbb{B}}(A)]_I = [\mathrm{eval}_{\mathbb{B}}(B)]_I \setminus [\mathrm{eval}_{\mathbb{B}}(C)]_I$, see Figure 2. [4] The idea is to consider the statements for updating u and v in Figure 1 as statements for computing right-hand sides α and β of an extended composition system.

It remains to argue that the algorithm in Figure 2 is indeed a polynomial time algorithm. By Lemma 8, the number of iterations of the while-loop is bounded by $|\Sigma|$. Hence, it suffices to show that a single iteration only needs polynomial time. The condition $\mathrm{eval}(\alpha \circ \pi_{\Sigma\setminus\Gamma}(B[i : j])) \preceq_I \mathrm{eval}(C)$ in line (*) can be checked in polynomial time by Lemma 4. Hence, the number j in line (*) can be computed in polynomial time via binary search. In line (**) we have to compute the set of positions $\mathrm{CP}(j+1) \subseteq \{1, \ldots, \mathrm{eval}(B)\}$. Recall that this set contains for every simple path p in the dependence graph (Σ, D) that starts in $\mathrm{eval}(B)[j+1]$ the position $\mathrm{pos}(j+1, p)$. The number of such paths only depends on (Σ, D) and is therefore bounded by a fixed constant. Finally, for a certain path p, we can compute the position $\mathrm{pos}(j+1, p)$ easily in polynomial time. We obtain the main result of this section:

Theorem 3. *If G is a graph group, then the compressed word problem for G belongs to the class P.*

Let us end this section with a generalization of both Theorem 2 and 3. A *graph product* is given by a triple $(\Sigma, I, (G_v)_{v\in\Sigma})$, where (Σ, I) is an independence alphabet

[4] In the algorithm we use the notation $\mathrm{eval}(\alpha)$ and $\mathrm{eval}(\beta)$ where α and β are right-hand sides of an extended composition system; the meaning is the obvious one. Moreover, concatenation of strings is denoted by $\circ$ for better readability.

and G_v is a group, which is associated with the node $v \in \Sigma$. W.l.o.g. assume that $\Sigma = \{1, \dots, n\}$. The group $\mathbb{G}(\Sigma, I, (G_v)_{v\in\Sigma})$ defined by this triple is the quotient

$$\mathbb{G}(\Sigma, I, (G_v)_{v\in\Sigma}) = (G_1 * G_2 * \cdots * G_n)/\{xy = yx \mid x \in G_u, y \in G_v, (u, v) \in I\},$$

i.e., we take the free product $(G_1 * G_2 * \cdots * G_n)$, but let elements from adjacent groups commute. Note that $\mathbb{G}(\Sigma, I, (G_v)_{v\in\Sigma})$ is the graph group $\mathbb{G}(\Sigma, I)$ in case every G_v is isomorphic to $\mathbb{Z}$. Moreover, free products and direct products appear as special cases of the graph product construction. Graph products were first studied by Green [20]. By combining the ideas from Section 4.2 with our algorithm for graph groups, one can prove:

Theorem 4. *Assume that G is a graph product of finitely generated groups $G_1, \dots, G_n$. Then* $\mathrm{CWP}(G) \leq_T^P (\mathrm{CWP}(G_1), \dots, \mathrm{CWP}(G_n))$.

4.4 Linear groups

Recall that a language L belongs to the complexity class RP (randomized polynomial time) if there exists a randomized polynomial time algorithm[5] A such that:

- if $x \notin L$ then Prob$[A$ accepts $x] = 0$
- if $x \in L$ then Prob$[A$ accepts $x] \geq 1/2$

The choice of the failure probability $1/2$ in case $x \in L$ is arbitrary: By repeating the algorithm c times (where c is some constant), we can reduce the failure probability to $(1/2)^c$ and still have a randomized polynomial time algorithm. A language L belongs to the class coRP, if the complement of L belongs to RP. This means that there exists a randomized polynomial time algorithm A such that:

- if $x \notin L$ then Prob$[A$ accepts $x] \leq 1/2$
- if $x \in L$ then Prob$[A$ accepts $x] = 1$

Theorem 5. *If G is a finitely generated linear group, then the compressed word problem for G belongs to coRP.*

Proof. Let G be linear over the field K. For the case that K has characteristic 0, it is shown in [27] that G is isomorphic to a group of matrices over the ring $\mathbb{Z}[x_1, \dots, x_n]$ (for some n). If K has prime characteristic $p > 0$, then G is isomorphic to a group of matrices over $\mathbb{F}_p[x_1, \dots, x_n]$ [40] (here $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$ is the field of cardinality p). Hence, we can reduce the compressed word problem for G to the following problem:

INPUT: A circuit C over the polynomial ring $\mathbb{Z}[x_1, \dots, x_n]$ (in case K has characteristic 0) or $\mathbb{F}_p[x_1, \dots, x_n]$ (in case K has characteristic $p > 0$).
QUESTION: Is the polynomial, to which the circuit C evaluates, the zero-polynomial?

This problem belongs to coRP by [23]. $\square$

Let us mention that graph groups are finitely generated linear [22].

[5] A randomized algorithm A may flip coins. Hence, it accepts a given input only with some probability. If there exists a polynomial $p(n)$ such that for every input of length n and every possible outcome of the coin flips, A runs in time at most $p(n)$, then A is a randomized polynomial time algorithm.

5 Applications

In this section, we present some immediate corollaries to the results from Section 3 and 4. We concentrate on automorphism groups.

It was shown in [25] (based on previous work from [39]) that the automorphism group of a graph group is finitely generated. Proposition 2 and Theorem 4 imply:

Corollary 1. *For a graph group G, the word problem for $\mathrm{Aut}(G)$ can be solved in polynomial time.*

Crisp and Wiest [9] have shown that the fundamental group of any orientable surface and of every non-orientable surfaces of genus at least 4 (see [41] for definitions) can be embedded in a graph group. Hence, by Proposition 1 and Theorem 4, the compressed word problems for these fundamental groups can be solved in polynomial time. The fundamental group of the non-orientable surface of genus 1 (the projective plane) is $\mathbb{Z}/2\mathbb{Z}$, hence its compressed word problem can be also solved in polynomial time. Finally, the fundamental group of the non-orientable surface of genus 2 (the Klein bottle) has the presentation $\langle x, y \mid x^2 = y^2 \rangle$, i.e., it is an amalgamated free product of two copies of $\mathbb{Z}$, amalgamating $2\mathbb{Z}$. Using techniques similar to those from Section 4.2 for free groups, one can show that for this group the compressed word problem can be solved in polynomial time as well. Hence, with Proposition 2 we obtain:

Corollary 2. *Let G be either the fundamental group of an orientable surface or the fundamental group of a non-orientable surface with genus different from 3. Then the word problem for $\mathrm{Aut}(G)$ can be solved in polynomial time.*

The case of the non-orientable surface of genus 3 remains open. Its fundamental group has the presentation $\langle x, y, z \mid x^2y^2 = z^2 \rangle$. Automorphism groups of fundamental groups of surfaces play an important role in algebraic topology; they are closely related to mapping class groups.

Another class of fundamental groups, which embed into graph groups are fundamental groups of finite state complexes [19]. Hence, by the above arguments, also the automorphism groups of these fundamental groups can be solved in polynomial time.

6 Open problems

Many open problems remain concerning compressed word problems. Let us mention some of them.

1. Is the compressed word problem for a hyperbolic group solvable in polynomial time? For torsion-free hyperbolic groups one might try to attack this question using the canonical representatives of Rips and Sela [36].
2. What about the compressed word problem for automatic groups? Is it possible to proof a non-trivial lower bound (e.g. NP-hardness or coNP-hardness) for the compressed word problem of some specific automatic group?

3. Is the uniform compressed word problem for graph groups solvable in polynomial time? In this problem, the independence alphabet (Σ, I), which defines the underlying graph group, is also part of the input. This question depends on whether Lemma 3 also holds for a variable terminal alphabet Γ.
4. Can Theorem 2 be generalized from free products to (suitably restricted) amalgamated free products and HNN-extensions?
5. Is it possible to relax the restriction to an automatic quotient group Q in Proposition 4?
6. The *compressed generalized word problem* for a finitely generated group G (with finite generating set Σ) asks, whether for given SLPs $\mathbb{A}, \mathbb{B}_1, \ldots, \mathbb{B}_n$ (over $\Sigma^{\pm 1}$), the word $\mathrm{eval}(\mathbb{A})$ represents a group element from the subgroup of G generated by $\mathrm{eval}(\mathbb{B}_1), \ldots, \mathrm{eval}(\mathbb{B}_n)$. Is the compressed generalized word problem for a finitely generated free group decidable in polynomial time? We are only aware of an exponential time algorithm for this problem.

References

1. A. Baudisch. Subgroups of semifree groups. *Acta Mathematica Academiae Scientiarum Hungaricae*, 38:19–28, 1981.
2. G. Baumslag, F. B. Cannonito, and C. F. Miller, III. Infinitely generated subgroups of finitely presented groups. I. *Mathematische Zeitschrift*, 153(2):117–134, 1977.
3. M. Beaudry, P. McKenzie, P. Péladeau, and D. Thérien. Finite monoids: From word to circuit evaluation. *SIAM Journal on Computing*, 26(1):138–152, 1997.
4. R. V. Book and F. Otto. *String–Rewriting Systems*. Springer, 1993.
5. W. W. Boone. The word problem. *Annals of Mathematics (2)*, 70:207–265, 1959.
6. N. Brady and J. Meier. Connectivity at infinity for right angled Artin groups. *Transactions of the American Mathematical Society*, 353:117–132, 2001.
7. F. B. Cannonito and R. W. Gatterdam. The word problem and power problem in 1-relator groups are primitive recursive. *Pacific Journal of Mathematics*, 61(2):351–359, 1975.
8. W. H. Cockcroft. The word problem in a group extension. *The Quarterly Journal of Mathematics. Oxford. Second Series*, 2:123–134, 1951.
9. J. Crisp and B. Wiest. Embeddings of graph braid and surface groups in right-angled Artin groups and braid groups. *Algebraic & Geometric Topology*, 4:439–472, 2004.
10. M. Dehn. Über die Toplogie des dreidimensionalen Raumes. *Mathematische Annalen*, 69:137–168, 1910. In German.
11. V. Diekert. *Combinatorics on Traces*. Number 454 in Lecture Notes in Computer Science. Springer, 1990.
12. V. Diekert. Word problems over traces which are solvable in linear time. *Theoretical Computer Science*, 74:3–18, 1990.
13. V. Diekert and M. Lohrey. Existential and positive theories of equations in graph products. *Theory of Computing Systems*, 37(1):133–156, 2004.
14. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.
15. C. Droms. Graph groups, coherence and three-manifolds. *Journal of Algebra*, 106(2):484–489, 1985.
16. C. Duboc. On some equations in free partially commutative monoids. *Theoretical Computer Science*, 46:159–174, 1986.
17. D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston. *Word processing in groups*. Jones and Bartlett, Boston, 1992.

18. L. Gasieniec, M. Karpinski, W. Plandowski, and W. Rytter. Efficient algorithms for Lempel-Ziv encoding (extended abstract). In R. G. Karlsson and A. Lingas, editors, *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory (SWAT 1996), Reykjavík (Iceland)*, number 1097 in Lecture Notes in Computer Science, pages 392–403. Springer, 1996.
19. R. Ghrist and V. Peterson. The geometry and topology of reconfiguration. *Advances in Applied Mathematics*, 38:302–323, 2007.
20. E. R. Green. *Graph Products of Groups*. PhD thesis, The University of Leeds, 1990.
21. C. Hagenah. *Gleichungen mit regulären Randbedingungen über freien Gruppen*. PhD thesis, University of Stuttgart, 2000.
22. S. P. Humphries. On representations of Artin groups and the Tits conjecture. *Journal of Algebra*, 169(3):847–862, 1994.
23. O. H. Ibarra and S. Moran. Probabilistic algorithms for deciding equivalence of straight-line programs. *Journal of the Association for Computing Machinery*, 30(1):217–228, 1983.
24. I. Kapovich, A. Myasnikov, P. Schupp, and V. Shpilrain. Generic-case complexity, decision problems in group theory, and random walks. *Journal of Algebra*, 264(2):665–694, 2003.
25. M. R. Laurence. A generating set for the automorphism group of a graph group. *Journal of the London Mathematical Society. Second Series*, 52(2):318–334, 1995.
26. Y. Lifshits. Solving classical string problems on compressed texts. Technical report, arXiv.org, 2006. `http://arxiv.org/abs/cs.DS/0604058`.
27. R. J. Lipton and Y. Zalcstein. Word problems solvable in logspace. *Journal of the Association for Computing Machinery*, 24(3):522–526, 1977.
28. M. Lohrey. Word problems and membership problems on compressed words. *SIAM Journal on Computing*, 35(5):1210 – 1240, 2006.
29. M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, Reading, MA, 1983.
30. R. C. Lyndon and P. E. Schupp. *Combinatorial Group Theory*. Springer, 1977.
31. M. Miyazaki, A. Shinohara, and M. Takeda. An improved pattern matching algorithm for strings in terms of straight-line programs. In A. Apostolico and J. Hein, editors, *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching (CPM 97), Aarhus (Denmark)*, Lecture Notes in Computer Science, pages 1–11. Springer, 1997.
32. P. S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *American Mathematical Society, Translations, II. Series*, 9:1–122, 1958.
33. C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.
34. W. Plandowski. Testing equivalence of morphisms on context-free languages. In J. van Leeuwen, editor, *Second Annual European Symposium on Algorithms (ESA'94), Utrecht (The Netherlands)*, number 855 in Lecture Notes in Computer Science, pages 460–470. Springer, 1994.
35. W. Plandowski and W. Rytter. Complexity of language recognition problems for compressed words. In J. Karhumäki, H. A. Maurer, G. Paun, and G. Rozenberg, editors, *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 262–272. Springer, 1999.
36. E. Rips and Z. Sela. Canonical representatives and equations in hyperbolic groups. *Inventiones Mathematicae*, 120:489–512, 1995.
37. W. Rytter. Compressed and fully compressed pattern matching in one and two dimensions. *Proceedings of the IEEE*, 88(11):1769–1778, 2000.
38. S. Schleimer. Polynomial-time word problems. Technical report, arXiv.org, 2006. `http://arxiv.org/abs/math.GR/0608563`.
39. H. Servatius. Automorphisms of graph groups. *Journal of Algebra*, 126(1):34–60, 1989.
40. H.-U. Simon. Word problems for groups and contextfree recognition. In *Proceedings of Fundamentals of Computation Theory (FCT'79), Berlin/Wendisch-Rietz (GDR)*, pages 417–422. Akademie-Verlag, 1979.

41. J. Stillwell. *Classical Topology and Combinatorial Group Theory (2nd edition)*. Springer, 1995.
42. C. Wrathall. The word problem for free partially commutative groups. *Journal of Symbolic Computation*, 6(1):99–104, 1988.