COMPRESSED DECISION PROBLEMS IN HYPERBOLIC GROUPS

DEREK HOLT, MARKUS LOHREY, AND SAUL SCHLEIMER

Abstract. We prove that the compressed word problem and the compressed simultaneous conjugacy problem are solvable in polynomial time in hyperbolic groups. In such problems, group elements are input as words defined by straight line programs defined over a finite generating set for the group. We prove also that, for any infinite hyperbolic group G, the compressed knapsack problem in G is NP-complete.

1. Introduction

Compression techniques in group theory have attracted a lot of attention in recent years [8, 9, 29, 34, 35]. The general idea is to consider classical group theoretical decision problems, such as the word problem or the conjugacy problem, but represent the input group element in a succinct way. Usually, group elements are specified by words over a finite generating set. Straight-line programs (SLPs for short) provide a well established succinct representation of words. An SLP can be seen as a context-free grammar $\mathcal{G}$ that produces a single word that we denote by $\mathsf{val}(\mathcal{G})$, see Section 4 for a precise definition. Here is a simple SLP: $A_0 \to A_1A_1$, $A_1 \to A_2A_2$, $A_2 \to A_3A_3$, $A_3 \to A_4A_4$, $A_4 \to ab$ (in the main text, we will use a slightly different notation). This SLP defines the word $(ab)^{16}$. The length of the word that is defined by an SLP can be exponential in the size of the SLP, where the latter is usually defined as the total number of symbols in all right-hand sides of productions $A \to u$. Thus, SLPs achieve exponential compression on some words. Applications of SLPs in group theory can be traced back to Babai and Szemeredi's reachability theorem for finite groups [2].

There are numerous papers in computer science that study the complexity of decision problems for words that are succinctly represented by SLPs, see [28] for a survey. Here we deal with the so called *compressed word problem* for a finitely generated group G. Let us fix a finite generating set Σ for G; all generating sets in this paper will be symmetric in the sense that $a \in \Sigma$ implies $a^{-1} \in \Sigma$. The word problem for G asks whether a given word $w \in \Sigma^*$ represents the group identity of G. It is one of the fundamental decision problems in group theory that goes back to the work of Dehn [7] from 1911. The compressed word problem for G is the same problem except that the input word w is represented by an SLP. We also say that the input is an SLP-compressed word w. Clearly, the compressed word problem for a group G is decidable if and only if the word problem for G is decidable, but with respect to computational complexity the compressed word problem for G can be more difficult than the word problem for G (and, assuming some computational hardness assumptions, there are such groups: more about that later).

It is interesting to note that the compressed word problem for a group G is exactly the *circuit evaluation problem* for G: there, the input is a circuit (a directed acyclic graph whose nodes are called gates), where the input gates are labelled with generators of G and the internal gates compute the product of their inputs.

The second author has been supported by the DFG research project LO 748/13-1.

The question is, whether a distinguished output gate evaluates to the identity of G. For finite groups, the complexity of the circuit evaluation problem (and hence, the compressed word problem) was clarified in [3]: if G is a finite solvable group, then the compressed word problem for G belongs to the parallel complexity class $\mathsf{DET} \subseteq \mathsf{NC}^2$, and if G is a finite non-solvable group, then the compressed word problem for G is P-complete. This dichotomy result naturally motivates the investigation of compressed word problems for infinite finitely generated groups. Moreover, the compressed word problem also has applications for the ordinary (uncompressed) word problem. The third author observed in [38] that the word problem for a finitely generated subgroup of the automorphism group $\mathsf{Aut}(G)$ is polynomial time reducible to the compressed word problem for G. Similar reductions exist for certain group extensions [29, Theorem 4.8 and 4.9]. This makes groups for which the compressed word problem can be solved in polynomial time interesting. Indeed the class of these groups is quite rich. A first result for infinite groups was obtained in [27], where it was shown that the compressed word problem for a finitely generated non-abelian free group is P-complete. This result was used in [38] by the third author to show that the word problem for $\mathsf{Aut}(F_n)$ can be solved in polynomial time, where F_n is the free group on n generators. This solved an open problem from [22]. Meanwhile two other important classes of groups in which the compressed word problem can be solved in polynomial time have been found:

- virtually special groups; that is, finite extensions of finitely generated subgroups of right-angled Artin groups. Right-angled Artin groups are also known as graph groups or partially commutative groups. Recent work related to three-dimensional topology has shown that the class of virtually special groups is very rich. It contains all Coxeter groups [18], one-relator groups with torsion [40], fully residually free groups [40] (for fully residually free groups, Macdonald [31] independently obtained a polynomial time solution for the compressed word problem), and fundamental groups of hyperbolic 3-manifolds [1].
- finitely generated nilpotent groups [29]. Here, the compressed word problem even belongs to the parallel complexity class DET [26].

Moreover, for finitely generated linear groups the compressed word problem belongs to the complexity class coRP [29, Theorem 4.15], which implies that there is an efficient randomized polynomial time algorithm that may err with a small probability on negative input instances. On the negative side, it is known that the compressed word problem for every restricted wreath product $G \wr \mathbb{Z}$ with G finitely generated non-abelian is coNP-hard [29, Theorem 4.21]. If G is also finite, then the word problem for $G \wr \mathbb{Z}$ can be easily solved in polynomial time, see also [39]. Assuming $\mathsf{P} \neq \mathsf{NP}$ this gives examples of groups in which the compressed word problem is harder than the word problem. Another interesting result that relates the compressed word problem to the area of algebraic complexity theory was shown in [29, Theorem 4.16]: The compressed word problem for the linear group $\mathsf{SL}_3(\mathbb{Z})$ is equivalent (w.r.t. polynomial time reductions) to polynomial identity testing (i.e., the problem whether a circuit over a polynomial ring $\mathbb{Z}[x_1, \ldots, x_n]$ evaluates to the zero polyomial).

In this paper, we prove that the compressed word problem can be solved in polynomial time in every hyperbolic group.[1] Hyperbolic groups have a Cayley-graph that satisfies a certain hyperbolicity condition, see Section 3 for a precise definition. Hyperbolic groups are of fundamental importance in geometric group theory. In a certain probabilistic sense, almost all finitely presented groups are hyperbolic [15, 36]. Also from a computational viewpoint, hyperbolic groups have nice properties: it is

[1] This result was announced in [29, Theorem 4.12] without proof.

known that the word problem and the conjugacy problem can be solved in linear time [11, 20]. They also have a nice shortlex automatic structure [10]. We show in Theorem 5.4 that, from a given SLP $\mathcal{G}$ over the generators of a hyperbolic group G, one can compute in polynomial time an SLP for the shortlex normal form of the word $\mathsf{val}(\mathcal{G})$ (this is the length lexicographically smallest word that represents the same group element as $\mathsf{val}(\mathcal{G})$). Since the shortlex normal form for a word w is the empty word if and only if $w =_G 1$ (here, and in the rest of the paper, we write $u =_G v$ if the words u and v represent the same element of the group G), we obtain the following corollary:

Corollary 1.1. *The compressed word problem for a hyperbolic group can be solved in polynomial time.*

We also use our polynomial time algorithm for the compressed word problem to solve some other compressed decision problems for hyperbolic groups. A relatively easy consequence of Corollary 1.1 is that for every hyperbolic group one can compute in polynomial time the order of the group element that is represented by a given SLP (Corollary 6.1). In Section 6.2 we consider the *compressed conjugacy problem*: the input consists of SLP-compressed words u, v and it is asked whether there exists a word x with $x^{-1}ux =_G v$. We prove that the compressed conjugacy problem for a hyperbolic group can be solved in polynomial time. For this, we show that the algorithm from [11], which solves the conjugacy problem for a hyperbolic group in linear time, can be implemented in polynomial time for SLP-compressed input words. Based on this algorithm, we then generalize our result on compressed conjugacy to the *compressed simultaneous conjugacy problem*, where the input consists of two finite lists $u_1, \dots, u_n$ and $v_1, \dots, v_n$ of SLP-compressed words over the generators of the group G, and it is asked whether there exists a word x with $x^{-1}u_i x =_G v_i$ for $1 \leq i \leq n$. This problem was shown to be solvable in polynomial time for finitely generated nilpotent groups in [32]. In the uncompressed setting, the simultaneous conjugacy problem was shown to be solvable in linear time for hyperbolic groups in [5]. Again, we show that the algorithm in [5] can be implemented in polynomial time for SLP-compressed input words, which yields:

Theorem 1.2. *Let G be a hyperbolic group. Then the compressed simultaneous conjugacy problem for G can be solved in polynomial time. Moreover, if the two input lists are conjugate, then we can compute an SLP for a conjugating element in polynomial time.*

The uncompressed simultaneous conjugacy problem has also been studied for classes of groups other than hyperbolic groups; see for example [24] and the references therein. Its SLP-compressed version is important because the word problem for finitely generated subgroups of the outer automorphism group $\mathsf{Out}(G)$ of G can be reduced to the compressed simultaneous conjugacy problem for G [19, Proposition 10]. Hence, we get the following corollary from our main results (in [6] it is shown that for a hyperbolic group G, $\mathsf{Aut}(G)$ and hence $\mathsf{Out}(G)$ are finitely generated).

Corollary 1.3. *For every hyperbolic group G, the word problems for $\mathsf{Aut}(G)$ and $\mathsf{Out}(G)$ can be solved in polynomial time.*

As a byproduct of our algorithm for the compressed simultaneous conjugacy problem we also show that for every hyperbolic group one can compute in polynomial time from a given finite set S of SLP-compressed group elements a finite generating set for the centralizer of S, where every element of this generating set is represented by an SLP. We call this computation problem the *compressed centralizer problem*.

Theorem 1.4. *Let G be a hyperbolic group. Then the compressed centralizer problem for G can be solved in polynomial time.*

Finally, we consider the compressed knapsack problem for a hyperbolic group. In the (uncompressed) knapsack problem for a finitely generated group G the input is a list of words $u_1, \ldots, u_n, v$ over the generators of G, and it is asked whether there exist natural numbers $n_1, \ldots, n_k$ such that $v =_G u_1^{n_1} \cdots u_k^{n_k}$. This problem has been studied in [12, 13, 25, 30, 33] for various classes of groups. In [33] it was shown that the knapsack problem for a hyperbolic group can be solved in polynomial time. In the compressed knapsack problem, the words $u_1, \ldots, u_n, v$ are represented by SLPs. For the special case $G = \mathbb{Z}$ this problem is a variant of the classical knapsack problem for binary encoded integers, which is known to be NP-complete. This makes it interesting to look for groups where the compressed knapsack problem belongs to NP. In [30] it was shown that compressed knapsack for every virtually special group belongs to NP. Here, we prove:

Theorem 1.5. *If G is an infinite hyperbolic group then the compressed knapsack problem for G is* NP*-complete.*

2. General notations

Zero is included in the set of natural numbers; that is, $\mathbb{N} = \{0, 1, 2, \ldots\}$. Suppose that Σ is an alphabet and Σ^* is the Kleene closure of Σ, that is, the set of all finite words over Σ. We use $\varepsilon \in \Sigma^*$ to denote the empty word. Suppose that $w = a_0 a_1 \cdots a_{n-1} \in \Sigma^*$ with $a_i \in \Sigma$. Here the *length* of w is $|w| = n$. For $0 \leq i \leq n-1$ we define $w[i] = a_i$. For $0 \leq i \leq j \leq n$ we define $w[i : j] = a_i \cdots a_{j-1}$. We use $w[: j]$ to mean $w[0 : j]$, the prefix of length j, and we also use $w[i :]$ to mean $w[i : n]$, the suffix of length $n - i$. Note that $w[i : i] = \varepsilon$ and $w = w[: i]w[i :]$ for all $0 \leq i \leq n$. We say that $u \in \Sigma^*$ is a *factor* of $w \in \Sigma^*$ if there exist $x, y \in \Sigma^*$ with $w = xuy$.

3. Hyperbolic groups

Let G be a finitely generated group with the finite symmetric generating set Σ. The Cayley-graph of G with respect to Σ is the undirected graph $\Gamma = \Gamma(G)$ with node set G and all edges (g, ga) for $g \in G$ and $a \in \Sigma$. We view Γ as a geodesic metric space, where every edge (g, ga) is identified with a unit-length interval. It is convenient to label the directed edge from g to ga with the generator a. The distance between two points p, q is denoted by $d_\Gamma(p, q)$. For $g \in G$ let $|g| := d_\Gamma(1, g)$; so $|g|$ is the length of a shortest word in Σ^* that represents g. For $r \geq 0$, let $\mathcal{B}_r(1) = \{g \in G : |g| \leq r\}$.

Given a word $w \in \Sigma^*$, one obtains a unique path $P[w]$ that starts at 1 and is labelled by the word w. This path ends in the group element represented by w. More generally, for $g \in G$ we denote by $g \cdot P[w]$ the path that starts at g and is labelled by w. We will mostly consider paths of the form $g \cdot P[w]$. One views $g \cdot P[w]$ as a continuous mapping from the real interval $[0, |w|]$ to Γ. Such a path $P : [0, n] \to \Gamma$ is *geodesic* if $d_\Gamma(P(0), P(n)) = n$. We say that a path $P : [0, n] \to \Gamma$ is path from $P(0)$ to $P(n)$. A word $w \in \Sigma^*$ is *geodesic* if the path $P[w]$ is geodesic, which means that there is no shorter word representing the same group element from G. A word $w \in \Sigma^*$ is *shortlex reduced* if it is the length-lexicographically least word that represents the same group element as w. For this, we have to fix an arbitrary linear order on Σ. Note that if $u = xy$ is shortlex reduced then x and y are shortlex reduced too. For a word $u \in \Sigma^*$ we denote by $\mathsf{shlex}(u)$ the unique shortlex reduced word that represents the same group element as u. Whenever appropriate, we identify elements of $\mathcal{B}_r(1)$ with geodesic words over Σ of length at most r.

A geodesic triangle consists of three points $p, q, r \in \Gamma$ and geodesic paths $P_{p,q}$, $P_{p,r}$, $P_{q,r}$ (the three sides of the triangle), where $P_{x,y}$ is a path from x to y. We call a geodesic triangle *δ-slim* for $\delta \geq 0$, if every point p on one of the three sides

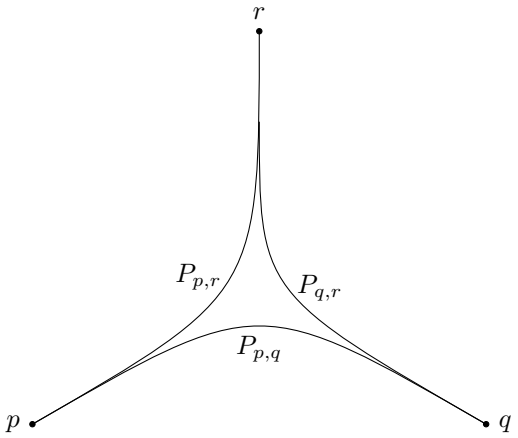

FIGURE 1. The shape of a geodesic triangle in a hyperbolic group

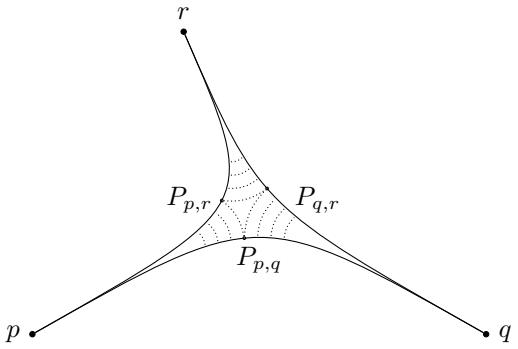

FIGURE 2. A δ-thin triangle in a hyperbolic group. Dotted lines represent geodesic paths of length at most δ.

has distance at most δ from a point p' belonging to one of the two sides that are opposite to p. The group G is called δ-hyperbolic if every geodesic triangle is δ-slim. Finally, G is hyperbolic, if it is δ-hyperbolic for some $\delta \geq 0$. Figure 1 shows the shape of a geodesic triangle in a hyperbolic group. Finitely generated free groups are for instance 0-hyperbolic. The property of being hyperbolic is independent of the chosen generating set Σ. The word problem for every hyperbolic group can be decided in real time time [20]. Moreover, one can compute $\mathsf{shlex}(w)$ from a given word w in linear time; see [11] where the result is attributed to Shapiro.

We will need an equivalent definition of hyperbolicity in terms of so called thin triangles. Again, consider three points $p, q, r \in \Gamma$ and let $P_{x,y}$ for $x, y \in \{p, q, r\}$ be a geodesic path from x to y, where $P_{y,x}$ is the path $P_{x,y}$ traversed in the reversed direction. Moreover, let $d_{x,y} = d_\Gamma(x, y)$ be the length of $P_{x,y}$. Since the three lengths $d_{p,q}$, $d_{p,r}$ and $d_{q,r}$ fulfill the triangle inequality, there exist unique real numbers $s_p, s_q, s_r \geq 0$ such that $s_x + s_y = d_{x,y}$ for all $x, y \in \{p, q, r\}$ with $x \neq y$. The geodesic triangle determined by the threes side $P_{p,q}$, $P_{p,r}$, $P_{q,r}$ is called δ-*thin* for $\delta \geq 0$, if for all x, y, z with $x \in \{p, q, r\}$ and $\{y, z\} = \{p, q, r\} \setminus \{x\}$ we have $d_\Gamma(P_{x,y}(t), P_{x,z}(t)) \leq \delta$ for all $t \in [0, s_x]$; see Figure 2 for an illustration. It is well known (see for example [21, Theorem 6.1.3]) that in a δ-hyperbolic group every geodesic triangle is δ'-thin for a constant $\delta' \geq \delta$.

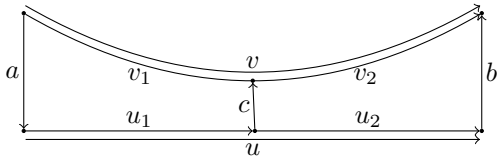

FIGURE 3. Splitting a geodesic rectangle according to Lemma 3.2.

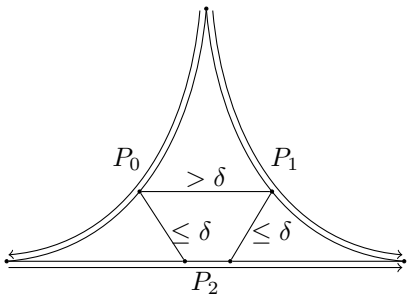

FIGURE 4. The situation from Lemma 3.3.

Let us fix a δ-hyperbolic group G with the finite symmetric generating set Σ for the rest of the section, and let Γ be the corresponding geodesic metric space. By choosing δ large enough, we can assume that all geodesic triangles in Γ are δ-thin. We will apply a couple of well-known results for hyperbolic groups.

Lemma 3.1 (c.f. [10, Theorem 3.4.5])**.** *The set* $\{\mathsf{shlex}(u) \colon u \in \Sigma^*\}$ *of all shortlex reduced words is a regular language.*

We use the following simple lemma for geodesic rectangles several times; see Figure 3.

Lemma 3.2. *Let* $a, b, u, v \in \Sigma^*$ *be geodesic words such that* $v =_G aub$ *and consider a factorization* $v = v_1 v_2$ *with* $|v_1| \geq |a| + 2\delta$ *and* $|v_2| \geq |b| + 2\delta$*. Then there exists a factorization* $u = u_1 u_2$ *and a geodesic word* c *with* $|c| \leq 2\delta$ *such that* $v_1 =_G au_1 c$ *and* $v_2 =_G c^{-1} u_2 b$*.*

Proof. Consider the endpoint p of the path $P[v_1]$ on the path $P[v]$. Since geodesic rectangles are 2δ-slim, there must exist a point q belonging to the union of the three paths $P[a]$, $a \cdot P[u]$, and $au \cdot P[b]$ such that $d_\Gamma(p, q) \leq 2\delta$. The point cannot belong to $P[a] \setminus \{a\}$ since this would imply that $|v_1| < |a| + 2\delta$ by the triangle inequality. Similarly, q cannot belong to $au \cdot P[b] \setminus \{v\}$. Hence, q belongs to $a \cdot P[u]$, which proves the lemma. □

The situation from the following Lemma is shown in Figure 4.

Lemma 3.3. *For* $i \in \{0, 1, 2\}$ *let* $P_i : [0, n_i] \to \Gamma$ *be geodesic paths such that* $P_0(0) = P_1(0)$*,* $P_0(n_0) = P_2(0)$ *and* $P_1(n_1) = P_2(n_2)$ *(so* P_0, P_1, P_2 *form a geodesic triangle). Let* $j \leq \min\{n_0, n_1\}$ *be any integer such that* $d_\Gamma(P_0(j), P_1(j)) > \delta$*. Then there exist integers* $i_0, i_1 \in [0, n_2]$ *with* $i_0 \leq i_1$*,* $d_\Gamma(P_0(j), P_2(i_0)) \leq \delta$*, and* $d_\Gamma(P_1(j), P_2(i_1)) \leq \delta$*.*

Proof. Consider the geodesic triangle with the points $p := P_0(0) = P_1(0)$, $q := P_0(n_0)$, $r := P_1(n_1)$ and the sides $P_{p,q} := P_0$, $P_{p,r} := P_1$, and $P_{q,r} := P_2$. By our choice of the constant δ, this triangle is δ-thin. For $x, y \in \{p, q, r\}$ let $d_{x,y} = d_\Gamma(x, y)$ and for $x \in \{p, q, r\}$ let $s_x \geq 0$ be the unique real numbers such that $s_x + s_y = d_{x,y}$ for all $x, y \in \{p, q, r\}$ with $x \neq y$. Then, in particular, $s_p + s_q = n_0$, $s_p + s_r = n_1$

and $s_q + s_r = n_2$. For all x, y, z with $x \in \{p, q, r\}$ and $\{y, z\} = \{p, q, r\} \setminus \{x\}$ we have $d_\Gamma(P_{x,y}(t), P_{x,z}(t)) \leq \delta$ for all $t \in [0, s_x]$. The condition on j from the lemma implies $j > s_p$. Thus, $n_0 - j < n_0 - s_p = s_q$ and $n_1 - j < n_1 - s_p = s_r$. Hence, $d_\Gamma(P_0(j), P_2(n_0 - j)) = d_\Gamma(P_{q,p}(n_0 - j), P_{q,r}(n_0 - j)) \leq \delta$ and $d_\Gamma(P_1(j), P_2(n_2 - n_1 + j)) = d_\Gamma(P_{r,p}(n_1 - j), P_{r,q}(n_1 - j)) \leq \delta$. Finally, we have $n_0 - j < s_q = n_2 - s_r < n_2 - n_1 + j$. Setting $i_0 = n_0 - j$ and $i_1 = n_2 - n_1 + j$ shows the lemma. □

4. Compressed words and the compressed word problem

4.1. **Straight-line programs.** A *straight-line program* (SLP for short) over the alphabet Σ is a triple $\mathcal{G} = (V, \rho, S)$, where V is a finite set of variables such that $V \cap \Sigma = \emptyset$, $S \in V$ is the start variable, and $\rho : V \to (V \cup \Sigma)^*$ is a mapping such that the relation $\{(B, A) \in V \times V : B \text{ occurs in } \rho(A)\}$ is acyclic. For the reader familiar with context free grammars, it might be helpful to view the SLP $\mathcal{G} = (V, \rho, S)$ as the context-free grammar (V, Σ, P, S), where P contains all productions $A \to \rho(A)$ for $A \in V$. The definition of an SLP implies that this context-free grammar derives exactly one terminal word, which will be denoted by $\mathsf{val}(\mathcal{G})$. One can define this string inductively as follows. First, for every $A \in V$ we define $\mathsf{val}_\mathcal{G}(A)$. Assume that $\rho(A) = w_0 A_1 w_1 \cdots A_k w_k$ with $k \geq 0$, $w_i \in \Sigma^*$ and $A_i \in V$. Then we define $\mathsf{val}_\mathcal{G}(A) = w_0 \mathsf{val}_\mathcal{G}(A_1) w_1 \cdots \mathsf{val}_\mathcal{G}(A_k) w_k$. Finally, we define $\mathsf{val}(\mathcal{G}) = \mathsf{val}_\mathcal{G}(S)$.

The word $\rho(A)$ is also called the *right-hand side* of A. Quite often, it is convenient to assume that all right-hand sides are of the form $a \in \Sigma$ or BC with $B, C \in V$. This corresponds to the well-known Chomsky normal form for context-free grammars. There is a simple linear time algorithm that transforms an SLP $\mathcal{G}$ with $\mathsf{val}(\mathcal{G}) \neq \varepsilon$ into an SLP $\mathcal{G}'$ in Chomsky normal form with $\mathsf{val}(\mathcal{G}) = \mathsf{val}(\mathcal{G}')$, see for example [29, Proposition 3.8].

We define the size of the SLP $\mathcal{G} = (V, \rho, S)$ as the total length of all right-hand sides: $|\mathcal{G}| = \sum_{A \in V} |\rho(A)|$. SLPs offer a succinct representation of words that contain many repeated substrings. For instance, the word $(ab)^{2^n}$ can be produced by the SLP $\mathcal{G} = (\{A_0, \ldots, A_n\}, \rho, A_n)$ with $\rho(A_0) = ab$ and $\rho(A_{i+1}) = A_i A_i$ for $0 \leq i \leq n - 1$. We need the following upper bound on the length of the word $\mathsf{val}(\mathcal{G})$:

Lemma 4.1 (c.f. [4, proof of Lemma 1])**.** *For every SLP $\mathcal{G}$ we have $|\mathsf{val}(\mathcal{G})| \leq 3^{|\mathcal{G}|/3}$.*

4.2. **Algorithms for SLP-compressed words.** We will use the fact that the following simple algorithmic tasks for SLPs can be solved in polynomial time; see also [29, Proposition 3.9].

- Given an SLP $\mathcal{G}$, compute the length $|\mathsf{val}(\mathcal{G})|$.
- Given an SLP $\mathcal{G}$ and an integer $0 \leq i < |\mathsf{val}(\mathcal{G})|$, compute the symbol $\mathsf{val}(\mathcal{G})[i]$.
- Given an SLP $\mathcal{G}$ and integers $0 \leq i \leq j \leq |\mathsf{val}(\mathcal{G})|$, compute an SLP for $\mathsf{val}(\mathcal{G})[i : j]$.

Also the following proposition is well-known:

Proposition 4.2 (c.f. [4, Lemma 2])**.** *For a given SLP $\mathcal{G}$ and integer n, we can compute an SLP $\mathcal{G}_n$ with $\mathsf{val}(\mathcal{G}_n) = \mathsf{val}(\mathcal{G})^n$ in time linear in $|\mathcal{G}| + \log |n|$.*

The following results are less trivial; this is true in particular for Theorem 4.4 and 4.5.

Proposition 4.3. *Given a complete deterministic finite state automaton M over the alphabet Σ and an SLP $\mathcal{G}$ over the alphabet Σ, we can determine in polynomial time whether $\mathsf{val}(\mathcal{G})$ is in the language $L(M)$ of M.*

Furthermore, we can determine in polynomial time whether all words in the set $\{\mathsf{val}(\mathcal{G})^k : k \in \mathbb{N}\}$ of non-negative powers of $\mathsf{val}(\mathcal{G})$ belong to $L(M)$.

Proof. The first part is proved in [29, Theorem 3.11]. For the second part, let $q_1, \ldots, q_m$ be the states of M, where q_1 is the initial state. Since M is deterministic and complete, for each word $w \in \Sigma^*$, each state q_i has a unique target state q_i^w defined by reading w through M starting at state q_i. Then $w \in L(M)$ if and only if q_1^w is an accepting state.

Let $s = \mathsf{val}(\mathcal{G})$. Now all non-negative powers of s lie in $L(M)$ if and only if $q_1^{s^k}$ is an accepting state for all $k \geq 0$. If this is the case, then there must exist integers k, l with $0 \leq k < l \leq m$ such that $q_1^{s^k} = q_1^{s^l}$. Conversely, it is easy to see that, if this condition holds and $q_1^{s^t}$ is an accepting state for $0 \leq t \leq l$, then all non-negative powers of s lie in $L(M)$.

To sum up, we have shown that $s^k \in L(M)$ for all $k \geq 0$ if and only if $s^k \in L(M)$ for all $0 \leq k \leq m$. By Proposition 4.2, we can compute for every $0 \leq k \leq m$ in polynomial time an SLP $\mathcal{G}_k$ with $\mathsf{val}(\mathcal{G}_k) = s^k$. Finally, using the first part of the proposition, we can test in polynomial time whether $\mathsf{val}(\mathcal{G}_k) \in L(M)$ for every $0 \leq k \leq m$. □

Theorem 4.4 (c.f. [37])**.** *Given two SLPs $\mathcal{G}$ and $\mathcal{H}$, one can check in polynomial time whether $\mathsf{val}(\mathcal{G}) = \mathsf{val}(\mathcal{H})$.*

The following result generalizes Theorem 4.4 to the so called *fully compressed pattern matching problem.*

Theorem 4.5 (c.f. [23])**.** *Given two SLPs $\mathcal{G}$ and $\mathcal{H}$, one can check in polynomial time whether $\mathsf{val}(\mathcal{H})$ is a factor of $\mathsf{val}(\mathcal{G})$ and, if so, find the smallest m such that $\mathsf{val}(\mathcal{G})[m : m + |\mathsf{val}(\mathcal{H})|] = \mathsf{val}(\mathcal{H})$.*

A word x is a *cyclic rotation* of the word y, if there are words x_1 and x_2 with $x = x_1x_2$ and $y = x_2x_1$. Note that x is a cyclic rotation of y if and only if $|x| = |y|$ and y if a factor of xx: If $|x| = |y|$ and $xx = x_1yx_2$, then $x = x_1x_2$ and $y = x_2x_1$. Hence, we obtain the following result from Theorem 4.5:

Proposition 4.6. *For given SLPs $\mathcal{G}$ and $\mathcal{H}$ one can check in polynomial time whether $\mathsf{val}(\mathcal{G})$ is a cyclic rotation of $\mathsf{val}(\mathcal{H})$. Moreover, if $\mathsf{val}(\mathcal{G})$ is a cyclic rotation of $\mathsf{val}(\mathcal{H})$ then we can compute in polynomial time SLPs $\mathcal{G}_1$ and $\mathcal{G}_2$ such that $\mathsf{val}(\mathcal{G}) = \mathsf{val}(\mathcal{G}_1)\mathsf{val}(\mathcal{G}_2)$ and $\mathsf{val}(\mathcal{H}) = \mathsf{val}(\mathcal{G}_2)\mathsf{val}(\mathcal{G}_1)$.*

4.3. **The compressed word problem.** The *compressed word problem* for a finitely generated group G with the finite symmetric generating set Σ is the following decision problem:

Input: an SLP $\mathcal{G}$ over the alphabet Σ.
Question: does $\mathsf{val}(\mathcal{G})$ represent the group identity of G?

It is an easy observation that the computational complexity of the compressed word problem for G does not depend on the chosen generating set Σ in the sense that if Σ' is another finite symmetric generating set for G, then the compressed word problem for G with respect to Σ is logspace reducible to the compressed word problem for G with respect to Σ' [29, Lemma 4.2]. Therefore we do not have to specify the generating set.

4.4. **Composition systems.** A useful generalization of SLPs, which are used for example in the polynomial time algorithm for the compressed word problem of a free group [27]), are the so called *composition systems*, which are called cut-SLPs (CSLPs for short) in [29]. CSLPs are defined analogously to SLPs, but in addition may contain right-hand sides of the form $B[: i]$ and $B[i :]$ for a variable B and an integer $i \geq 0$. We call $[: i]$ and $[i :]$ cut operators.

If $\rho(A) = B[:i]$ or $\rho(A) = B[i:]$ in a CSLP $\mathcal{G}$, then we define $\mathsf{val}_{\mathcal{G}}(A) = \mathsf{val}_{\mathcal{G}}(B)[:i]$ or $\mathsf{val}_{\mathcal{G}}(A) = \mathsf{val}_{\mathcal{G}}(B)[i:]$, respectively. The following result was shown by Hagenah in his PhD thesis [17] (in German), see also [29, Theorem 3.14].

Theorem 4.7 (c.f. [17])**.** *From a given CSLP $\mathcal{G}$ one can compute in polynomial time an SLP $\mathcal{G}'$ such that $\mathsf{val}(\mathcal{G}) = \mathsf{val}(\mathcal{G}')$.*

Theorems 4.4 and 4.7 imply that one can also check for two given CSLPs $\mathcal{G}_1$ and $\mathcal{G}_2$ whether $\mathsf{val}(\mathcal{G}_1) = \mathsf{val}(\mathcal{G}_2)$.

5. The compressed word problem for hyperbolic groups

Fix a δ-hyperbolic group G with the finite symmetric generating set Σ, where $\delta > 0$ is chosen in such a way that all geodesic triangles are δ-thin. We can moreover assume that δ is an integer (later, we we want to cut off from a word its prefix and suffix of length a certain multiple of δ). Let us set $\zeta := 2\delta \geq 1$ for the following.

We need an extension of CSLPs by so called twist operators. A TCSLP (T stands for "twisted") over the alphabet Σ is a tuple $\mathcal{G} = (V, \rho, S)$, where V is a finite set of variables such that $V \cap \Sigma = \emptyset$, $S \in V$ is the start variable, and ρ is a mapping with domain V such that for every $A \in V$, $\rho(A)$ is of one of the following forms:

(1) a word $w \in (V \cup \Sigma)^*$
(2) an expression $B[:i]$ or $B[i:]$ with $B \in V$ and $i \in \mathbb{N}$,
(3) an expression $B\langle a, b\rangle$ with $a, b \in \mathcal{B}_\zeta(1)$.

Moreover, we require that the relation $\{(B, A) \in V \times V : B \text{ occurs in } \rho(A)\}$ is acyclic. The reflexive and transitive closure of this relation is denoted with $\leq_{\mathcal{G}}$. We evaluate variables of type (1) or (2) as for CSLPs. For a variable A with $\rho(A) = B\langle a, b\rangle$ we define $\mathsf{val}_{\mathcal{G}}(A) := \mathsf{shlex}(a\, \mathsf{val}_{\mathcal{G}}(B)\, b)$. For convenience we will also allow more complex right-hand sides such as $(A[:i]\langle a, b\rangle)(B[j:]\langle c, d\rangle)$. We define the *size* of such a right-hand side as the total number of occurrences of symbols from $\Sigma \cup V$ in the right-hand side. The size of $\mathcal{G}$ is obtained by taking the sum over all variables.

If right-hand sides of type (2) do not occur, we speak of a TSLP. We say that the TCSLP $\mathcal{G}$ is *valid*, if for every variable A, the word $\mathsf{val}_{\mathcal{G}}(A)$ is shortlex reduced. Note that right-hand sides of type (1) may lead to words that are not shortlex reduced, since the concatenation of two shortlex reduced words is not necessarily shortlex reduced.

For a variable A, we define the height, $\mathsf{height}(A)$ for short, inductively by

$$\mathsf{height}(A) = \max\{\mathsf{height}(B) + 1 : B \in V \text{ occurs in } \rho(A)\},$$

where $\max \emptyset = 0$. Let $\mathsf{height}(\mathcal{G}) = \mathsf{height}(S)$; it is the length of a longest chain in the partial order $\leq_{\mathcal{G}}$ that ends in S.

Lemma 5.1. *Given a TSLP $\mathcal{G}$ over the alphabet Σ, we can check in polynomial time whether $\mathcal{G}$ is valid. Moreover, if $\mathcal{G}$ is valid, then we can compute in polynomial time an SLP $\mathcal{G}'$ over G such that $\mathsf{val}(\mathcal{G}) =_G \mathsf{val}(\mathcal{G}')$.*

Proof. Let $\mathcal{G} = (V, \rho, S)$. In the same way as for SLPs, we can assume that all right-hand sides from $(V \cup \Sigma)^*$ are of the form $a \in \Sigma$ or BC with $B, C \in V$. Let $\mu = \mathsf{height}(\mathcal{G})$. We will transform $\mathcal{G}$ into the desired SLP $\mathcal{G}'$. This will be done by a bottom-up process; that is we consider the variables in $\mathcal{G}$ in order of increasing height. If $\mathcal{G}$ is not valid, we will detect this during the transformation.

For a variable A, we define the twist-height, $\mathsf{theight}(A)$ for short, inductively as follows:

- if $\rho(A) = a$, then $\mathsf{theight}(A) = 0$,
- if $\rho(A) = BC$, then $\mathsf{theight}(A) = \max\{\mathsf{theight}(B), \mathsf{theight}(C)\}$, and
- if $\rho(A) = B\langle s, t\rangle$ then $\mathsf{theight}(A) = \mathsf{theight}(B) + 1$.

By removing unused variables, we can assume that S has maximal height and maximal twist height among all variables. For a nonterminal A we define $\eta_A := \mathsf{theight}(S) - \mathsf{theight}(A) + 1 > 0$.

Consider a nonterminal A. Since we are processing the variables in order of increasing height, we can assume that for all $B <_{\mathcal{G}} A$ the word $\mathsf{val}_{\mathcal{G}}(B)$ is shortlex reduced. Let $w := \mathsf{val}_{\mathcal{G}}(A)$. If $|w| \leq 16\zeta\eta_A + 2\zeta$ then we will explicitly compute the word w in the process of defining the SLP $\mathcal{G}'$. Otherwise, we will compute explicitly words ℓ_A, r_A such that $w = \ell_A w' r_A$ for some word w' of length at least 2ζ. The words ℓ_A and r_A will satisfy the length constraints

$$8\zeta\eta_A \leq |\ell_A|, |r_A| \leq 8\zeta\eta_A + 2\zeta\mathsf{height}(A).$$

Moreover, in the latter case, the SLP $\mathcal{G}'$ will contain variables $A'_{a,b}$ for all $a, b \in \mathcal{B}_\zeta(1)$ such that

$$\mathsf{val}_{\mathcal{G}'}(A'_{a,b}) = \mathsf{shlex}(aw'b).$$

The $A'_{a,b}$, together with a start variable S', which will be added at the end of the process, will be the only variables that we include in the SLP $\mathcal{G}'$. All of the words that we compute and store, such as the ℓ_A and r_A, are to enable us to carry out the necessary computations, and are not stored as part of $\mathcal{G}'$.

Case 1. $\rho(A) \in \Sigma$. Thus, $\mathsf{val}_{\mathcal{G}}(A)$ is shortlex reduced. Since $1 \leq 16\zeta\eta_A + 2\zeta$, there is nothing to do in this case.

Case 2. $\rho(A) = BC$ for variables B and C. We can assume without loss that $\eta_B = \eta_C = \eta_A$. This can be achieved by adding dummy variables with right-hand sides of the form $X\langle 1, 1\rangle$ if needed. Let $u := \mathsf{val}_{\mathcal{G}}(B)$, $v := \mathsf{val}_{\mathcal{G}}(C)$, and $w := \mathsf{val}_{\mathcal{G}}(A) = uv$. The words u and v are shortlex reduced by assumption. Let us write η for $\eta_A = \eta_B = \eta_C$.

Case 2.1. $|u| > 16\zeta\eta + 2\zeta$ and $|v| > 16\zeta\eta + 2\zeta$. Hence, we have computed words ℓ_B, r_B, ℓ_C, r_C such that the following hold:

- $8\zeta\eta \leq |\ell_B|, |r_B| \leq 8\zeta\eta + 2\zeta\mathsf{height}(B)$,
- $8\zeta\eta \leq |\ell_C|, |r_C| \leq 8\zeta\eta + 2\zeta\mathsf{height}(C)$,
- $u = \ell_B u' r_B$ and $v = \ell_C v' r_C$ for words u', v' of length at least 2ζ.

Moreover, we already have defined variables $B'_{a,b}$ and $C'_{a,b}$ for all $a, b \in \mathcal{B}_\zeta(1)$, which produce $\mathsf{shlex}(au'b)$ and $\mathsf{shlex}(av'b)$, respectively. The situation is shown in Figure 5.

We first check whether the word

$$\ell_B\, \mathsf{val}_{\mathcal{G}'}(B'_{1,1})\, r_B\, \ell_C\, \mathsf{val}_{\mathcal{G}'}(C'_{1,1})\, r_C = \ell_B u' r_B \ell_C v' r_C = uv = w$$

is shortlex reduced. This can be done in polynomial time by Proposition 4.3. If not, then we stop because $\mathcal{G}$ is not valid. Otherwise we continue as follows: we set $\ell_A := \ell_B$ and $r_A := r_C$. Note that, since $\eta_A = \eta_B = \eta_C$ and $\mathsf{height}(B) \leq \mathsf{height}(A) \geq \mathsf{height}(C)$, we have the length constraints $8\zeta\eta_A \leq |\ell_A|, |r_A| \leq 8\zeta\eta_A + 2\zeta\mathsf{height}(A)$. We also have the required bound $|u' r_B \ell_C v'| \geq 2\delta$.

It remains to define the right-hand sides for the variables $A'_{a,b}$ for all $a, b \in \mathcal{B}_\zeta(1)$. Let us fix $a, b \in \mathcal{B}_\zeta(1)$. For all $c, d \in \mathcal{B}_\zeta(1)$ we compute $z := \mathsf{shlex}(c^{-1} r_B \ell_C d^{-1})$ and check, using Proposition 4.3, whether the word

$$\mathsf{val}_{\mathcal{G}'}(B'_{a,c})\, z\, \mathsf{val}_{\mathcal{G}'}(C'_{d,b}) \quad = \quad \mathsf{shlex}(au'c)\, \mathsf{shlex}(c^{-1} r_B \ell_C d^{-1})\, \mathsf{shlex}(dv'b)$$

is shortlex reduced, in which case it is $\mathsf{shlex}(au' r_B \ell_C v' b)$. By Lemma 3.2, there must be at least one such pair c, d (there might be several pairs, in which case we choose an arbitrary one) and define

$$\rho'(A'_{a,b}) := B'_{a,c}\, z\, C'_{d,b}.$$

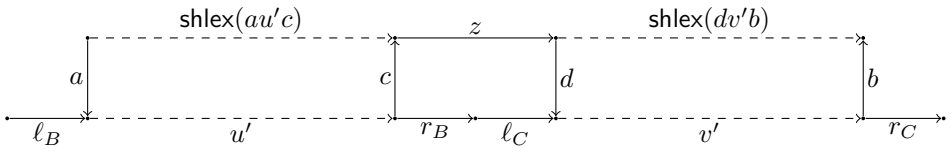

FIGURE 5. Case 2.1 from the proof of Lemma 5.1. Dotted lines represent words that are given by SLPs.

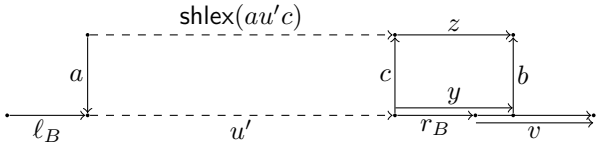

FIGURE 6. Case 2.2 from the proof of Lemma 5.1. Dotted lines represent words that are given by SLPs.

Case 2.2. $|u| > 16\zeta\eta + 2\zeta$ and $|v| \leq 16\zeta\eta + 2\zeta$. We have computed the word v explicitly. Moreover, we have computed explicit words ℓ_B and r_B such that the following hold:

- $8\zeta\eta \leq |\ell_B|, |r_B| \leq 8\zeta\eta + 2\zeta\mathsf{height}(B)$,
- $u = \ell_B u' r_B$ for a word u' of length at least 2ζ.

Moreover, we have already produced variables $B'_{a,b}$ for all $a, b \in \mathcal{B}_\zeta(1)$, which produce $\mathsf{shlex}(au'b)$.

We first check whether the word

$$\ell_B\, \mathsf{val}_{\mathcal{G}'}(B'_{1,1})\, r_B\, v = \ell_B u' r_B v = uv = w$$

is shortlex reduced. This can be done in polynomial time by Proposition 4.3. If not, then we stop because $\mathcal{G}$ is not valid.

Otherwise we continue as follows. If $|v| \leq 2\zeta$, we set $\ell_A := \ell_B$ and $r_A := r_B v$ and define $\rho'(A'_{a,b}) := B'_{a,b}$ for all $a, b \in \mathcal{B}_\zeta(1)$. In this case, we have $8\zeta\eta \leq |\ell_A| \leq 8\zeta\eta + 2\zeta\mathsf{height}(B) \leq 8\zeta\eta + 2\zeta\mathsf{height}(A)$ and $8\zeta\eta \leq |r_A| \leq 8\zeta\eta + 2\zeta(\mathsf{height}(B) + 1) \leq 8\zeta\eta + 2\zeta\mathsf{height}(A)$.

Now assume that $|v| > 2\zeta$. We set $\ell_A := \ell_B$. Since $|r_B v| \geq 8\zeta\eta$ we can define r_A as the suffix of $r_B v$ of length $8\zeta\eta$; that is, $r_B v = y r_A$ for some word y of length $|y| = |r_B| + |v| - |r_A| \geq |v| > 2\zeta$. This satisfies the required bounds on the lengths of ℓ_A and r_A. It remains to define the right-hand sides for the variables $A'_{a,b}$ for all $a, b \in \mathcal{B}_\zeta(1)$. Let us fix $a, b \in \mathcal{B}_\zeta(1)$. For all $c \in \mathcal{B}_\zeta(1)$ we compute $z := \mathsf{shlex}(c^{-1}yb)$ and check whether the word

$$\mathsf{val}_{\mathcal{G}'}(B'_{a,c})\, z = \mathsf{shlex}(au'c)\, \mathsf{shlex}(c^{-1}yb)$$

is shortlex reduced, in which case it is $\mathsf{shlex}(au'yb)$; see also Figure 6. By Lemma 3.2, there must be at least one such c, for which we define

$$\rho'(A'_{a,b}) = B'_{a,c}\, z.$$

Case 2.3. $|u| \leq 16\zeta\eta + 2\zeta$ and $|v| > 16\zeta\eta + 2\zeta$. This case is symmetric to Case 2.2.

Case 2.4. $|u| \leq 16\zeta\eta + 2\zeta$ and $|v| \leq 16\zeta\eta + 2\zeta$. In this case, we have computed u and v explicitly. We first check whether $w := uv$ is shortlex reduced. If not, then we stop. Otherwise we have to distinguish the cases $|w| \leq 16\zeta\eta + 2\zeta$ and $|w| > 16\zeta\eta + 2\zeta$. In the first case, there is nothing to do. If $|w| > 16\zeta\eta + 2\zeta$, we

factorize w as $w = \ell_A w' r_A$ with $|\ell_A| = |r_A| = 8\zeta\eta$, and thus $|w'| \geq 2\zeta$. We can compute for all $a, b \in \mathcal{B}_\zeta(1)$ the word $\mathsf{shlex}(aw'b)$ and set $\rho'(A'_{a,b}) = \mathsf{shlex}(aw'b)$.

Case 3. $\rho(A) = B\langle a, b\rangle$ for $a, b \in \mathcal{B}_\zeta(1)$. Let $u := \mathsf{val}_\mathcal{G}(B)$ and $v := \mathsf{val}_\mathcal{G}(A) = \mathsf{shlex}(aub)$. The word u is shortlex reduced by assumption, and v is shortlex reduced by definition. Let $\eta = \eta_B$. We have $\eta_A = \eta - 1 \geq 1$.

Case 3.1. $|u| \leq 16\zeta\eta + 2\zeta$. Hence we have explicitly computed the word u. We explicitly compute the word $v = \mathsf{shlex}(aub)$, and then distinguish the cases $|v| \leq 16\zeta\eta + 2\zeta$ and $|v| > 16\zeta\eta + 2\zeta$. The rest of the argument is analogous to Case 2.4.

Case 3.2. $|u| > 16\zeta\eta + 2\zeta$. We have computed words ℓ_B, r_B such that $8\zeta\eta \leq |\ell_B|, |r_B| \leq 8\zeta\eta + 2\zeta\mathsf{height}(B)$ and $u = \ell_B u' r_B$ for a word u' of length at least 2ζ. Moreover, we have already defined variables $B'_{c,d}$ for all $c, d \in \mathcal{B}_\zeta(1)$, which produce $\mathsf{shlex}(cu'd)$.

We check for all $c, d \in \mathcal{B}_\zeta(1)$ whether

$$\mathsf{shlex}(a\ell_B c^{-1})\mathsf{val}_{\mathcal{G}'}(B'_{c,d})\mathsf{shlex}(d^{-1}r_B b) = \mathsf{shlex}(a\ell_B c^{-1})\mathsf{shlex}(cu'd)\mathsf{shlex}(d^{-1}r_B b)$$

is shortlex reduced, in which case it is $\mathsf{shlex}(a\ell_B u' r_B b) = \mathsf{shlex}(aub) = v$; see Figure 7. By Lemma 3.2, there must exist such $c, d \in \mathcal{B}_\zeta(1)$. Let $s = \mathsf{shlex}(a\ell_B c^{-1})$ and $t = \mathsf{shlex}(d^{-1}r_B b)$. By the triangle inequality, these words have length at least $8\zeta\eta - 2\zeta$. Hence we can factorize these words as $s = wx$ and $t = yz$ with $|w| = |z| = 8\zeta(\eta - 1) = 8\zeta\eta_A \geq 8\zeta$. The words x and y have length at least 6ζ. We set $\ell_A := w$ and $r_A := z$. These words satisfy the required bounds on their lengths. Note that $\mathsf{val}_\mathcal{G}(A) = \mathsf{shlex}(aub) = \ell_A x\,\mathsf{shlex}(cu'd) y r_A$ and $|x\,\mathsf{shlex}(cu'd)y| \geq 12\zeta \geq 2\zeta$.

It remains to define the right-hand sides of the variables $A'_{g,h}$ for all $g, h \in \mathcal{B}_\zeta(1)$. Let us fix $g, h \in \mathcal{B}_\zeta(1)$. The lower bounds on the lengths of w, x, y, z allow us to apply Lemma 3.2 to the geodesic rectangles with sides a, ℓ_B, c, wx and d, r_B, b, yz, respectively (all of these words have been computed explicitly). We can compute in polynomial time $e, f \in \mathcal{B}_\zeta(1)$ and factorizations $\ell_B = w'x'$, $r_B = y'z'$ as shown in Figure 7. By the triangle inequality, the words x' and y' must have length at least 4ζ. Now consider the geodesic rectangle with sides $x'u'y'$, $\mathsf{shlex}(ge)$, $\mathsf{shlex}(fh)$, and $\mathsf{shlex}(gex'u'y'fh)$. Since $|x'|, |y'| \geq 4\zeta$ and $|\mathsf{shlex}(ge)|, |\mathsf{shlex}(fh)| \leq 2\zeta$, we can apply Lemma 3.2 again. There must exist $i, j \in \mathcal{B}_\zeta(1)$ such that the word

$$\mathsf{shlex}(gex'i^{-1})\,\mathsf{val}_{\mathcal{G}'}(B'_{i,j})\,\mathsf{shlex}(j^{-1}y'fh) = \mathsf{shlex}(gex'i^{-1})\,\mathsf{shlex}(iu'j)\,\mathsf{shlex}(j^{-1}y'fh)$$

is shortlex reduced, in which case the above word is $\mathsf{shlex}(gex'u'y'fh)$. As before, we can compute such $i, j \in \mathcal{B}_\zeta(1)$ in polynomial time. We finally define the right-hand side of $A'_{g,h}$ as

$$\rho'(A'_{g,h}) = \mathsf{shlex}(gex'i^{-1})\,B'_{i,j}\,\mathsf{shlex}(j^{-1}y'fh).$$

This concludes the definition of the right-hand sides for the variables $A'_{a,b}$. We complete the definition of the SLP $\mathcal{G}'$ by adding a start variable S' to $\mathcal{G}'$ and setting $\rho'(S') = \ell_S S'_{1,1} r_S$. □

The next lemma generalizes Lemma 5.1 to TCSLPs.

Lemma 5.2. *Given a TCSLP $\mathcal{G}$ over the alphabet Σ, we can check in polynomial time whether $\mathcal{G}$ is valid. Moreover, if $\mathcal{G}$ is valid, then we can compute in polynomial time an SLP $\mathcal{G}'$ over G such that $\mathsf{val}(\mathcal{G}) =_G \mathsf{val}(\mathcal{G}')$.*

Proof. The idea of the proof is taken from [17], where it is shown that a CSLP can be transformed in polynomial time into an equivalent SLP. Let $\mathcal{G} = (V, \rho, S)$ be the input TCSLP. We can assume that all right-hand sides from $(V \cup \Sigma)^*$ are of the form $a \in \Sigma$ or BC with $B, C \in V$. By Lemma 5.1 it suffices to transform $\mathcal{G}$ into an

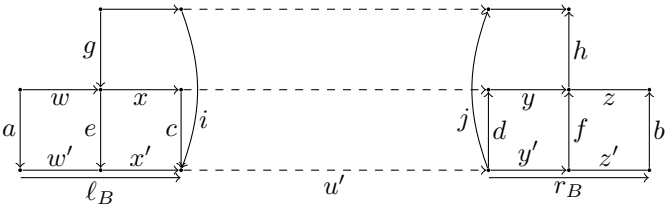

FIGURE 7. Case 3 from the proof of Lemma 5.1. Dotted lines represent words that are given by SLPs.

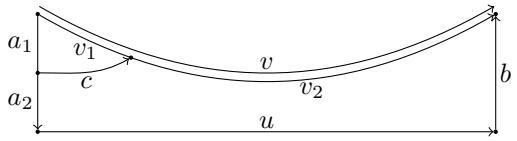

FIGURE 8. Case 3.1 in the proof of Lemma 5.2.

equivalent TSLP. Let $\mu = \mathsf{height}(\mathcal{G})$. Consider a variable A such that $\rho(A) = B[:i]$; the case that $\rho(A) = B[i:]$ can be dealt with analogously. By considering the variables in order of increasing height, we can assume that no cut operator occurs in the right-hand side of any variable $C <_{\mathcal{G}} A$. We show how to eliminate the cut operator in $\rho(A)$. Thereby we add at most μ new variables to the TCSLP. Moreover the height of the TCSLP after the cut elimination is still bounded by μ. Hence, the final TSLP has at most $\mu \cdot |V|$ variables. In addition, every right-hand side of the final TSLP will have length at most $2\zeta + 1$ (this maximum value could occur in Case 3.2 below), so its size will be polynomially bounded.

The idea of the cut elimination is to push the cut operator towards smaller variables. For this, we make a case distinction on the right-hand side of B:

Case 1. $\rho(B) = a \in \Sigma$. If $i = 1$ we redefine $\rho(A) = a$, and if $i = 0$ we redefine $\rho(A) = \varepsilon$.

Case 2. $\rho(B) = CD$ with $C, D \in V$. Since cut operators do not occur below B, we have TSLPs for $\mathsf{val}_{\mathcal{G}}(C)$ and $\mathsf{val}_{\mathcal{G}}(D)$ available. Using Lemma 5.1, we can transform these TSLPs into SLPs $\mathcal{G}_C$ and $\mathcal{G}_D$ such that $\mathsf{val}(\mathcal{G}_C) = \mathsf{val}_{\mathcal{G}}(C)$ and $\mathsf{val}(\mathcal{G}_D) = \mathsf{val}_{\mathcal{G}}(D)$. These SLPs are only used temporarily in the definition of the new right-hand side for A; later we can discard them. Let $n_C = |\mathsf{val}(\mathcal{G}_C)|$ and $n_D = |\mathsf{val}(\mathcal{G}_D)|$. These lengths can be computed in polynomial time. If $i \leq n_C$ then we redefine $\rho(A) := C[:i]$. If $i > n_C$ then we redefine $\rho(A) := CX$ for a new variable X and set $\rho(X) := D[:i - n_C]$. We then continue with the elimination of the remaining cut operator in $C[:i]$ or in $D[:i - n_C]$.

Case 3. $\rho(B) = C\langle a, b\rangle$ with $C \in V$ and $a, b \in \mathcal{B}_\zeta(1)$. Analogously to Case 2, we can assume that we have SLPs $\mathcal{G}_B$ and $\mathcal{G}_C$ with $\mathsf{val}(\mathcal{G}_B) = \mathsf{val}_{\mathcal{G}}(B)$ and $\mathsf{val}(\mathcal{G}_C) = \mathsf{val}_{\mathcal{G}}(C)$. Moreover, we can compute the lengths $n_B = |\mathsf{val}(\mathcal{G}_B)|$ and $n_C = |\mathsf{val}(\mathcal{G}_C)|$ in polynomial time. Let $u = \mathsf{val}(\mathcal{G}_C)$ and $v = \mathsf{val}(\mathcal{G}_B)$. Moreover, let $v = v_1 v_2$ with $|v_1| = i$. Thus, we have $\mathsf{val}_{\mathcal{G}}(A) = v_1$ and $v = \mathsf{shlex}(aub)$.

Case 3.1. There exists $c \in \mathcal{B}_\zeta(1)$ and a factorization $a = a_1 a_2$ such that $v_1 =_G a_1 c$; see Figure 8. Note that this implies that $i = |v_1| \leq 2\zeta$. Hence, we can check in polynomial time whether this case holds by computing the prefix of the SLP-compressed word v of length i. We redefine $\rho(A) = v_1$.

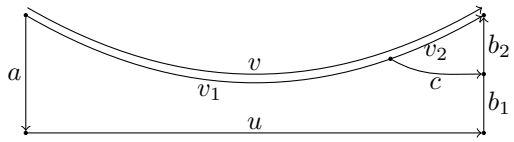

FIGURE 9. Case 3.2 in the proof of Lemma 5.2.

Case 3.2. There exists $c \in \mathcal{B}_\zeta(1)$ and a factorization $b = b_1 b_2$ such that $v_2 =_G cb_2$; see Figure 9. As in Case 3.1 we can check in polynomial time whether this condition holds. We redefine $\rho(A) = X\langle a, c^{-1}\rangle$ for a new variable X and set $\rho(X) = C\langle 1, b_1\rangle$.

Case 3.3. Neither Case 3.1 nor Case 3.2 holds. Then, there exists a factorization $u = u_1u_2$ and $c \in \mathcal{B}_\zeta(1)$ such that $v_1 =_G au_1c$ and $v_2 =_G c^{-1}u_2b$; see Figure 3. The triangle inequality implies $i - 2\zeta \le |u_1| \le i + 2\zeta$. Hence, we can find such a factorization of u in polynomial time, by computing for every $j \in \mathbb{N}$ with $|i-j| \le 2\zeta$ SLPs for the words $u[:j]$ and $u[j:]$. Then we apply Lemma 5.1 and compute for every $c \in \mathcal{B}_\zeta(1)$ SLPs for the words $w_1 := \mathsf{shlex}(au[:j]c)$ and $w_2 := \mathsf{shlex}(c^{-1}u[j:]b)$. Finally, we check using Theorem 4.4 whether $v_1 = w_1$ and $v_2 = w_2$. We will find at least one such j and c. Finally, we redefine $\rho(A) = X\langle a, c\rangle$ for a new variable X and set $\rho(X) = C[:j]$. We then continue with the elimination of the cut operator in $C[:j]$. This concludes the proof of the lemma. □

Lemma 5.3. *Given valid TCSLPs $\mathcal{G}_0$ and $\mathcal{G}_1$ such that $\mathsf{val}(\mathcal{G}_i)$ represents the group element g_i, we can check in polynomial time, whether $d_\Gamma(g_0, g_1) \le \delta$. Moreover, if this is true, we can compute $a \in \mathcal{B}_\delta(1)$ such that $g_0a =_G g_1$.*

Proof. For all $a \in \mathcal{B}_\delta(1)$ we compute, by adding one new variable to $\mathcal{G}_0$, a valid TCSLP $\mathcal{G}_{0,a}$ for $\mathsf{shlex}(\mathsf{val}(\mathcal{G}_0)a)$. Using Lemma 5.2 and Theorem 4.4 we can check in polynomial time whether $\mathsf{val}(\mathcal{G}_{0,a}) = \mathsf{val}(\mathcal{G}_1)$, which is equivalent to $g_0a =_G g_1$. □

Finally, we can prove the main technical result of this section:

Theorem 5.4. *From a given SLP $\mathcal{G}$ over the alphabet Σ we can compute in polynomial time an SLP $\mathcal{G}'$ for $\mathsf{shlex}(\mathsf{val}(\mathcal{G}))$.*

Proof. By Lemma 5.2 it suffices to compute in polynomial time a valid TCSLP $\mathcal{G}'$ for $\mathsf{shlex}(\mathsf{val}(\mathcal{G}))$. For this, we process $\mathcal{G}$ bottom-up; that is, we consider the variables in order of increasing height. Assume that $\mathcal{G} = (V, \rho, S)$ and that $\mathcal{G}$ is in Chomsky normal form. The TCSLP $\mathcal{G}'$ will contain all variables from V plus some auxiliary variables. Let us write $\mathcal{G}' = (V', \rho', S)$. For every variable $A \in V$ we will have $\mathsf{val}_{\mathcal{G}'}(A) = \mathsf{shlex}(\mathsf{val}_{\mathcal{G}}(A))$. Consider a variable $A \in V$ and assume that, for all variables $B <_{\mathcal{G}} A$, we have already defined $\rho'(B)$ in such a way that $\mathsf{val}_{\mathcal{G}'}(B) = \mathsf{shlex}(\mathsf{val}_{\mathcal{G}}(B))$.

If $\rho(A) = a \in \Sigma$ then we set $\rho'(A) := \mathsf{shlex}(a)$. Now assume that $\rho(A) = BC$. Thus we have already defined TCSLPs for the words $u := \mathsf{shlex}(\mathsf{val}_{\mathcal{G}}(B))$ and $v := \mathsf{shlex}(\mathsf{val}_{\mathcal{G}}(C))$. Moreover, by Lemma 5.2 we can transform these TCSLPs into SLPs. Using these SLPs, we can compute the lengths $m = |u|$ and $n = |v|$. If $m = 0$ or $n = 0$, then we set $\rho'(A) := C$ or $\rho'(A) := B$, respectively. So let us assume that m and n are both non-zero. Moreover, we only consider the case that $m \le n$; the other case is symmetric. From the SLP for u we can compute an SLP for u^{-1}. Consider the geodesic paths $P_0 := P[u^{-1}]$ and $P_1 := P[v]$. Using Lemma 5.3 we can check whether $d_\Gamma(P_0(m), P_1(m)) \le \delta$.

Case 1. $d_\Gamma(P_0(m), P_1(m)) \le \delta$. In this case, we can compute by Lemma 5.3 a word a of length at most δ such that $a =_G uv[:m]$. The situation is shown in Figure 10 on the left. We set $\rho'(A) := C[m:]\langle a, 1\rangle$.

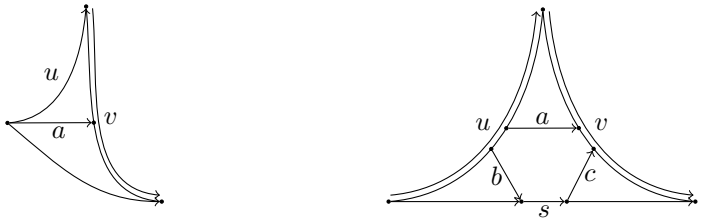

FIGURE 10. Case 1 (left) and 2 (right) from the proof of Theorem 5.4.

Case 2. $d_\Gamma(P_0(m), P_1(m)) > \delta$. Using binary search, we compute an integer $i \in [0, m-1]$ such that $d_\Gamma(P_0(i), P_1(i)) \leq \delta$ and $d_\Gamma(P_0(i+1), P_1(i+1)) > \delta$. For this we store an interval $[p, q] \subseteq [0, m]$ such that $p < q$, $d_\Gamma(P_0(p), P_1(p)) \leq \delta$ and $d_\Gamma(P_0(q), P_1(q)) > \delta$. Initially, we set $p = 0$ and $q = m$, and we stop if $q = p + 1$. In each iteration, we compute $r = \lceil (p+q)/2 \rceil$ and check, using Lemma 5.3, whether $d_\Gamma(P_0(r), P_1(r)) \leq \delta$ or $d_\Gamma(P_0(r), P_1(r)) > \delta$. In the first case we set $p := r$ and do not change q, and in the second case we set $q := r$ and do not change p. Hence, in each iteration the size of the interval $[p, q]$ is roughly halved. Therefore, the binary search stops after $\mathcal{O}(\log(m))$ iterations, which is polynomial in the input length. In addition to the position i, we can also compute $a \in \mathcal{B}_\delta(1)$ that labels a path from $P_0(i)$ to $P_1(i)$.

Let P_2 be the unique geodesic path from $P_0(m)$ to $P_1(n)$ that is labelled with a shortlex reduced word. Note that this path is labelled with $\mathsf{shlex}(uv)$. By Lemma 3.3 there exist $i_0 \leq i_1$ such that $d_\Gamma(P_0(i+1), P_2(i_0)) \leq \delta$ and $d_\Gamma(P_1(i+1), P_2(i_1)) \leq \delta$. We therefore iterate over all $b, c \in \mathcal{B}_\delta(1)$, compute the word

$$s := \mathsf{shlex}(b^{-1}u[m-i-1]av[i]c^{-1})$$

explicitly, and check whether the word

$$\mathsf{shlex}(u[:m-i-1]b)\, s\, \mathsf{shlex}(cv[i+1:]) \tag{1}$$

is shortlex reduced too, in which case it is $\mathsf{shlex}(uv)$. This can be done using Lemma 3.1 and using the fact that SLPs for u and v are available. From these SLPs we can compute TCSLPs for $\mathsf{shlex}(u[:m-i-1]b)$ and $\mathsf{shlex}(cv[i+1:])$, which can be transformed into SLPs using Lemma 5.2. It is guaranteed by Lemma 3.3 that we will find $b, c \in \mathcal{B}_\delta(1)$ such that the word in (1) is shortlex reduced. For these b, c we finally set

$$\rho'(A) := (B[:m-i-1]\langle 1, b\rangle)\ s\ (C[i+1:]\langle c, 1\rangle).$$

This concludes the proof of the theorem. □

A word $w \in \Sigma^*$ represents the group identity if and only if $\mathsf{shlex}(w) = \varepsilon$. Hence, Corollary 1.1 from the introduction follows directly from Theorem 5.4.

6. Further compressed decision problems

6.1. **Compressed order problem.** For a finitely generated group G we define the *compressed order problem* for G as the following computation problem, where Σ is an arbitrary finite symmetric generating set for G:

Input: SLP $\mathcal{G}$ over the alphabet Σ.

Output: The order (an element from $\mathbb{N} \cup \{\infty\}$) of the group element represented by $\mathsf{val}(\mathcal{G})$.

An easy consequence of Corollary 1.1 is the following result:

Corollary 6.1. *For every hyperbolic group G, the compressed order problem can be solved in polynomial time.*

Proof. Let $\mathcal{G}$ be an SLP over a fixed finite symmetric generating set for G. It is known that every hyperbolic group has a finite number of conjugacy classes of finite subgroups, and hence that there is a bound on the order of its finite subgroups [21, Theorem 6.8.4]. So there exists a constant $c = c(G)$ such that the order of every element $g \in G$ belongs to $\{1, \ldots, c, \infty\}$. Hence, in order to compute the order of (the group element represented by) $\mathsf{val}(\mathcal{G})$, it suffices to check whether $\mathsf{val}(\mathcal{G})^k =_G 1$ for all $1 \leq k \leq c+1$. By Corollary 1.1, this can be done in polynomial time. □

6.2. **Compressed conjugacy and centralizers.** Let G be a finitely generated group G with a fixed finite symmetric generating set for G. For group elements $g, h \in G$ we use the standard abbreviation $g^h = h^{-1}gh$, which is extended to lists $\mathcal{L} = (g_1, \ldots, g_k)$ with $g_i \in G$ by $\mathcal{L}^g = (g_1^h, \ldots, g_k^h)$. We extend these definitions to words over Σ in the obvious way. The *compressed conjugacy problem* for G is the following problem:

Input: SLPs $\mathcal{G}$ and $\mathcal{H}$ over the alphabet Σ.
Question: Do $\mathcal{G}$ and $\mathcal{H}$ represent conjugate elements in G? That is, does there exist $g \in G$ with $\mathsf{val}(\mathcal{G})^g =_G \mathsf{val}(\mathcal{H})$?

More generally, we can define the *compressed simultaneous conjugacy problem* for G:

Input: Finite lists $\mathcal{L}_\mathcal{G} := (\mathcal{G}_1, \ldots, \mathcal{G}_k)$ and $\mathcal{L}_\mathcal{H} := (\mathcal{H}_1, \ldots, \mathcal{H}_k)$ of SLPs over the alphabet Σ.
Question: Do $\mathcal{L}_\mathcal{G}$ and $\mathcal{L}_\mathcal{H}$ represent conjugate lists of elements in G? That is, does there exist $g \in G$ with $\mathsf{val}(\mathcal{G}_i)^g =_G \mathsf{val}(\mathcal{H}_i)$ for $1 \leq i \leq k$?

In the case when the answer to either of these questions is positive, we might also want to compute an SLP for an element that conjugates $\mathsf{val}(\mathcal{G})$ to $\mathsf{val}(\mathcal{H})$ or $\mathcal{L}_\mathcal{G}$ to $\mathcal{L}_\mathcal{H}$.

The *compressed centralizer problem* for G is the following computation problem:

Input: A finite list $(\mathcal{G}_1, \ldots, \mathcal{G}_k)$ of SLPs over G.
Output: A finite list of SLPs $(\mathcal{H}_1, \ldots, \mathcal{H}_l)$ such that $\{\mathsf{val}(\mathcal{H}_1), \ldots, \mathsf{val}(\mathcal{H}_l)\}$ is a generating set for the centralizer of the group elements represented by $\mathsf{val}(\mathcal{G}_1), \ldots, \mathsf{val}(\mathcal{G}_k)$.

The remainder of this subsection is devoted to the proofs of Theorem 1.2 and 1.4. A linear-time algorithm for solving the conjugacy problem of a hyperbolic group G is described in [11, Section 3]. This was generalized in [5] to a linear-time algorithm for the (uncompressed) simultaneous conjugacy problem and the centralizer problem. We shall show that essentially the same algorithms can be used to solve the compressed (simultaneous) conjugacy problem for G and the compressed centralizer problem in polynomial time. We first deal with the compressed conjugacy problem in Section 6.2.1. Building on the algorithm for compressed conjugacy, we will solve the compressed simultaneous conjugacy problem and the compressed centralizer problem in the case that one of the input group elements has infinite order in Section 6.2.2. Finally, in Section 6.2.3 we deal with the case that all input group elements have finite order.

6.2.1. *Compressed conjugacy of elements.* In this section we deal with the compressed conjugacy problem. The input consists of two SLPs $\mathcal{G}$ and $\mathcal{H}$ and we want to test the group elements defined by $u := \mathsf{val}(\mathcal{G})$ and $v := \mathsf{val}(\mathcal{H})$ for conjugacy. For this we basically use the conjugacy algorithm from [11] on the words u and v. We will describe this algorithm step by step in the following. Our description of each step is comprised of two paragraphs, the first describing operations relating to the words u and v, and the second explaining how we effect these operations in polynomial time using the SLPs $\mathcal{G}$ and $\mathcal{H}$. All assertions that we make in the first of these

paragraphs are justified in [11]. In most cases it is straightforward to see from the results that we have already established that the corresponding processes can be carried out in polynomial time when applied to SLPs.

Let $\delta \in \mathbb{N} \setminus \{0\}$ be a thinness constant of G on the specified symmetric generating set Σ. We define constants K and L as in [11], that is, $L := 34\delta + 2$ and $K := 17(2L+1)/7$.

A word $w \in \Sigma^*$ is said to be *shortlex straight* if all non-negative powers w^k for $k \geq 0$ are shortlex reduced words. By the second part of Proposition 4.3, we can use the finite state automaton that accepts the set of all shortlex reduced words to test in polynomial time whether compressed words are shortlex straight.

As a preprocessing step, we make a list of all pairs of shortlex reduced words of length at most K that are conjugate in G.

Step 1. We first replace u and v by $\mathsf{shlex}(u)$ and $\mathsf{shlex}(v)$.

By Theorem 5.4 we can replace in polynomial time $\mathcal{G}$ and $\mathcal{H}$ by SLPs for $\mathsf{shlex}(\mathsf{val}(\mathcal{G}))$ and $\mathsf{shlex}(\mathsf{val}(\mathcal{H}))$, respectively.

Step 2. For a word w, we define $w_C := w_R w_L$, where $w = w_L w_R$ with $|w_L| \leq |w_R| \leq |w_L| + 1$. Replace u by $\mathsf{shlex}(u_C)$ and v by $\mathsf{shlex}(v_C)$.

By Theorem 5.4 we can make the corresponding replacements to $\mathcal{G}$ and $\mathcal{H}$.

Step 3. If $|u|, |v| \leq K$ then use the precomputed list to test conjugacy of u and v. Otherwise, at least one of the words, u say, satisfies $|u| \geq K \geq 2L+1$. If $|v| < 2L+1$ then u and v are not conjugate [11, Section 3.1], and we return false. So assume from now on that $|u|, |v| \geq 2L + 1$. (We also know at this stage that all positive powers of u and v are L-local $(1, 2\delta)$-quasigeodesics.)

For the compressed conjugacy problem, if $|\mathsf{val}(\mathcal{G})|, |\mathsf{val}(\mathcal{G})| \leq K$ then we can compute $\mathsf{val}(\mathcal{G})$ and $\mathsf{val}(\mathcal{H})$ explicitly and do the same as in the uncompressed setting.

Step 4. There exists an element $g \in \mathcal{B}_{4\delta}(1)$ and m with $1 \leq m \leq |\mathcal{B}_{4\delta}(1)|^2$ such that the shortlex reduction of $g^{-1}u^m g$ is shortlex straight [11, Section 3.2]. Find such g and m, replace u by $\mathsf{shlex}(g^{-1}ug)$, and let $z := \mathsf{shlex}(u^m)$ (so z is shortlex straight).

Using Proposition 4.3, we can perform the equivalent operations in polynomial time with $\mathcal{G}$ and compute an SLP $\mathcal{G}'$ with $\mathsf{val}(\mathcal{G}') = z$.

Step 5. Test whether v^m is conjugate to z as follows. For all $h \in \mathcal{B}_{6\delta}(1)$, compute $v_h := \mathsf{shlex}(hv^m h^{-1})$, and then test whether v_h is a cyclic rotation of z. If this test fails for all h, then v^m and $z =_G u^m$ are not conjugate [11, Section 3.3]. But then, u and v are not conjugate, so we return false. Otherwise, we find h and v_h with this property. Let z_1 be a prefix of z such that $z =_G z_1 h v^m h^{-1} z_1^{-1}$, and replace v by $\mathsf{shlex}(z_1 h v h^{-1} z_1^{-1})$. Now we have $v^m =_G u^m =_G z$. From this we get that every $g \in G$ with $g^{-1}ug =_G v$ belongs to the centralizer $C_G(z)$ of z in G. In particular, u and v are conjugate in G if and only if they are conjugate in $C_G(z)$.

Using Proposition 4.6 we can do the corresponding calculations with $\mathcal{H}$ and $\mathcal{G}'$. Checking whether v_h is a cyclic rotation of z can be accomplished in polynomial time by the first statement of Proposition 4.6 and the second statement allows us to compute in polynomial time an SLP for z_1.

Step 6. Find the shortest prefix y of z such that $z = y^l$ for some $l \geq 1$. We do that by finding the second occurrence of the substring z in the word z^2.

Perform the corresponding calculation with $\mathcal{G}'$ and, if $l > 1$, find an SLP $\mathcal{G}''$ with $\mathsf{val}(\mathcal{G}'') = y$. Theorem 4.5 implies that we can do this in polynomial time.

Step 7. For each $h \in \mathcal{B}_{2\delta}(1)$, compute $\mathsf{shlex}(hzh^{-1})$ and test whether it is a cyclic rotation of z. If so, find a prefix z_2 of z with $hzh^{-1} =_G z_2^{-1} z z_2$, and compute and store $\mathsf{shlex}(z_2 h)$ (which lies in $C_G(z)$) in a list C_z. Then $|C_z| \leq J := |\mathcal{B}_{2\delta}(1)|$.

Proposition 4.6 allows us to do the corresponding calculations with $\mathcal{G}'$ in polynomial time. We obtain a list of SLPs that evaluate to the words in the list C_z.

Step 8. For each n with $0 \leq n \leq (J-1)!$ and each $z' \in C_z$, let $g = y^n z'$ and test whether $u =_G gvg^{-1}$. If so then return true (and a conjugating element). If not, then u and v are not conjugate [11, Section 3.4], so return false.

We can perform corresponding operations in polynomial time with the SLPs. This concludes the description of a polynomial time algorithm for the compressed conjugacy problem.

6.2.2. *Compressed simultaneous conjugacy and centralizers: the infinite order case.* We now move on to the compressed simultaneous conjugacy problem and the compressed centralizer problem. So the input consists of two lists $\mathcal{L}_\mathcal{G} := (\mathcal{G}_1, \ldots, \mathcal{G}_k)$ and $\mathcal{L}_\mathcal{H} := (\mathcal{H}_1, \ldots, \mathcal{H}_k)$ of SLPs over the alphabet Σ. For the compressed centralizer problem we assume that $\mathcal{G}_i = \mathcal{H}_i$. Let $u_i := \mathsf{val}(\mathcal{G}_i)$ and $v_i := \mathsf{val}(\mathcal{H}_i)$.

The algorithm below involves a number of replacements of the elements in the first list by conjugates, culminating in a check for a conjugating element among the elements of an explicit finite set. In each of the replacements, SLPs are known for the conjugating elements involved, and so by keeping track of these, we can find (in the case when the lists are conjugate) a conjugating element for the original input lists. We shall give no further details of this process in the proofs below.

As in [5] we start by solving the problems in the case when at least one of the group elements defined in the list $\mathcal{L}_\mathcal{G}$ has infinite order. By Corollary 6.1 we can check in polynomial time whether some u_i has infinite order. Let us assume that this holds. By reordering the lists if necessary, we may assume that u_1 has infinite order. We can assume that the same applies to v_1 since otherwise the lists are not conjugate. We now apply the element conjugacy algorithm to u_1 and v_1 to find an SLP for an element $g \in G$ with $u_1^g =_G v_1$ (if there is no such g then the lists are not conjugate). This process involves replacing u_1 and v_1 by conjugates in some of the steps, and when we do that we must of course make corresponding replacements to the other elements u_i and v_i in the lists. Finally, by replacing each u_i by its conjugate under g, we may assume that $u_1 = v_1$.

So we need to decide whether there exists $g \in C_G(u_1)$ with $u_i^g =_G v_i$ for $2 \leq i \leq k$. In the following we refer to the eight steps from Section 6.2.1. Recall that after Step 5, we have $u_1^m =_G v_1^m =_G z$ for some shortlex straight element z and $m > 0$. These equations are preserved if we conjugate with an element from $C_G(z)$. Hence, for the $u_1 = v_1$ computed in the previous paragraph we also have $u_1^m = z$. In Step 6, we write $z = y^l$. It is shown in [11, Section 3.4] that all elements $g \in C_G(z)$ have the form $g =_G y^n z'$, for some $n \in \mathbb{Z}$ and $z' \in C_z$, where C_z is the list (of SLPs) of elements in $C_G(z)$ that we computed in Step 7. So the same applies to any $g \in C_G(u_1) \subseteq C_G(z)$.

Now, by trying each $z' \in C_z$ in turn, and replacing each v_i by $z' v_i (z')^{-1}$ in each case, the problem reduces to the following: does there exist $n \in \mathbb{Z}$ such that $u_i^{y^n} = v_i$ for $1 \leq i \leq k$?

To solve this problem, we apply [5, Proposition 3.23] to each pair u_i, v_i in turn. For each i, there are three possibilities.

(i) there exist $0 \leq r_i < t_i \leq |\mathcal{B}_{2\delta}(1)|$ such that $u_i^{y^j} =_G v_i$ if and only if $j \equiv r_i \bmod t_i$;
(ii) there is a unique $r_i \in \mathbb{Z}$ with $u_i^{y^{r_i}} =_G v_i$, where $|r_i|$ is bounded by a linear function of $|u_i|$ and $|v_i|$;
(iii) there is no $r_i \in \mathbb{Z}$ with $u_i^{y^{r_i}} =_G v_i$.

[5, Proposition 3.23] provides an algorithm for determining which case applies, and for finding r_i, t_i in cases (i) and (ii). This involves calculating a bounded number of powers u_i^n, v_i^n and y^n for integers n such that $|n|$ is bounded by a linear function of $|u_i|$ and $|v_i|$, and we can perform that calculation in polynomial time with SLPs by Proposition 4.2.

After performing this calculation for each i with $1 \leq i \leq k$, the conjugacy problem for the lists reduces to solving some modular linear equations involving the integers r_i and t_i, as described in [5, Section 3.4]. Since the r_i and t_i in case (i) are bounded by a constant and, for r_i in case (ii), $\log |r_i|$ is bounded by a linear function of the size of the SLPs representing u_i and v_i, these equations can be solved in polynomial time using standard arithmetical operations on the binary representations of r_i and t_i.

For the compressed centralizer problem, we are in the same situation but with $v_i = u_i$ for all i. We perform the same calculations as above, but we do it for each $z' \in C_z$ and, if there are solutions, then we can find them by solving modular equations. The centralizer of the list is either finite, or is virtually cyclic, in which case it contains a subgroup of $\langle y \rangle$ of finite index, which we will find when we do the calculation for $z' = 1$. See [5, Section 3.5] for more details.

6.2.3. *Compressed simultaneous conjugacy and centralizers: the finite order case.* This case is handled for lists of elements given by standard input words in [5, Section 4]. In fact no new complications arise when applying the same methods to lists defined by SLPs. Indeed some of the steps become easier, because we are only interested in achieving polynomial time rather than linear time.

We follow the steps of the algorithm described in [5, Section 4.5]. Note that the conjugacy and centralizer problems are handled together, where the two lists are taken to be equal for the centralizer calculation. At this stage we have already verified that all u_i and v_i have finite order, and these words are all shortlex reduced. We start by eliminating any duplicates in the lists $u_1, \ldots, u_k$ and by $v_1, \ldots, v_k$. So we can now assume that the u_i represent distinct group elements, and similarly for the v_i.

Let $n := \min\{|\mathcal{B}_{2\delta}(1)|^4 + 1, k\}$. We consider the initial bounded length sublists $\mathcal{L}_1 := (u_1, \ldots, u_n)$ and $\mathcal{L}_2 := (v_1, \ldots, v_n)$ of the lists of group elements. We apply the function ShortenWords from [5, Section 4.2] to the lists $\mathcal{L}_1$ and $\mathcal{L}_2$. This function involves finding subwords, replacing words by cyclic rotations, and applying shlex to words, which can be executed in polynomial time when working with SLPs. Since there is an absolute bound $|\mathcal{B}_{2\delta}(1)|^4 + 1$ on the lengths of $\mathcal{L}_1$ and $\mathcal{L}_2$, the complete application of ShortenWords to each list takes place in polynomial time.

ShortenWords either finds a product $u_r u_{r+1} \cdots u_s$ of elements of $\mathcal{L}_1$ that has infinite order, and thereby reduces the problem to the previous case; or it replaces $\mathcal{L}_1$ and $\mathcal{L}_2$ by conjugates and then calculates lists $\mathcal{L}_1' := (u_1', \ldots, u_n')$ and $\mathcal{L}_2' = (v_1', \ldots, v_n')$ with $|u_i'|$ and $|v_i'|$ bounded by a constant, and such that $\mathcal{L}_1^g = \mathcal{L}_2$ if and only if $(\mathcal{L}_1')^g = \mathcal{L}_2'$.

In the second case, we can test in time $O(1)$ using standard operations in the group whether there exists $g \in G$ with $(u_1', \ldots, u_n')^g = (v_1', \ldots, v_n')$ (or we could look this up on a precomputed list). If so, we replace $(u_1, \ldots, u_k)$ by $(u_1, \ldots, u_k)^g$ and thereby assume that $u_i = v_i$ for $1 \leq i \leq n$. For the centralizer problem, methods are described in [14, Proposition 2.3] of finding a generating set of the centralizer of any quasiconvex subgroup of any biautomatic group, and finitely generated subgroups of hyperbolic groups satisfy these conditions. Since they need only be applied to words of bounded length their complexity does not matter - indeed, we could precompute all such centralizers.

This completes the proof in the case $n = k$. In the case $k > n$, it is proved in [5, Corollary 4.6] that the centralizer C of the subgroup $\langle u_1, \ldots, u_n \rangle$ is finite, and that the elements of C have lengths bounded by a constant. So we can compute the elements of C explicitly (in time $O(1)$). Then we simply need to check whether any $g \in C$ satisfies $(u_{n+1}, \ldots, u_k)^g = (v_{n+1}, \ldots, v_k)$.

6.3. **Compressed knapsack.** Let G be a finitely generated group with the finite symmetric generating set Σ. A *knapsack expression* over G is a formal expression of the form $E = v_0 u_1^* v_1 u_2^* v_2 \cdots u_k^* v_k$ with $k \geq 0$ and $u_i, v_i \in \Sigma^*$. A solution for E is a tuple $(n_1, n_2, \ldots, n_k) \in \mathbb{N}^k$ of natural numbers such that $v_0 u_1^{n_1} v_1 u_2^{n_2} v_2 \cdots u_k^{n_k} v_k =_G 1$. The *length* of E is defined as $|E| = |v_0| + \sum_{i=1}^k |u_i| + |v_i|$. The knapsack problem for G is the following decision problem:

Input: A knapsack expression E over G.
Question: Does E has a solution?

Note that $v_0 u_1^{n_1} v_1 u_2^{n_2} v_2 \cdots u_k^{n_k} v_k =_G 1$ if and only if

$$(v_0 u_1 v_0^{-1})^{n_1} (v_0 v_1 u_2 v_1^{-1} v_0^{-1})^{n_2} \cdots (v_0 \cdots v_{k-1} u_k v_{k-1}^{-1} \cdots v_0^{-1})^{n_k} (v_0 \cdots v_k) = 1.$$

Hence, it suffices to consider knapsack expressions of the form $u_1^* u_2^* \cdots u_k^* v$.

In [33] it was shown that the knapsack problem for a hyperbolic group can be solved in polynomial time. A crucial step in the proof for this fact is the following result, which is of independent interest:

Theorem 6.2 (c.f. [33])**.** *For every hyperbolic group G there exists a polynomial $p(x)$ such that the following holds: if a knapsack expression $E = v_0 u_1^{x_1} v_1 u_2^{x_2} v_2 \cdots u_k^{x_k} v_k$ over G has a solution then it has a solution $(n_1, \ldots, n_k) \in \mathbb{N}^k$ such that $n_i \leq p(|E|)$ for all $1 \leq i \leq k$.*

Let us now consider the *compressed knapsack problem* for G. It is defined in the same way as the knapsack problem, except that the words $u_i, v_i \in \Sigma^*$ are given by SLPs. It is well known that the compressed knapsack problem for $\mathbb{Z}$ is NP-complete [16, Proposition 4.1.1]. In fact, this problem corresponds to a variant of the classical knapsack problem for binary encoded integers (for an integer z, it is easy to construct in polynomial time from the binary encoding of z an SLP over the symmetric generating set $\{a, a^{-1}\}$ of $\mathbb{Z}$ which evaluates to a^z or to $(a^{-1})^{-z}$). Hence, for every non-torsion group, the compressed knapsack problem is NP-hard. This makes it interesting to look for groups where the compressed knapsack problem is NP-complete.

From Corollary 1.1 and Theorem 6.2 we can easily prove Theorem 1.5, which states that compressed knapsack for an infinite hyperbolic group G is NP-complete.

Proof of Theorem 1.5. Consider a knapsack expression $E = v_0 u_1^* v_1 u_2^* v_2 \cdots u_k^* v_k$ over G, where the u_i and v_i are given by SLPs $\mathcal{G}_i$ and $\mathcal{H}_i$, respectively. Let $N := |\mathcal{H}_0| + \sum_{i=1}^k (|\mathcal{G}_i| + |\mathcal{H}_i|)$ be the input length. By Theorem 6.2, there exists a polynomial $p(x)$ such that E has a solution if and only if it has a solution $(n_1, \ldots, n_k) \in \mathbb{N}^k$ of $E = 1$ such that $n_i \leq p(|E|)$ for all $1 \leq i \leq k$. Since $|u_i| \leq 3^{|\mathcal{G}_i|/3}$ and $|v_i| \leq 3^{|\mathcal{H}_i|/3}$) by Lemma 4.1, we obtain a bound of the form $2^{\mathcal{O}(N)}$ on the n_i. Hence, we can guess a tuple $(n_1, \ldots, n_k) \in \mathbb{N}^k$ with all n_i bounded by $2^{\mathcal{O}(N)}$ and then check whether it is a solution of E. The latter can be done in polynomial time by constructing from the SLPs $\mathcal{G}_i$ and $\mathcal{H}_i$ an SLP $\mathcal{G}$ for $v_0 u_1^{n_1} v_1 u_2^{n_2} v_2 \cdots u_k^{n_k} v_k$ using Proposition 4.2. Finally, we check in polynomial time whether $\mathsf{val}(\mathcal{G}) =_G 1$ using Corollary 1.1.

That compressed knapsack is NP-hard for G follows from the well known fact that every infinite hyperbolic group is non-torsion together with the above mentioned result for $\mathbb{Z}$ [16, Proposition 4.1.1]. □

References

[1] Ian Agol. The virtual Haken conjecture. *Documenta Mathematica*, 18:1045–1087, 2013. With an appendix by Ian Agol, Daniel Groves, and Jason Manning.

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

[3] Martin Beaudry, Pierre McKenzie, Pierre Péladeau, and Denis Thérien. Finite monoids: From word to circuit evaluation. *SIAM Journal on Computing*, 26(1):138–152, 1997.

[4] Moses Charikar, Eric Lehman, April Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005.

[5] J. Buckley D. and Derek F. Holt. The conjugacy problem in hyperbolic groups for finite lists of group elements. *International Journal of Algebra and Computation*, 23(5):1127–1150, 2013.

[6] François Dahmani and Vincent Guirardel. The isomorphism problem for all hyperbolic groups. *Geometric and Functional Analysis*, 21(2):223–300, 2011.

[7] Max Dehn. Über unendliche diskontinuierliche Gruppen. *Mathematische Annalen*, 71:116–144, 1911.

[8] Volker Diekert, Jürn Laun, and Alexander Ushakov. Efficient algorithms for highly compressed data: the word problem in Higman's group is in P. *International Journal of Algebra and Computation*, 22(8), 2012.

[9] Will Dison, Eduard Einstein, and Timothy R. Riley. Ackermannian integer compression and the word problem for hydra groups. In *Proceedings of MFCS 2016*, volume 58 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[10] David B. A. Epstein, James W. Cannon, Derek F. Holt, Silvio V. F. Levy, Michael S. Paterson, and William P. Thurston. *Word Processing in Groups*. Jones and Bartlett, 1992.

[11] David B. A. Epstein and Derek F. Holt. The linearity of the conjugacy problem in word-hyperbolic groups. *International Journal of Algebra and Computation*, 16(2):287–306, 2006.

[12] E. Frenkel, A. Nikolaev, and A. Ushakov. Knapsack problems in products of groups. *Journal of Symbolic Computation*, 74:96–108, 2016.

[13] Moses Ganardi, Daniel König, Markus Lohrey, and Georg Zetzsche. Knapsack problems for wreath products. In *Proceedings of STACS 2018*, volume 96 of *LIPIcs*, pages 32:1–32:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[14] S.M. Gersten and H.B. Short. Rational subgroups of biautomatic groups. *Ann. of Math.*, 134(1):125–158, 1991.

[15] Mikhail Gromov. Hyperbolic groups. In S. M. Gersten, editor, *Essays in Group Theory*, number 8 in MSRI Publ., pages 75–263. Springer, 1987.

[16] Christoph Haase. *On the complexity of model checking counter automata*. PhD thesis, University of Oxford, St Catherine's College, 2011.

[17] Christian Hagenah. *Gleichungen mit regulären Randbedingungen über freien Gruppen*. PhD thesis, University of Stuttgart, 2000.

[18] Frédéric Haglund and Daniel T. Wise. Coxeter groups are virtually special. *Advances in Mathematics*, 224(5):1890–1903, 2010.

[19] Nico Haubold, Markus Lohrey, and Christian Mathissen. Compressed decision problems for graph products of groups and applications to (outer) automorphism groups. *International Journal of Algebra and Computation*, 22(8), 2013.

[20] Derek Holt. Word-hyperbolic groups have real-time word problem. *International Journal of Algebra and Computation*, 10:221–228, 2000.

[21] Derek F. Holt, Sarah Rees, and Claas E. Röver. *Groups, Languages and Automata*, volume 88 of *London Mathematical Society Student Texts*. Cambridge University Press, 2017.

[22] Ilya Kapovich, Alexei Myasnikov, Paul Schupp, and Vladimir Shpilrain. Generic-case complexity, decision problems in group theory, and random walks. *Journal of Algebra*, 264(2):665–694, 2003.

[23] Marek Karpinski, Wojciech Rytter, and Ayumi Shinohara. Pattern-matching for strings with short descriptions. In *Proceedings of CPM 1995*, volume 937 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 1995.

[24] Martin Kassabov and Francesco Matucci. The simultaneous conjugacy problem in groups of piecewise linear functions. *Groups, Geometry, and Dynamics*, 6(2):279–315, 2012.

[25] D. König, M. Lohrey, and G. Zetzsche. Knapsack and subset sum problems in nilpotent, polycyclic, and co-context-free groups. In *Algebra and Computer Science*, volume 677 of *Contemporary Mathematics*, pages 138–153. American Mathematical Society, 2016.

[26] Daniel König and Markus Lohrey. Evaluation of circuits over nilpotent and polycyclic groups. *Algorithmica*, 80(5):1459–1492, 2018.

[27] Markus Lohrey. Word problems and membership problems on compressed words. *SIAM Journal on Computing*, 35(5):1210 – 1240, 2006.
[28] Markus Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.
[29] Markus Lohrey. *The Compressed Word Problem for Groups*. SpringerBriefs in Mathematics. Springer, 2014.
[30] Markus Lohrey and Georg Zetzsche. Knapsack in graph groups. *Theory of Computing Systems*, 62(1):192–246, 2018.
[31] Jeremy Macdonald. Compressed words and automorphisms in fully residually free groups. *International Journal of Algebra and Computation*, 20(3):343–355, 2010.
[32] Jeremy MacDonald, Alexei G. Myasnikov, and Denis Ovchinnikov. Low-complexity computations for nilpotent subgroup problems. *CoRR*, abs/1706.01092, 2017.
[33] Alexei G. Myasnikov, Andrey Nikolaev, and Alexander Ushakov. Knapsack problems in groups. *Mathematics of Computation*, 84:987–1016, 2015.
[34] Alexei G. Myasnikov, Alexander Ushakov, and Dong Wook Won. The word problem in the Baumslag group with a non-elementary Dehn function is polynomial time decidable. *Journal of Algebra*, 345(1):324–342, 2011.
[35] Alexei G. Myasnikov, Alexander Ushakov, and Dong Wook Won. Power circuits, exponential algebra, and time complexity. *International Journal of Algebra and Computation*, 22(6), 2012.
[36] A. Yu. Ol'shanskii. Almost every group is hyperbolic. *International Journal of Algebra and Computation*, 2(1):1–17, 1992.
[37] Wojciech Plandowski. Testing equivalence of morphisms on context-free languages. In *Proceedings of ESA 1994*, volume 855 of *Lecture Notes in Computer Science*, pages 460–470. Springer, 1994.
[38] Saul Schleimer. Polynomial-time word problems. *Commentarii Mathematici Helvetici*, 83(4):741–765, 2008.
[39] Stephan Waack. The parallel complexity of some constructions in combinatorial group theory. *Journal of Information Processing and Cybernetics EIK*, 26:265–281, 1990.
[40] Daniel T. Wise. Research announcement: the structure of groups with a quasiconvex hierarchy. *Electronic Research Announcements in Mathematical Sciences*, 16:44–55, 2009.

University of Warwick, UK
E-mail address: D.F.Holt@warwick.ac.uk

Universität Siegen, Germany
E-mail address: lohrey@eti.uni-siegen.de

University of Warwick, UK
E-mail address: s.schleimer@warwick.ac.uk