Groups with ALOGTIME-Hard Word Problems and PSPACE-Complete Circuit Value Problems

Laurent Bartholdi
ENS Lyon, Unité de Mathématiques Pures et Appliquées, France
Universität Göttingen, Mathematisches Institut, Germany
laurent.bartholdi@gmail.com

Michael Figelius
Universität Siegen, Germany
figelius@eti.uni-siegen.de

Markus Lohrey
Universität Siegen, Germany
lohrey@eti.uni-siegen.de

Armin Weiß
Universität Stuttgart, Institut für Formale Methoden der Informatik (FMI), Germany
armin.weiss@fmi.uni-stuttgart.de

Abstract

We give lower bounds on the complexity of the word problem of certain non-solvable groups: for a large class of non-solvable infinite groups, including in particular free groups, Grigorchuk's group and Thompson's groups, we prove that their word problem is ALOGTIME-hard. For some of these groups (including Grigorchuk's group and Thompson's groups) we prove that the circuit value problem (which is equivalent to the circuit evaluation problem) is PSPACE-complete.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Circuit complexity; Mathematics of computing → Combinatorics

Keywords and phrases NC^1-hardness, word problem, G-programs, straight-line programs, non-solvable groups, self-similar groups, Thompson's groups, Grigorchuk's group

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.29

Related Version For the full version see `https://arxiv.org/abs/1909.13781`.

Funding *Michael Figelius*: Funded by DFG project LO 748/12-1.
Markus Lohrey: Funded by DFG project LO 748/12-1.
Armin Weiß: Funded by DFG project DI 435/7-1.

Acknowledgements The authors are grateful to Schloss Dagstuhl and the organizers of Seminar 19131 for the invitation, where this work began.

1 Introduction

Groups and word problems. The *word problem* of a finitely generated group G is the most fundamental algorithmic problem in group theory [28, 42]. Recall that a group G with identity element 1 is finitely generated (f.g. for short) if there is a finite set $\Sigma \subseteq G$ such that every element of G can be written as a product of elements of Σ; this product can be formally written as a word from Σ^*. For technical reasons we assume that $1 \in \Sigma$ (which is needed for padding reasons) and that for every $a \in \Sigma$ also the inverse a^{-1} belongs to Σ; such a generating set Σ is called *standard*. We have a natural involution on Σ^* defined by $(a_1 \cdots a_n)^{-1} = a_n^{-1} \cdots a_1^{-1}$ for $a_i \in \Sigma$ (which is the same as forming inverses in the group). For words $u, v \in \Sigma^*$ we write $u =_G v$ if u and v are equal in G; sometimes we just say $u = v$

in G. The word problem for G, $\mathrm{WP}(G)$ for short, is the question whether $u =_G 1$ holds for a given word $u \in \Sigma^*$. In this formulation the word problem depends on the generating set Σ, but it is well-known that the complexity/decidability status of the word problem does not depend on Σ.

The original motivation for the word problem came from topology and group theory [14] within Hilbert's "Entscheidungsproblem". Nevertheless, it also played a role in early computer science when Novikov and Boone constructed finitely presented groups with an undecidable word problem [10, 40]. Still, in many classes of groups it is (efficiently) decidable, a prominent example being the class of linear groups: Lipton and Zalcstein [36] (for linear groups over a field of characteristic zero) and Simon [44] (for linear groups over a field of prime characteristic) showed that their word problem is in LOGSPACE. A striking connection between the word problem for groups and complexity theory was established by Barrington [3]: for every finite non-solvable group G, the word problem of G is complete for ALOGTIME, which is the same as DLOGTIME-uniform NC^1. Moreover, the reduction is as simple as it could be: every output bit depends on only one input bit. Thus, one can say that ALOGTIME is completely characterized via group theory. Moreover, this idea has been extended to characterize ACC^0 by solvable monoids [4]. On the other hand, the word problem of a finite p-group is in $\mathsf{ACC}^0[p]$, so Smolensky's lower bound [45] implies that it is strictly easier than the word problem of a finite non-solvable group.

Barrington's construction is based on the observation that an and-gate can be simulated by a commutator. This explains the connection to non-solvability. In this light, it seems natural that the word problem of finite p-groups is not ALOGTIME-hard: they are all nilpotent, so iterated commutators eventually become trivial. For infinite groups, a construction similar to Barrington's was used by Robinson [41] to show that the word problem of a non-abelian free group is ALOGTIME-hard. Since by [36] the word problem of a free group is in LOGSPACE, the complexity is narrowed down quite precisely (although no completeness is known).

Strongly efficiently non-solvable groups and ALOGTIME. The first contribution of this paper is to identify the essence of Barrington's and Robinson's constructions. For this we introduce a strengthened condition of non-solvability. Here $[h, g] = h^{-1}g^{-1}hg$ denotes the *commutator* of h and g.

▶ **Definition 1.** *We call a group G with the finite standard generating set Σ* uniformly strongly efficiently non-solvable (uniformly SENS) *if there is a constant $\mu \in \mathbb{N}$ and words $g_{d,v} \in \Sigma^*$ for all $d \in \mathbb{N}$, $v \in \{0,1\}^{\leq d}$ such that*

(a) *$|g_{d,v}| = 2^{\mu d}$ for all $v \in \{0,1\}^d$,*

(b) *$g_{d,v} = \left[g_{d,v0},\, g_{d,v1}\right]$ for all $v \in \{0,1\}^{<d}$ (here we take the commutator of words),*

(c) *$g_{d,\varepsilon} \neq 1$ in G, and*

(d) *given $v \in \{0,1\}^d$, a positive integer i encoded in binary with μd bits, and $a \in \Sigma$ one can decide in* DLINTIME *(see Section 3 for a definition of* DLINTIME*) whether the i-th letter of $g_{d,v}$ is a.*

If G is required to only satisfy (a)–(c), then G is called SENS.

In a SENS group G, non-solvability is witnessed by efficiently computable balanced nested commutators of arbitrary depth that are non-trivial in G. The class of (uniformly) SENS groups enjoys several nice properties: in particular, the definition is independent of the choice of the generating set, it is inherited from subquotients (quotients of subgroups) and it is preserved under forming the quotient by the center of a group (see Lemmas 12–14). By following Barrington's arguments we show:

▶ **Theorem 2.** *Let G be uniformly SENS. Then WP(G) is hard for* ALOGTIME *under* DLOGTIME*-reductions (and also* DLOGTIME*-uniform projection reductions or* AC^0*-reductions).*

That means that for every non-solvable group G, the word problem for G is ALOGTIME-hard, unless the word length of the G-elements witnessing the non-solvability grows very fast (in the full version [5] we give an example of a non-solvable group where the latter happens) or these elements cannot be computed efficiently. For Theorem 2 the padding letter 1 in the generating set for G is important; otherwise, we only get a TC^0-many-one reduction.

Examples of SENS groups. Finite non-solvable groups and non-abelian free groups are easily seen to be uniformly SENS; this was essentially shown by Barrington (for finite non-solvable groups) and Robinson (for non-abelian free groups) in their ALOGTIME-hardness proofs for the word problem. We go beyond these classes and present in the full version [5] a general criterion that implies the uniform SENS-condition. As an application, we can show ALOGTIME-hardness of the word problems for several famous groups:

▶ **Corollary 3.** *The word problems for the following groups are hard for* ALOGTIME*:*

- *Thompson's groups,*
- *weakly branched self-similar groups with a finitely generated branching subgroup.*

Thompson's groups $F < T < V$ (introduced in 1965) belong due to their unusual properties to the most intensively studied infinite groups. From a computational perspective it is interesting to note that all three Thompson's groups are co-context-free (i.e., the set of all non-trivial words over any set of generators is a context-free language) [33]. This implies that the word problems for Thompson's groups are in LOGCFL. To the best of our knowledge no better upper complexity bound is known. Weakly branched groups form an important subclass of the self-similar groups [39], containing several celebrated groups like the Grigorchuk group (the first example of a group with intermediate word growth) and the Gupta-Sidki groups. We also show that the word problem for so-called contracting self-similar groups is in LOGSPACE. This result is well-known, but to the best of our knowledge no proof has appeared in the literature. The Grigorchuk group as well as the Gupta-Sidki groups are known to be contracting and have finitely generated branching subgroups, so Corollary 3 leaves only a small range for the complexity of their word problems.

The proof of the general result implying Corollary 3 is deferred to the full version [5] (we give a sketch in Section 4). Nevertheless, in this work we present direct proofs that Thompson's groups F and the Grigorchuk group are SENS, which yields Corollary 3 for these special cases.

In [31, Theorem 7], König and the third author showed that the word problem of f.g. solvable linear group is in TC^0. They asked the question whether there is a dichotomy in the sense that the word problem of a linear group either is in TC^0 or ALOGTIME-hard. As another application of the SENS condition, we can answer this question affirmatively using the famous Tits' alternative [46]:

▶ **Corollary 4.** *For every f.g. linear group the word problem either is in* DLOGTIME*-uniform* TC^0 *or the word problem is* ALOGTIME*-hard.*

Circuit value problems for groups. In the second part of the paper we study the *circuit value problem* for a finitely generated group G, CVP(G) for short. Fix a standard generating set Σ for G. The input for CVP(G) is a circuit in which input gates are labelled with generators

from Σ and every non-input gate computes the group-product of its two predecessor gates (we have to distinguish the left and right predecessor gate since G is in general not commutative). There is a distinguished output gate and the question is whether it evaluates to 1. In the group-theoretic literature, the circuit value problem for G is usually called the *compressed word problem* [38]. The reason for this is that one can evaluate a circuit $\mathcal{G}$ of the above form also in the free monoid Σ^*; it then corresponds to a context-free grammar that generates a single word w. The circuit $\mathcal{G}$ can be seen as a compressed representation of this word w. The circuit value problem is the succinct version of the word problem, where the input word is represented by a circuit.

Circuit value problems for finite groups have been studied in [8]. It was shown that $\mathrm{CVP}(G)$ is P-complete for finite non-solvable groups (by a Barrington style argument) and in NC^2 for finite solvable groups. The circuit value problem for linear groups is tightly related to PIT (polynomial identity testing, i.e., the question whether a circuit over a polynomial ring evaluates to the zero-polynomial; see e.g. [43]): For every f.g. linear group the circuit value problem reduces in polynomial time to PIT for $\mathbb{Z}[x]$ or $\mathbb{F}[x]$ and hence belongs to coRP, the complement of randomized polynomial time [38, Theorem 4.15]. Moreover, the circuit value problem for the group $\mathrm{SL}_3(\mathbb{Z})$ is equivalent to PIT for $\mathbb{Z}[x]$ with respect to polynomial time reductions [38, Theorem 4.16].

From a group theoretic viewpoint, the circuit value problem is interesting not only because it is a natural succinct version of the word problem, but also because several classical word problems efficiently reduce to circuit value problems. For instance, the word problem for a finitely generated subgroup of $\mathrm{Aut}(G)$ reduces in polynomial time to the circuit value problem for G [38, Theorem 4.6]. Similar statements hold for certain group extensions [38, Theorems 4.8 and 4.9]. This motivates the search for groups in which the circuit value problem can be solved in polynomial time. This applies to finitely generated nilpotent groups [31] (for which the circuit value problem can be even solved in NC^2), hyperbolic groups [27] and virtually special groups [38]. The latter are defined as finite extensions of subgroups of right-angled Artin groups and form a very rich class of groups containing for instance Coxeter groups [21], fully residually free groups [51] and fundamental groups of hyperbolic 3-manifolds [1].

Recently, Wächter and the fourth author constructed an automaton group (a finitely generated group of tree automorphism, where the action of generators is defined by a Mealy automaton) with a PSPACE-complete word problem and $\mathsf{EXPSPACE}$-complete circuit value problem [49]. The group arises by encoding a Turing machine into a group; in particular, one cannot call this group natural. In this paper, we exhibit several natural groups (that were intensively studied in other parts of mathematics) with a PSPACE-complete circuit value problem (and a word problem in $\mathsf{LOGSPACE}$). The two main ingredients for our construction is the uniform SENS-property defined above and the wreath product construction.

Circuit value problems for wreath products. The wreath product $G \wr H$ is a fundamental construction in group theory and semigroup theory; important applications are the Krasner–Kaloujnine embedding theorem in group theory and the Krohn-Rhodes decomposition theorem in semigroup theory. The formal definition of wreath products can be found in Section 2. We are interested in the circuit value problem for wreath products of the form $G \wr \mathbb{Z}$ (for G f.g.). Such groups are also called lamplighter groups (the classical lamplighter group is $(\mathbb{Z}/2) \wr \mathbb{Z}$). The following result was shown in [38] (for G non-abelian) and [32] (for G abelian via a reduction to polynomial identity testing).

▶ **Theorem 5** (c.f. [32, 38]). *If G is a f.g. group, then*
- *$CVP(G \wr \mathbb{Z})$ is coNP-hard if G is non-abelian and*
- *$CVP(G \wr \mathbb{Z})$ belongs to coRP (co-randomized polynomial time) if G is abelian.*

Our main result for the circuit value problem in wreath products pinpoints the exact complexity of $\mathrm{CVP}(G \wr \mathbb{Z})$ for the case that G has a trivial center (recall that the *center* $Z(G)$ of the group G is the normal subgroup consisting of all elements $g \in G$ that commute with every element from G). Theorem 6 below uses the concept of leaf languages [11, 23, 25, 26, 29], which is formally defined in Appendix A. For a language $K \subseteq \Gamma^*$ over a finite alphabet Γ one considers nondeterministic polynomial time machines M that after termination print a symbol from Γ on every computation path. Moreover, one fixes a linear ordering on the transition tuples of M. As a consequence the computation tree $T(x)$ for a machine input x becomes a finite ordered tree. The corresponding leaf string $\mathrm{leaf}(M, x)$ is obtained by listing symbols from Γ that are printed in the leafs of $T(x)$ from left to right. The class $\mathsf{LEAF}(K)$ consists of all languages L for which there exists a nondeterministic polynomial time machines as described above such that $x \in L$ if and only if $\mathrm{leaf}(M, x) \in K$. As a prototypical example note that $\mathsf{NP} = \mathsf{LEAF}(\{0,1\}^*1\{0,1\}^*)$. Here, we are interested in leaf language classes where K is the word problem for a f.g. group. For this we identify the word problem with the language $\mathrm{WP}(G, \Sigma) = \{w \in \Sigma^* \mid w =_G 1\}$. One can easily show that the generating set Σ has no influence on the class $\mathsf{LEAF}(\mathrm{WP}(G, \Sigma))$ (see Lemma 30 in the appendix). Hence, we simply write $\mathsf{LEAF}(\mathrm{WP}(G))$. We are actually interested in the class $\forall\mathsf{LEAF}(\mathrm{WP}(G))$, where for a complexity class C we denote by $\forall\mathsf{C}$ the class of all languages L such that there exists a polynomial $p(n)$ and a language $K \in \mathsf{C}$ with $L = \{u \mid \forall v \in \{0,1\}^{p(|u|)} : u\#v \in K\}$ (hence, for instance $\forall\mathsf{P} = \mathsf{coNP}$ and $\forall\mathsf{PSPACE} = \mathsf{PSPACE}$). Our main result for the circuit value problem in wreath products is:

▶ **Theorem 6.** *Let G be a f.g. non-trivial group with center $Z = Z(G)$.*

- *$CVP(G \wr \mathbb{Z})$ belongs to $\forall\mathsf{LEAF}(WP(G))$.*
- *$CVP(G \wr \mathbb{Z})$ is hard for the class $\forall\mathsf{LEAF}(WP(G/Z))$.*

In particular, if $Z = 1$, then $CVP(G \wr \mathbb{Z})$ is complete for $\forall\mathsf{LEAF}(WP(G))$.

PSPACE-complete circuit value problems. From Theorem 6 we derive PSPACE-completeness of the circuit value problem for some interesting groups:

▶ **Corollary 7.** *The circuit value problem for the following groups is* PSPACE*-complete:*

(i) *wreath products $G \wr \mathbb{Z}$ where G is finite non-solvable or free of rank at least two,*

(ii) *Thompson's groups,*

(iii) *the Grigorchuk group, and*

(iv) *all Gupta-Sidki groups.*

In order to derive this corollary from Theorem 6 we also need a kind of padded version of Theorem 2 saying that PSPACE is contained in $\mathsf{LEAF}(\mathrm{WP}(G/Z(G)))$ (this yields PSPACE-hardness of $\mathrm{CVP}(G \wr \mathbb{Z})$ for every SENS group G). For Thompson's groups, the Grigorchuk group, and the Gupta-Sidki groups we also use a certain self-embedding property: for all these groups G a wreath product $G \wr A$ embeds into G for some $A \neq 1$. Thompson's group F has this property for $A = \mathbb{Z}$ [19]. For the Grigorchuk group and the Gupta-Sidki groups (and, more generally, weakly branched groups whose branching subgroup is finitely generated and has elements of finite order) we show that one can take $A = \mathbb{Z}/p$ for some $p \geq 2$.

Some of the proofs can be found only in the full version [5] of this paper. The proof of Theorem 6 can be found in the appendix.

2 Background from group theory

For group elements $g, h \in G$ or words $g, h \in \Sigma^*$ we write g^h for the *conjugate* $h^{-1}gh$ and $[h, g]$ for the *commutator* $h^{-1}g^{-1}hg$. We call g a *d-fold nested commutator*, if $d = 0$ or $g = [h_1, h_2]$ for $(d-1)$-fold nested commutators h_1, h_2. A *subquotient* of a group G is a quotient of a subgroup of G.

Wreath products. We consider groups G that act on a set X on the left or right. For $g \in G$ and $x \in X$ we write $x^g \in X$ (resp., ${}^g x$) for the result of a right (resp., left) action. An important case arises when $G = \mathrm{Sym}(X)$ is the symmetric group on a set X, which acts on X on the right.

A fundamental group construction that we shall use is the *wreath product*: given groups G and H acting on the right on sets X and Y respectively, their *wreath product* $G \wr H$ is a group acting on $X \times Y$. We start with the restricted direct product $G^{(Y)}$ (the base group) of all mappings $f : Y \to G$ having finite support $\mathrm{supp}(f) = \{y \mid f(y) \neq 1\}$ with the operation of pointwise multiplication. The group H has a natural left action on $G^{(Y)}$: for $f \in G^{(Y)}$ and $h \in H$, we define ${}^h f \in G^{(Y)}$ by $({}^h f)(y) = f(y^h)$. The corresponding semidirect product $G^{(Y)} \rtimes H$ is the *wreath product* $G \wr H$. In other words:

- Elements of $G \wr H$ are pairs $(f, h) \in G^{(Y)} \times H$; we simply write fh for this pair.
- The multiplication in $G \wr H$ is defined as follows: Let $f_1 h_1, f_2 h_2 \in G \wr H$. Then $f_1 h_1 f_2 h_2 = f_1 {}^{h_1} f_2 h_1 h_2$, where the product $f_1 {}^{h_1} f_2 \colon y \mapsto f_1(y) f_2(y^{h_1})$ is the pointwise product.

The wreath product $G \wr H$ acts on $X \times Y$ by $(x, y)^{fh} = (x^{f(y)}, y^h)$. The wreath product defined above is also called the *(restricted) permutational wreath product*. There is also the variant where $G = X$, $H = Y$ and both groups act on themselves by right-multiplication, which is called the *(restricted) regular wreath product* (or *standard wreath product*). A subtle point is that the permutational wreath product is an associative operation whereas the regular wreath product is in general not. The term "restricted" refers to the fact that the base group is $G^{(Y)}$, i.e., only finitely supported mappings are taken into account. If $G^{(Y)}$ is replaced by G^Y (i.e., the set of all mappings from Y to G with pointwise multiplication), then one speaks of an unrestricted wreath product. For Y finite this makes of course no difference. We will only deal with restricted wreath products. The action of G on X is usually not important for us, but it is nice to have an associative operation. The right group H will be either a symmetric group $\mathrm{Sym}(Y)$ acting on the right on Y or a (finite or infinite) cyclic group acting on itself by $g^h = g + h$. Thus, if H is cyclic, the permutational wreath product and the regular wreath product (both denoted by $G \wr H$) coincide. Nevertheless, be aware that $G \wr (H \wr H) = (G \wr H) \wr H$ holds only for the permutational wreath product even if H is cyclic. Note that if G is generated by Σ and H is generated by Γ then $G \wr H$ is generated by $\Sigma \cup \Gamma$.

Richard Thompson's groups. In 1965 Richard Thompson introduced three finitely presented groups $F < T < V$ acting on the unit-interval, the unit-circle and the Cantor set, respectively. Of these three groups, F received most attention (the reader should not confuse F with a free group). This is mainly due to the still open conjecture that F is not amenable, which would imply that F is another counterexample to a famous conjecture of von Neumann (a counterexample was found by Ol'shanskii). A standard reference for Thompson's groups is [12]. The group F consists of all homeomorphisms of the unit interval that are piecewise affine, with slopes a power of 2 and dyadic breakpoints. Famously, F has a finite presentation with two generators: $F = \langle x_0, x_1 \mid [x_0 x_1^{-1}, x_0^{-1} x_1 x_0], [x_0 x_1^{-1}, x_0^{-2} x_1 x_0^2] \rangle$. Very convenient is

also the following infinite presentation: $F = \langle x_0, x_1, x_2, \dots \mid x_k^{x_i} = x_{k+1} (i < k) \rangle$. The group F is orderable (so in particular torsion-free), its derived subgroup $[F, F]$ is simple and the center of F is trivial. Important for us is the following fact:

▶ **Lemma 8** ([19, Lemma 20]). *The group F contains a subgroup isomorphic to $F \wr \mathbb{Z}$.*

Hence, the limit group $H_\infty = \bigcup_{i \geq 0} H_i$, where $H_0 = \mathbb{Z}$ and $H_{i+1} = H_i \wr \mathbb{Z}$, is contained in F.

Weakly branched groups. We continue our list of examples with an important class of groups acting on rooted trees. For more details, [6, 39] serve as good references. Let X be a finite set. The free monoid X^* serves as the vertex set of a regular rooted tree with an edge between v and vx for all $v \in X^*$ and all $x \in X$. The group W of automorphisms of this tree naturally acts on the set X of level-1 vertices, and permutes the subtrees hanging from them. Exploiting the bijection $X^+ = X^* \times X$, we thus have an isomorphism

$$\varphi \colon W \to W \wr \mathrm{Sym}(X) = W^X \rtimes \mathrm{Sym}(X), \tag{1}$$

mapping $g \in W$ to elements $f \in W^X$ and $\pi \in \mathrm{Sym}(X)$ as follows: π is the restriction of g to $X \subseteq X^*$, and f is uniquely defined by $(xv)^g = x^\pi v^{f(x)}$. We always write $g@x$ for $f(x)$ and call it the *state (or coordinate) of g at x*. If $X = \{0, \dots, k\}$ we write $g = \langle\!\langle g@0, \dots, g@k \rangle\!\rangle \pi$.

▶ **Definition 9.** *A subgroup $G \leq W$ is* self-similar *if $\varphi(G) \leq G \wr \mathrm{Sym}(X)$. In other words: the actions on subtrees xX^* are given by elements of G itself. A self-similar group G is* weakly branched *if there exists a non-trivial subgroup $K \leq G$ with $\varphi(K) \geq K^X$. In other words: for every $k \in K$ and every $x \in X$ the element acting as k on the subtree xX^* and trivially elsewhere belongs to K. A subgroup K as above is called a* branching subgroup.

Note that we are weakening the usual definition of "weakly branched": indeed it is usually additionally required that G act transitively on X^n for all $n \in \mathbb{N}$. This extra property is not necessary for our purposes, so we elect to simply ignore it. In fact, all the results concerning branched groups that we shall use will be proven directly from Definition 9.

Note also that the join $\langle K_1 \cup K_2 \rangle$ of two branching subgroups K_1 and K_2 is again a branching subgroup. Hence, there exists a maximal branching subgroup. It immediately follows from the definition that, if G is weakly branched, then for every $v \in X^*$ there is in G a copy of its branching subgroup K whose action is concentrated on the subtree vX^*.

There exist important examples of f.g. self-similar weakly branched groups, notably the *Grigorchuk group* G, see [17]. It may be described as a self-similar group in the following manner: it is a group generated by $\{a, b, c, d\}$, and acts on the rooted tree X^* for $X = \{0, 1\}$. The action, and therefore the whole group, are defined by the restriction of φ to G's generators: $\varphi(a) = (0, 1)$, $\varphi(b) = \langle\!\langle a, c \rangle\!\rangle$, $\varphi(c) = \langle\!\langle a, d \rangle\!\rangle$, and $\varphi(d) = \langle\!\langle 1, b \rangle\!\rangle$, where we use the notation $(0, 1)$ for the non-trivial element of $\mathrm{Sym}(X)$ (that permutes 0 and 1) and $\langle\!\langle w_0, w_1 \rangle\!\rangle$ for a tuple in $G^{\{0,1\}} \cong G \times G$. We record some classical facts:

▶ **Lemma 10.** *The Grigorchuk group G is infinite, torsion, weakly branched, and all its finite subquotients are 2-groups (so in particular nilpotent). It has a f.g. branching subgroup.*

Other examples of f.g. self-similar weakly branched groups with a f.g. branching subgroup include the Gupta-Sidki groups [20], the Hanoi tower groups [18], and all iterated monodromy groups of degree-2 complex polynomials [7] except z^2 and $z^2 - 2$.

Contracting self-similar groups. Recall the notation $g@x$ for the coordinates of $\varphi(g)$. We iteratively define $g@v = g@x_1 \cdots @x_n$ for any word $v = x_1 \cdots x_n \in X^*$. A self-similar group G is called *contracting* if there is a finite subset $N \subseteq G$ such that, for all $g \in G$, we have $g@v \in N$ whenever v is long enough (depending on g), see also [39, Definition 2.11.1].

If G is a f.g. contracting group with word norm $\|\cdot\|$ (i.e., for $g \in G$, $\|g\|$ is the length of a shortest word over a fixed generating set of G that represents g), then a more quantitative property holds: there are constants $0 < \lambda < 1$, $h \geq 1$ and $k \geq 0$ such that for all $g \in G$ we have $\|g@v\| \leq \lambda\|g\| + k$ for all $v \in X^h$; see e.g. [28, Proposition 9.3.11]. Then, for $c = -h/\log\lambda$ and a possibly larger k we have $g@v \in N$ whenever $|v| \geq c\log\|g\| + k$. It is well-known and easy to check that the Grigorchuk group, the Gupta-Sidki groups and the Hanoi tower group for three pegs are contracting. The following result has been quoted numerous times, but has never appeared in print. We give a proof in the full version [5]. A proof for the Grigorchuk group may be found in [16]:

▶ **Proposition 11.** *Let G be a f.g. contracting self-similar group. Then $WP(G)$ can be solved in* LOGSPACE *(deterministic logarithmic space).*

For the proof of Proposition 11 one shows that if an element g of a contracting self-similar group G acts as the identity on all words $v \in X^*$ of length $\mathcal{O}(\log\|g\|)$, then $g = 1$.

3 Complexity theory

Since we also deal with sublinear time complexity classes, we use Turing machines with *random access*. Such a machine has an additional index tape and some special query states. Whenever the Turing machine enters a query state, the following transition depends on the input symbol at the position which is currently written on the index tape in binary notation. We use the abbreviations DTM/NTM/ATM for deterministic/non-deterministic/alternating Turing machine. We define the following complexity classes:

- DLINTIME: the class of languages that can be accepted by a DTM in linear time.
- DLOGTIME: the class of languages that can be accepted by a DTM in logarithmic time.
- ALOGTIME: the class of languages that can be accepted by an ATM in logarithmic time. It is well-known that ALOGTIME = DLOGTIME-uniform NC^1, see [47] for details.
- APTIME: the class of languages that can be accepted by an ATM in polynomial time. We have APTIME = PSPACE.

A function $f\colon \Gamma^* \to \Sigma^*$ is DLOGTIME-computable if there is some polynomial p with $|f(x)| \leq p(|x|)$ for all $x \in \Gamma^*$ and the set $L_f = \{(x, a, i) \mid x \in \Gamma^*$ and the i-th letter of $f(x)$ is $a\}$ belongs to DLOGTIME. Here i is a binary encoded integer. A DLOGTIME-reduction is a DLOGTIME-computable many-one reduction.

The class AC^0 (resp. TC^0) is defined as the class of languages (respectively functions) accepted (respectively computed) by circuits of constant depth and polynomial size with not-gates and unbounded fan-in and- and or-gates (resp. unbounded fan-in threshold-gates).

4 Efficiently non-solvable groups and ALOGTIME

Recall the definition of a SENS (strongly efficiently non-solvable) group from Definition 1; (a)–(d) refer to this definition in the following. We start with some observations:

- A SENS group is clearly non-solvable, so the terminology makes sense. In the full version [5] we give an example of a f.g. group that is non-solvable, has decidable word problem, but is not SENS. The construction is inspired from [50].

- If one can find suitable $g_{d,v}$ of length at most $2^{\mu d}$, then these words can always be padded to length $2^{\mu d}$ thanks to the padding letter 1.
- It suffices to specify $g_{d,v}$ for $v \in \{0,1\}^d$; the other $g_{d,v}$ are then defined by Condition (b).
- We have $|g_{d,v}| = 2^{\mu d + 2(d-|v|)}$ for all $v \in \{0,1\}^{\leq d}$. Thus, all $g_{d,v}$ have length $2^{\mathcal{O}(d)}$.
- Equivalently to Condition (d), we can require that given $v \in \{0,1\}^d$ and a binary encoded number i with μd bits, one can compute the i-th letter of $g_{d,v}$ in DLINTIME.

Proof sketch for Theorem 2. The proof of Theorem 2 essentially follows Barrington's proof that the word problem of finite non-solvable groups is ALOGTIME-hard [3]. The entire proof can be found in the full version [5], where we also state a non-uniform version of Theorem 2. Since the proof for the uniform case is not difficult but tedious, in this sketch we primarily focus on the non-uniform version, which only gives us hardness via (non-uniform) AC^0-reductions instead of DLOGTIME-reductions.

Like in Barrington's proof, we start with an ALOGTIME-machine (resp. NC^1-circuit) and construct a family of so-called G-programs. Since we are dealing with finitely generated, but infinite groups, we have to adapt the definition of G-programs slightly.

Fix a finite standard generating set Σ of G. A G-program P of length m and input length n is a sequence of *instructions* $\langle i_j, b_j, c_j\rangle$ for $0 \leq j \leq m-1$ where $i_j \in [1..n]$ and $b_j, c_j \in \Sigma$. On input of a word $x = x_1 \cdots x_n \in \{0,1\}^*$, an instruction $\langle i_j, b_j, c_j\rangle$ evaluates to b_j if $x_{i_j} = 1$ and to c_j otherwise. The evaluation of a G-program is the product (in the specified order) of the evaluations of its instructions, and is denoted with $P[x] \in \Sigma^*$.

Let M be an ALOGTIME-machine in input normal form [47, Lemma 2.41], i. e., every computation path queries at most one input bit and M halts immediately after the query. For every input size n, the computation tree of M translates immediately into a Boolean circuit of depth $d \in \mathcal{O}(\log n)$. Moreover, M can be normalized such that this circuit is a fully balanced binary tree meaning that the gates of the circuit are indexed by the set $\{0,1\}^{\leq d}$, where $\{0,1\}^{<d}$ are the inner gates (where ε is the output gate, which counts here as an inner gate) and $\{0,1\}^d$ are the leaves (input gates). We can assume that all inner gates are nand-gates (where the Boolean function $\mathsf{nand} : \{0,1\}^2 \to \{0,1\}$ is defined by $\mathsf{nand}(0,0) = \mathsf{nand}(0,1) = \mathsf{nand}(1,0) = 1$ and $\mathsf{nand}(1,1) = 0$) and each leaf is labelled by a possibly negated input variable or constant via an input mapping $q_n \colon \{0,1\}^d \to [1..n] \times \{0,1\} \times \{0,1\}$. The meaning of this mapping is as follows: if the input to the circuit is the bit string $x_1 x_2 \cdots x_n$ and $q_n(v) = \langle i, a, b\rangle$, then the input gate $v \in \{0,1\}^d$ evaluates to a (resp., b) if $x_i = 1$ (resp., $x_i = 0$).

The family of circuits obtained this way can be shown to be DLOGTIME-uniform in an even stronger sense than the usual definition (see e. g. [47]). For the sake of a simpler description, we fix the input length n and write C for the n-input circuit of depth $d \in \mathcal{O}(\log n)$. W. l. o. g. for every $x \in \{0,1\}^n$, we have $x \in L(M)$ if and only if the output gate of C evaluates to 0 on input x.

For each gate $v \in \{0,1\}^{\leq d}$, let $g_v = g_{d,v}$ as in Definition 1. We construct two G-programs P_v and P_v^{-1} (both of input length n) such that for every input $x \in \{0,1\}^n$ we have

$$P_v[x] =_G \begin{cases} g_v & \text{if gate } v \text{ evaluates to 1,} \\ 1 & \text{if gate } v \text{ evaluates to 0,} \end{cases} \tag{2}$$

and $P_v^{-1}[x] = P_v[x]^{-1}$ in G. Notice that $g_v P_v^{-1}[x] = g_v$ if v evaluates to 0 and $g_v P_v^{-1}[x] = 1$, otherwise. Thus, $g_v P_v^{-1}$ is a G-program for the "negation" of P_v. Moreover, by Equation (2), P_ε evaluates to 1 on input x if and only if the output gate evaluates to 0 which by our assumption is the case if and only if $x \in L$.

The construction of the P_v and P_v^{-1} is straightforward: For an input gate $v \in \{0,1\}^d$ with $q_n(v) = \langle i, a, b\rangle$ we define P_v to be a G-program evaluating to g_v or 1 depending on the i-th input bit. More precisely, write $g_v = a_1 \cdots a_m$ with $a_i \in \Sigma$. If $q_n(v) = \langle i, a, b\rangle$ for $i \in [1..n]$ and $a, b \in \{0,1\}$, we set $P_v = \langle i, a_1^a, a_1^b\rangle \cdots \langle i, a_m^a, a_m^b\rangle$ and $P_v^{-1} = \langle i, a_m^{-a}, a_m^{-b}\rangle \cdots \langle i, a_1^{-a}, a_1^{-b}\rangle$. For a nand-gate v with inputs from $v0$ and $v1$, we define

$$\begin{aligned} P_v &= g_v[P_{v1}, P_{v0}] = g_v P_{v1}^{-1} P_{v0}^{-1} P_{v1} P_{v0}, \\ P_v^{-1} &= [P_{v0}, P_{v1}] g_v^{-1} = P_{v0}^{-1} P_{v1}^{-1} P_{v1} P_{v1} g_v^{-1}, \end{aligned}$$

where the g_v and g_v^{-1} represent constant G-programs evaluating to g_v and g_v^{-1}, respectively, irrespective of the actual input (such constant G-programs consist of triples of the form $\langle 1, a, a\rangle$ for $a \in \Sigma$). These constant G-programs are defined via the commutator identities $g_v = \left[g_{v0}, g_{v1}\right]$ for $v \in \{0,1\}^{<d}$ in Definition 1.

Clearly, by induction we have $P_v[x]^{-1} = P_v^{-1}[x]$ in G (for every input x). Let us show that Equation (2) holds: For an input gate $v \in \{0,1\}^d$, Equation (2) holds by definition. Now, let $v \in \{0,1\}^{<d}$. Then, by induction, we have the following equalities in G:

$$\begin{aligned} P_v[x] = g_v[P_{v1}[x], P_{v0}[x]] &= \begin{cases} g_v & \text{if } v0 \text{ or } v1 \text{ evaluates to 0,} \\ g_v[g_{v1}, g_{v0}] & \text{if } v0 \text{ and } v1 \text{ evaluate to 1,} \end{cases} \\ &= \begin{cases} g_v & \text{if } v \text{ evaluates to 1,} \\ 1 & \text{if } v \text{ evaluates to 0.} \end{cases} \end{aligned}$$

Note that $[g_{v1}, g_{v0}] = [g_{v0}, g_{v1}]^{-1} = g_v^{-1}$ for the last equality. Thus, P_v satisfies Equation (2). For P_v^{-1} the analogous statement can be shown with the same calculation. For a leaf $v \in \{0,1\}^d$, we have $|g_v| \in 2^{\mathcal{O}(d)} = n^{\mathcal{O}(1)}$ by Condition (a) from Definition 1 (recall that $d \in \mathcal{O}(\log n)$). Hence, P_v^{-1} and P_v have polynomial length in n. Finally, also P_ε has polynomial length in n.

This gives us a non-uniform AC^0-reduction (more precisely, a projection reduction) of $L(M)$ to $\mathrm{WP}(G)$. In order to obtain a DLOGTIME-reduction, we apply essentially the same construction. However, we need to introduce some padding so that the function mapping an index i encoded in binary to the i-th instruction of P_ε can be computed in DLINTIME. In order to show the last point, we need condition (d) of the uniform SENS definition (Definition 1). ◀

Let us next state some algebraic properties of SENS groups. The proofs of the following three lemmas are straightforward.

▶ **Lemma 12.** *The property of being (uniformly) SENS is independent of the choice of the standard generating set.*

▶ **Lemma 13.** *If $Q = H/K$ is a f.g. subquotient of a f.g. group G and Q is (uniformly) SENS, then G is also (uniformly) SENS.*

▶ **Lemma 14.** *If G is (uniformly) SENS, then $G/Z(G)$ is (uniformly) SENS.*

The following result is, for $G = A_5$, the heart of Barrington's argument:

▶ **Lemma 15.** *If G is a finite non-solvable group, then G is uniformly SENS.*

Proof. Let us first show the statement for a non-abelian finite simple group G. By the proof of Ore's conjecture [34], every element of G is a commutator. This means that we may choose $g_\varepsilon \neq 1$ at will, and given g_v we define g_{v0}, g_{v1} by table lookup, having chosen once and for all for each element of G a representation of it as a commutator. Computing g_v requires $|v|$ steps and bounded memory.

If G is finite non-solvable, then any composition series of G contains a non-abelian simple composition factor G_i/G_{i+1}. Hence, we can apply Lemma 13. ◀

We remark that a direct proof of Lemma 15 without using the deep result [34] is also not difficult, but requires some more work.

By Lemmas 13 and 15, every group having a finite non-solvable subquotient is uniformly SENS. Since every free group of rank $n \geq 2$ projects to a finite non-solvable group, we get:

▶ **Corollary 16.** *A f.g. free group of rank $n \geq 2$ is uniformly SENS.*

This result was essentially shown by Robinson [41], who showed that the word problem of a free group of rank two is ALOGTIME-hard. He used a similar commutator approach as Barrington. One can prove Corollary 16 also directly by exhibiting a free subgroup of infinite rank whose generators are easily computable. For example, in the free group $F_2 = \langle x_0, x_1 \rangle$ take $g_{d,v} = x_0^{-v} x_1 x_0^v$ for $v \in \{0,1\}^d$ viewing the string v as a binary encoded number (the other $g_{d,v}$ for $v \in \{0,1\}^{<d}$ are then defined by the commutator identity in Definition 1), and appropriately padding with 1's. It is even possible to take the $g_{d,v}$ of constant length: consider a free group $F = \langle x_0, x_1, x_2 \rangle$ of rank 3 and the elements $g_{d,v} = x_{v \bmod 3}$ with v read as the binary representation of an integer. It is easy to see that the nested commutator $g_{d,\varepsilon}$ is non-trivial.

A dichotomy for linear groups. Instead of Corollary 4, we prove a slightly more detailed result. Recall that a group G is called C_1-by-C_2 for group classes C_1 and C_2 if G has a normal subgroup $K \in \mathsf{C}_1$ such that $G/K \in \mathsf{C}_2$.

▶ **Theorem 17.** *For every f.g. linear group the word problem is in* DLOGTIME*-uniform* TC^0 *or* ALOGTIME*-hard. More precisely: let G be a f.g. linear group.*

- *If G is finite solvable, then $WP(G)$ belongs to* DLOGTIME*-uniform* ACC^0*.*
- *If G is infinite solvable, then $WP(G)$ is complete for* DLOGTIME*-uniform* TC^0 *(via uniform* AC^0*-Turing-reductions).*
- *If G is solvable-by-(finite non-solvable), then $WP(G)$ is complete for* ALOGTIME *(via* DLOGTIME*-reductions).*
- *In all other cases, $WP(G)$ is* ALOGTIME*-hard and in* LOGSPACE*.*

Note that we can obtain a similar dichotomy for hyperbolic groups: they are either virtually abelian or contain a non-abelian free subgroup. In the first case, the word problem is in DLOGTIME-uniform TC^0, in the second case it is ALOGTIME-hard.

Proof. Let G be f.g. linear. First of all, by [36, 44], $\mathrm{WP}(G)$ belongs to LOGSPACE. By Tits alternative [46], G either contains a free subgroup of rank 2 or is virtually solvable. In the former case, $\mathrm{WP}(G)$ is ALOGTIME-hard by Corollary 16 and Theorem 2. Let us now assume that G is virtually solvable. Let K be a solvable subgroup of G of finite index. By taking the intersection of all conjugates of K in G, we can assume that K is a normal subgroup of G. If also G/K is solvable, then G is solvable. Hence, $\mathrm{WP}(G)$ is in DLOGTIME-uniform ACC^0 (if G is finite) or, by [31], complete for DLOGTIME-uniform TC^0 (if G is infinite). Finally, assume that the finite group G/K is non-solvable (thus, G is solvable-by-(finite non-solvable).

By Lemmas 13 and 15, G is uniformly SENS, and Theorem 2 implies that $\mathrm{WP}(G)$ is $\mathsf{ALOGTIME}$-hard. Moreover, by [41, Theorem 5.2], $\mathrm{WP}(G)$ is AC^0-reducible to $\mathrm{WP}(K)$ and $\mathrm{WP}(G/K)$. The latter belongs to $\mathsf{ALOGTIME}$ and $\mathrm{WP}(K)$ belongs to $\mathsf{DLOGTIME}$-uniform ACC^0 if K is finite and to $\mathsf{DLOGTIME}$-uniform TC^0 if K is infinite (note that K as a finite index subgroup of G is f.g. linear too). In all cases, $\mathrm{WP}(G)$ belongs to $\mathsf{ALOGTIME}$. ◀

New examples of SENS groups. In order to get new examples of (uniformly) SENS groups, we use the following result on groups with a certain self-embedding property with respect to wreath products.

▶ **Theorem 18.** *Let G be a finitely generated group with $G \wr H \leq G$ for some non-trivial group H. Then G is uniformly SENS.*

As an immediate consequence of this theorem and Lemma 8, we obtain:

▶ **Corollary 19.** *Thompson's groups $F < T < V$ are uniformly SENS.*

By the following result, Grigorchuk's group and the Gupta-Sidki groups are uniformly SENS.

▶ **Theorem 20.** *Let G be a weakly branched self-similar group, and assume that it admits a f.g. branching subgroup K. Then K and hence G are uniformly SENS.*

In the long version [5] we derive Theorems 18 and 20 from a single result. Roughly speaking, it states that a f.g. group G is uniformly SENS if G contains a subgroup $\langle h_0, h_1, h_2, \ldots \rangle$ that acts on a tree X^* (where X can be also infinite) in such a way that there exist $x_{-1}, x, x_1 \in X$ with the following properties for all $k \geq 0$:

(i) h_k only acts non-trivially on the subtree below x^k and

(ii) the three tree nodes $x^k x_{-1}$, $x^k x$, and $x^k x_1$ are consecutive in the orbit of h_k.

In addition, h_k must be of word length $2^{\mathcal{O}(k)}$ (with respect to the generators of G) and the symbol in h_k at a certain position must be computable in linear time.

Theorem 18 can be derived from this statement as follows: we can use the embedding $G \wr H \leq G$ in order to find in G a subgroup $\langle h_0, h_1, \ldots \rangle \cong (\cdots \wr \mathbb{Z}) \wr \mathbb{Z}$ or $\langle h_0, h_1, \ldots \rangle \cong (\cdots \wr (\mathbb{Z}/p)) \wr (\mathbb{Z}/p)$. This subgroup acts in a canonical way on the tree X^* for $X = \mathbb{Z}$ or $X = \mathbb{Z}/p$. Let the element h_k be a generator of the $(k+1)$-st cyclic factor from the right. Then h_0 cyclically permutes the children of the root ε, and, more generally, h_k cyclically permutes the children of the node 0^k and stabilizes all other nodes. Using an appropriate padding, the symbols of the h_k are computable in linear time, so we can apply the above mentioned result from the long version [5]. For weakly branched self-similar groups, after overcoming some minor technical difficulties, the proof follows the same outline.

Direct proofs for Thompson's and Grigorchuk's groups. One can also prove the uniform SENS property for Thompson's group F directly. Recall the infinite presentation $F = \langle x_0, x_1, x_2, \ldots \mid x_k^{x_i} = x_{k+1} (i < k) \rangle$.

▶ **Proposition 21.** *Let $g = x_3 x_2^{-1} \in F$ and define c_v for $v \in \{0,1\}^*$ inductively via*

$$c_\varepsilon = \varepsilon, \quad c_{v0} = x_1 c_v, \text{ and } c_{v1} = x_0^{-1} x_1 c_v.$$

Finally, for $d \in \mathbb{N}$ and $v \in \{0,1\}^{\leq d}$ let $g_{d,v} = g^{c_v}$. Then $g_{d,v} = [g_{d,v0}, g_{d,v1}]$ for all $d \in \mathbb{N}$ and $v \in \{0,1\}^{\leq d}$ and $g_\varepsilon = g \neq 1$ in F. In particular, G is uniformly SENS.

Proof. Obviously, we have $g_\varepsilon = g$ in F. Moreover, since $x_2 \neq x_3$ (which follows directly from the normal form theorem for F; see e.g. [12, Corollary 2.7]), we have $g \neq 1$ in F (Condition (c) of Definition 1). The identity $g = [g, g^{x_0^{-1}}]^{x_1} = [g^{x_1}, g^{x_0^{-1}x_1}]$ is straightforward to check. Thus, we obtain Condition (b):

$$g_{d,v} = g^{c_v} = [g^{x_1 c_v}, g^{x_0^{-1}x_1 c_v}] = [g^{c_{v0}}, g^{c_{v1}}] = [g_{d,v0}, g_{d,v1}].$$

Moreover, since $|g_{d,v}| = |g| + 2|c_v| \leq |g| + 4d$, we can pad with the identity symbol writing $g_{v,d}$ as a word of length $2^{\mu d}$ for some proper constant $\mu \in \mathbb{N}$ in order to meet Condition (a) (be aware that the lengths need to be computed over a finite standard generating set Σ, e. g. $\Sigma = \{1, x_0, x_0^{-1}, x_1, x_1^{-1}\}$). This shows that F is SENS.

Condition (d) of Definition 1 is also straightforward to check by introducing some more padding: we slightly change the definition of the c_v by setting $c_\varepsilon = \varepsilon, c_{v0} = 1x_1c_v$, and $c_{v1} = x_0^{-1}x_1c_v$ where $1 \in \Sigma$. This new c_v represents the same group element as the old c_v, but now we have $|c_v| = 2|v|$ for all $v \in \{0,1\}^*$ and all letters at even positions are x_1, while letters at odd positions are either 1 or x_0^{-1} (the $(2j+1)$-st letter of c_v depends on the $(|v| - j)$-th bit of v). Notice that we have $|g_{d,v}| = |g| + 2|c_v| = |g| + 4d$.

On input of $v \in \{0,1\}^d$, $i \in \mathbb{N}$ (encoded with μd bits), and $a \in \Sigma$, first decide whether $1 \leq i \leq 2d$, or $2d < i \leq 2d + |g|$, or $2d + |g| < i \leq 4d + |g|$, or $i > 4d + |g|$. This clearly can be done in DLINTIME (recall that g is a constant). Now assume that $1 \leq i \leq 2d$ (i. e., i points into the leading c_v^{-1} of $g_{d,v} = c_v^{-1}gc_v$). If i is odd, one accepts if and only if $a = x_1^{-1}$; if i is even, one accepts if and only if the $i/2$-th bit of v is zero and $a = 1$ or if the $i/2$-th bit of v is one and $a = x_0$. The other cases are similar. ◀

For the special case of Grigorchuk's group we give below an alternative proof for the uniform SENS property, where the $g_{d,v}$ are of constant length.

▶ **Proposition 22.** *Consider in the Grigorchuk group $G = \langle a, b, c, d\rangle$ the elements $x = (abad)^2$ and $y = x^b = bababadabac$. Define inductively $z_v \in \{x, y, x^{-1}, y^{-1}\}$ for $v \in \{0,1\}^*$: $z_\varepsilon = x$ and if z_v is defined, then we define z_{v0} and z_{v1} according to the following table:*

z_v	z_{v0}	z_{v1}
x	x^{-1}	y^{-1}
x^{-1}	y^{-1}	x^{-1}
y	y	x
y^{-1}	x	y

For every $d \in \mathbb{N}$ and $v \in \{0,1\}^{\leq d}$ let $g_{d,v} = z_v$ for $|v| = d$ and $g_{d,v} = [g_{d,v0}, g_{d,v1}]$ for $|v| < d$. We then have $g_{d,\varepsilon} \neq 1$ in G. In particular, G is uniformly SENS.

Proof. That $x \neq 1 \neq y$ is easy to check by computing the action of x and y on the third level of the tree. Now the following equations are easy to check in G:

$$[x, y] = \langle\!\langle 1, \langle\!\langle 1, y^{-1}\rangle\!\rangle\rangle\!\rangle \qquad [x^{-1}, y^{-1}] = \langle\!\langle 1, \langle\!\langle 1, x\rangle\!\rangle\rangle\!\rangle$$
$$[y, x] = \langle\!\langle 1, \langle\!\langle 1, y\rangle\!\rangle\rangle\!\rangle \qquad [y^{-1}, x^{-1}] = \langle\!\langle 1, \langle\!\langle 1, x^{-1}\rangle\!\rangle\rangle\!\rangle$$

Hence, $[z_{v0}, z_{v1}] = \langle\!\langle 1, \langle\!\langle 1, z_v\rangle\!\rangle\rangle\!\rangle$. The checks are tedious to compute by hand, but easy in the GAP package FR (note that vertices are numbered from 1 in GAP and from 0 here):

```
gap> LoadPackage("fr");
gap> AssignGeneratorVariables(GrigorchukGroup);
gap> x := (a*b*a*d)^2; y := x^b;
```

```
gap> Assert(0,Comm(x,y) = VertexElement([2,2],y^-1));
gap> Assert(0,Comm(x^-1,y^-1) = VertexElement([2,2],x));
gap> Assert(0,Comm(y,x) = VertexElement([2,2],y));
gap> Assert(0,Comm(y^-1,x^-1) = VertexElement([2,2],x^-1));
```

We claim that $g_{d,\varepsilon} \neq 1$ in G. The equation $[z_{v0}, z_{v1}] = \langle\!\langle 1, \langle\!\langle 1, z_v \rangle\!\rangle \rangle\!\rangle$ immediately implies that $g_{d,v}$ acts as z_v on the subtree below vertex $1^{2(d-|v|)}$ and trivially elsewhere. In particular, $g_{d,\varepsilon}$ acts as $z_\varepsilon = x \neq 1$ on the subtree below vertex 1^{2d} and is non-trivial. With this definition, the $g_{d,v}$ satisfy the definition of a SENS group. Moreover, given some $v \in \{0,1\}^d$, $g_{d,v}$ can be computed in time $\mathcal{O}(d)$ by a deterministic finite automaton with state set $\{x^{\pm 1}, y^{\pm 1}\}$. ◀

5 Circuit value problems for wreath products

This section provides further details regarding Theorem 6. We start with two applications for Mod-classes (applications for PSPACE can be found in the next section). For a complexity class C we define the class $\mathsf{Mod}_m\mathsf{C}$ by $L \in \mathsf{Mod}_m\mathsf{C}$ if there exists a polynomial $p(n)$ and a language $K \in \mathsf{C}$ such that $L = \{u \mid |\{v \in \{0,1\}^{p(|u|)} : u\#v \in K\}| \not\equiv 0 \mod m\}$.

▶ **Example 23.** If G is a finite non-abelian p-group, then $\mathsf{LEAF}(\mathrm{WP}(G)) \subseteq \mathsf{Mod}_p \cdots \mathsf{Mod}_p\mathsf{P} = \mathsf{Mod}_p\mathsf{P} \subseteq \mathsf{LEAF}(\mathrm{WP}(G))$ by [22, Satz 4.32], [9, Theorem 6.7], and [24, Theorem 2.2] and likewise $\mathsf{LEAF}(\mathrm{WP}(G/Z(G))) = \mathsf{Mod}_p\mathsf{P}$. Hence, in this case $\mathrm{CVP}(G \wr \mathbb{Z})$ is complete for $\forall\mathsf{Mod}_p\mathsf{P}$.

▶ **Example 24.** Consider the symmetric group on three elements S_3. By [24, Example 2.5] we have $\mathsf{LEAF}(\mathrm{WP}(S_3)) = \mathsf{Mod}_3\mathsf{Mod}_2\mathsf{P}$ (also written as $\mathsf{Mod}_3{\oplus}\mathsf{P}$). Since S_3 has trivial center, it follows that $\mathrm{CVP}(S_3 \wr \mathbb{Z})$ is complete for $\forall\mathsf{Mod}_3{\oplus}\mathsf{P}$.

In the rest of the section, we outline the proof of Theorem 6; full details can be found in the appendix. The first statement of Theorem 6 (that $\mathrm{CVP}(G \wr \mathbb{Z})$ belongs to $\forall\mathsf{LEAF}(\mathrm{WP}(G))$) is the easy part: Let Σ be a standard generating set for G and fix a generator t for $\mathbb{Z}$. Then $\Gamma = \Sigma \cup \{t, t^{-1}\}$ is a standard generating set for $G \wr \mathbb{Z}$. Consider a circuit over the wreath product $G \wr \mathbb{Z}$ whose input gates are labelled with generators from Γ. We can evaluate this circuit also in the free monoid Γ^* and obtain a word $w \in \Gamma^*$ as the evaluation of the output gate. We have to verify that $w = 1$ in $G \wr \mathbb{Z}$. One first checks in polynomial time whether the exponent sum of t in w is zero. If not, the algorithm rejects, otherwise the word w represents in $G \wr \mathbb{Z}$ a function $f\colon \mathbb{Z} \to G$ with finite support. One can easily compute in polynomial time two binary encoded integers i, j such that $\mathrm{supp}(f)$ is contained in $[i..j]$ (the integer interval from i to j). It remains to verify that $\forall x \in [i..j] : f(x) = 1$. The $\forall$-quantifier over $[i..j]$ corresponds to the $\forall$-part in $\forall\mathsf{LEAF}(\mathrm{WP}(G))$. Finally, for a specific number $x \in [i..j]$ the machine then produces a leaf string $w_x \in \Sigma^*$ such that w_x represents the group element $f(x) \in G$. Basically, the machine branches to all positions p in the word w and prints the symbol a at that position p, if (i) $a \in \Sigma$ and (ii) the exponent sum of all t's in the prefix up to position $p-1$ in w is x (this can be checked in polynomial time). Otherwise, the machine prints the padding letter 1.

The hard part of Theorem 6 is showing hardness for $\forall\mathsf{LEAF}(\mathrm{WP}(G/Z(G)))$. The proof for this uses some of the techniques from the paper [37], where a connection between leaf strings and string compression was established. Instead of going into the details (which can be found in Appendix C) we want to explain another perspective on the theorem. Let us restrict to the case that the center of G is trivial; hence the circuit value problem for $G \wr \mathbb{Z}$ is complete for $\forall\mathsf{LEAF}(\mathrm{WP}(G))$. Also fix a standard generating set Γ for G. Recall that a circuit over

the group G can be seen also as a succinct representation of a string over the alphabet Γ, which is obtained by evaluating the circuit in the free monoid Γ^*. The circuit value problem for G then asks whether this word belongs to $\mathrm{WP}(G)$. Leaf languages correspond to an even more succinct form of string compression by Boolean circuits. A Boolean circuit $\mathcal{C}$ with n inputs represents the binary string of length 2^n, where the i-th symbol is 1 if and only if $\mathcal{C}$ evaluates to true under the i-th truth assignment. In order to represent an arbitrary string $w \in \Gamma^*$ by a Boolean circuit one has to (i) fix an encoding of the symbols from Γ by binary strings and (ii) specify in addition to the circuit also the length of w. It is then well-known that the question whether a string specified by a Boolean circuit belongs to a fixed language K is complete for $\mathsf{LEAF}(K)$ (strictly speaking this is only true if $\mathsf{LEAF}(K)$ is replaced by the corresponding balanced leaf language class, but for $K = \mathrm{WP}(G)$ this makes no difference due to the padding letter 1; see Appendix A). Compression by Boolean circuits is much more succinct than compression by group circuits. For instance, for a finite non-solvable group G, $\mathsf{LEAF}(\mathrm{WP}(G)) = \mathsf{PSPACE}$ but $\mathrm{CVP}(G)$ is P-complete. Roughly speaking, Theorem 6 says that compression by group circuits over the wreath product $G \wr \mathbb{Z}$ has the same power as compression by Boolean circuits over the group G.

6 PSPACE-complete circuit value problems

In this section we apply Theorem 6 to SENS groups. The results are summarized in Corollary 7 from the introduction. The proofs of this section can be found in the full version [5].

The following result can be derived from Theorem 2 using a padding argument (recall that $\mathsf{PSPACE} = \mathsf{APTIME}$). Another point of view is that Lemma 25 generalizes the inclusion $\mathsf{PSPACE} \subseteq \mathsf{LEAF}(\mathrm{WP}(G))$ for a finite simple group (in which case equality holds) [25]. Note that, by Lemma 14, $G/Z(G)$ is uniformly SENS if G is uniformly SENS.

▶ **Lemma 25.** *If G is uniformly SENS, then $\mathsf{PSPACE} \subseteq \mathsf{LEAF}(WP(G/Z(G)))$.*

From Theorem 6 and Lemma 25 we get:

▶ **Corollary 26.** *If G is uniformly SENS, then $CVP(G \wr \mathbb{Z})$ is PSPACE-hard.*

We can apply Corollary 26 to wreath products $G \wr \mathbb{Z}$ where G is finite non-solvable or free of rank $n \geq 2$ (statement i in Corollary 7 from the introduction). In this case, the word problem for $G \wr \mathbb{Z}$ can be solved in $\mathsf{LOGSPACE}$ (which follows from a transfer theorem for wreath products [48] and the fact the word problem for finite groups and free groups can be solved in $\mathsf{LOGSPACE}$ [36]). This in turn implies that $\mathrm{CVP}(G \wr \mathbb{Z})$ belongs to PSPACE.

For Thompson's group F we have $F \wr \mathbb{Z} \leq F$ (Lemma 8). Moreover, F is uniformly SENS (Corollary 19). Hence, Corollary 26 shows that $\mathrm{CVP}(F)$ is PSPACE-hard. Furthermore, $\mathrm{CVP}(F)$ belongs to PSPACE. This follows from the fact that F is co-context-free, i.e., the complement of the word problem of F is a context-free language [33] (this is independent of the finite generating set). We obtain statement ii in Corollary 7.

Finally, statements iii and iv from Corollary 7 are consequences of the following result:

▶ **Corollary 27.** *If G is a weakly branched torsion group whose branching subgroup is f.g., then $CVP(G)$ is PSPACE-hard. If, in addition, G is contracting, then $CVP(G)$ is PSPACE-complete.*

Proof sketch. The second statement follows easily from the first statement since for a contracting group the word problem belongs to $\mathsf{LOGSPACE}$ (Proposition 11), which implies that the circuit value problem belongs to PSPACE (see Lemma 34 in the appendix). For the

first statement, the main observation is that the hardness proof for the second statement of Theorem 6 (see Appendix C) uses only a subinterval of the integers whose length is exponentially bounded in the input length. This means that it suffices to find a copy of $G \wr (\mathbb{Z}/p^n)$ in G for each n and some fixed $p \geq 2$. Moreover, the embedding $\varphi\colon G \wr (\mathbb{Z}/p^n) \to G$ must be circuit-efficient in the sense that for every generator a of $G \wr (\mathbb{Z}/p^n)$ one can compute from n (given in unary notation) a circuit $\mathcal{G}_{n,a}$ for $\varphi(a)$. For the case of a weakly branched torsion group G whose branching subgroup K is finitely generated we cannot show the existence of such a circuit-efficient embedding for G itself, but we can prove it for K (in fact, it suffices to show an embedding $K \wr (\mathbb{Z}/p) \leq K$, which can then be iterated n-times in order to get the embedding $K \wr (\mathbb{Z}/p^n) \leq K$). This implies that the circuit value problem for K (and hence for G) is PSPACE-hard. ◀

7 Conclusion and open problems

We have added an algorithmic constraint (uniformly SENS) to the algebraic notion of being a non-solvable group, which implies that the word problem is ALOGTIME-hard. Using this, we produced several new examples of non-solvable groups with an ALOGTIME-hard word problem. However, the question remains open whether every non-solvable group has an ALOGTIME-hard word problem, even if it is not SENS. We showed that for every contracting self-similar group the word problem belongs to LOGSPACE. Here, the question remains whether there exists a contracting self-similar group with a LOGSPACE-complete word problem. In particular, is the word problem for the Grigorchuk group LOGSPACE-complete? (We proved that it is ALOGTIME-hard.) Also the precise complexity of the word problem for Thompson's group F is open. It is ALOGTIME-hard and belongs to LOGCFL; the latter follows from [33]. In fact, from the proof in [33] one can deduce that the word problem for F belongs to LOGDCFL (the closure of the deterministic context-free languages with respect to LOGSPACE-reductions).

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 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. doi:10.1017/CBO9780511804090.

3 David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989. doi:10.1016/0022-0000(89)90037-8.

4 David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35:941–952, 1988. doi:10.1145/48014.63138.

5 Laurent Bartholdi, Michael Figelius, Markus Lohrey, and Armin Weiß. Groups with ALOGTIME-hard word problems and PSPACE-complete compressed word problems. Technical report, arXiv.org, 2020. arXiv:1909.13781.

6 Laurent Bartholdi, Rostislav I. Grigorchuk, and Zoran Šuniḱ. Branch groups. In *Handbook of algebra, Volume 3*, pages 989–1112. Elsevier/North-Holland, Amsterdam, 2003. doi:10.1016/S1570-7954(03)80078-5.

7 Laurent Bartholdi and Volodymyr V. Nekrashevych. Iterated monodromy groups of quadratic polynomials. I. *Groups, Geometry, and Dynamics*, 2(3):309–336, 2008. doi:10.4171/GGD/42.

8 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. doi:10.1137/S0097539793249530.

9 Richard Beigel and John Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103(1):3–23, 1992. doi:10.1016/0304-3975(92)90084-S.

10 William W. Boone. The Word Problem. *Annals of Mathematics*, 70(2):207–265, 1959. URL: www.jstor.org/stable/1970103.

11 Daniel P. Bovet, Pierluigi Crescenzi, and Riccardo Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, 104(2):263–283, 1992. doi:10.1016/0304-3975(92)90125-Y.

12 John W. Cannon, William J. Floyd, and Walter R. Parry. Introductory notes on Richard Thompson's groups. *L'Enseignement Mathématique*, 42(3):215–256, 1996.

13 Moses Charikar, Eric 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. doi:10.1109/TIT.2005.850116.

14 Max Dehn. Über unendliche diskontinuierliche Gruppen. *Mathematische Annalen*, 71(1):116–144, 1911. doi:10.1007/BF01456932.

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

16 Max Garzon and Yechezkel Zalcstein. The complexity of Grigorchuk groups with application to cryptography. *Theoretical Computer Science*, 88(1):83–98, 1991. doi:10.1016/0304-3975(91)90074-C.

17 Rostislav I. Grigorchuk. Burnside's problem on periodic groups. *Functional Analysis and Its Applications*, 14:41–43, 1980. doi:10.1007/BF01078416.

18 Rostislav I. Grigorchuk and Zoran Šunić. Asymptotic aspects of Schreier graphs and Hanoi Towers groups. *C. R. Math. Acad. Sci. Paris*, 342(8):545–550, 2006. doi:10.1016/j.crma.2006.02.001.

19 Victor S. Guba and Mark V. Sapir. On subgroups of the R. Thompson group F and other diagram groups. *Matematicheskii Sbornik*, 190(8):3–60, 1999. doi:10.1070/SM1999v190n08ABEH000419.

20 Narain Gupta and Saïd Sidki. On the Burnside problem for periodic groups. *Mathematische Zeitschrift*, 182(3):385–388, 1983. doi:10.1007/BF01179757.

21 Frédéric Haglund and Daniel T. Wise. Coxeter groups are virtually special. *Advances in Mathematics*, 224(5):1890–1903, 2010. doi:10.1016/j.aim.2010.01.011.

22 Ulrich Hertrampf. *Über Komplexitätsklassen, die mit Hilfe von k-wertigen Funktionen definiert werden*. Habilitationsschrift, Universität Würzburg, 1994.

23 Ulrich Hertrampf. The shapes of trees. In *Proceedings of COCOON 1997*, volume 1276 of *Lecture Notes in Computer Science*, pages 412–421. Springer, 1997. doi:10.1007/BFb0045108.

24 Ulrich Hertrampf. Algebraic acceptance mechanisms for polynomial time machines. *SIGACT News*, 31(2):22–33, 2000. doi:10.1145/348210.348215.

25 Ulrich Hertrampf, Clemens Lautemann, Thomas Schwentick, Heribert Vollmer, and Klaus W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 200–207. IEEE Computer Society Press, 1993. doi:10.1109/SCT.1993.336526.

26 Ulrich Hertrampf, Heribert Vollmer, and Klaus Wagner. On balanced versus unbalanced computation trees. *Mathematical Systems Theory*, 29(4):411–421, 1996. doi:10.1007/BF01192696.

27 Derek F. Holt, Markus Lohrey, and Saul Schleimer. Compressed decision problems in hyperbolic groups. In *Proceedings of STACS 2019*, volume 126 of *LIPIcs*, pages 37:1–37:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: http://www.dagstuhl.de/dagpub/978-3-95977-100-9.

28 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. doi:10.1017/9781316588246.

29 Birgit Jenner, Pierre McKenzie, and Denis Thérien. Logspace and logtime leaf languages. *Information and Computation*, 129(1):21–33, 1996. doi:10.1006/inco.1996.0071.

30 Howard J. Karloff and Walter L. Ruzzo. The iterated mod problem. *Information and Computation*, 80(3):193–204, 1989. doi:10.1016/0890-5401(89)90008-4.

31 Daniel König and Markus Lohrey. Evaluation of circuits over nilpotent and polycyclic groups. *Algorithmica*, 80(5):1459–1492, 2018. doi:10.1007/s00453-017-0343-z.

32 Daniel König and Markus Lohrey. Parallel identity testing for skew circuits with big powers and applications. *International Journal of Algebra and Computation*, 28(6):979–1004, 2018. doi:10.1142/S0218196718500431.

33 Jörg Lehnert and Pascal Schweitzer. The co-word problem for the Higman-Thompson group is context-free. *Bulletin of the London Mathematical Society*, 39(2):235–241, February 2007. doi:10.1112/blms/bdl043.

34 Martin W. Liebeck, Eamonn A. O'Brien, Aner Shalev, and Pham Huu Tiep. The Ore conjecture. *Journal of the European Mathematical Society*, 12(4):939–1008, 2010. doi:10.4171/JEMS/220.

35 Yury Lifshits and Markus Lohrey. Querying and embedding compressed texts. In *Proceedings of MFCS 2006*, volume 4162 of *Lecture Notes in Computer Science*, pages 681–692. Springer, 2006. doi:10.1007/11821069_59.

36 Richard J. Lipton and Yechezkel Zalcstein. Word problems solvable in logspace. *Journal of the Association for Computing Machinery*, 24(3):522–526, 1977. doi:10.1145/322017.322031.

37 Markus Lohrey. Leaf languages and string compression. *Information and Computation*, 209(6):951–965, 2011. doi:10.1016/j.ic.2011.01.009.

38 Markus Lohrey. *The Compressed Word Problem for Groups*. Springer Briefs in Mathematics. Springer, 2014. doi:10.1007/978-1-4939-0748-9.

39 Volodymyr Nekrashevych. *Self-similar groups*, volume 117 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2005. doi:10.1090/surv/117.

40 Piotr S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov*, pages 1–143, 1955. In Russian. URL: http://mi.mathnet.ru/eng/tm1180.

41 David Robinson. *Parallel Algorithms for Group Word Problems*. PhD thesis, University of California, San Diego, 1993.

42 Joseph J. Rotman. *An Introduction to the Theory of Groups (fourth edition)*. Springer, 1995.

43 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. doi:10.1561/0400000039.

44 Hans-Ulrich Simon. Word problems for groups and contextfree recognition. In *Proceedings of FCT 1979*, pages 417–422. Akademie-Verlag, 1979.

45 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of STOC 1987*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

46 Jacques Tits. Free subgroups in linear groups. *Journal of Algebra*, 20(2):250–270, 1972. doi:10.1016/0021-8693(72)90058-0.

47 Heribert Vollmer. *Introduction to Circuit Complexity*. Springer, Berlin, 1999. doi:10.1007/978-3-662-03927-4.

48 Stephan Waack. The parallel complexity of some constructions in combinatorial group theory. *Journal of Information Processing and Cybernetics EIK*, 26:265–281, 1990. doi:10.1007/BFb0029647.

49 Jan Philipp Wächter and Armin Weiß. An automaton group with pspace-complete word problem. In *Proceedings of STACS 2020*, volume 154 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://www.dagstuhl.de/dagpub/978-3-95977-140-5.

50 John S. Wilson. Embedding theorems for residually finite groups. *Mathematische Zeitschrift*, 174(2):149–157, 1980. doi:10.1007/BF01293535.

51 Daniel T. Wise. Research announcement: the structure of groups with a quasiconvex hierarchy. *Electronic Research Announcements in Mathematical Sciences*, 16:44–55, 2009. URL: http://aimsciences.org//article/id/8d3fc128-8d02-4fee-af67-38001ca1d0ac.

A Leaf languages

In the following, we introduce more details concerning leaf languages that were briefly explained in the introduction. An NTM M with input alphabet Γ is called *adequate*, if (i) for every input $x \in \Gamma^*$, M does not have an infinite computation on input x, (ii) the finite set of transition tuples of M is linearly ordered, and (iii) when terminating M prints a symbol $\alpha(q)$ from a finite alphabet Σ, where q is the current state of M. For an input $x \in \Gamma^*$, we define the computation tree by unfolding the configuration graph of M from the initial configuration. By condition (i) and (ii), the computation tree can be identified with a finite ordered tree $T(x) \subseteq \mathbb{N}^*$. For $u \in T(x)$ let $q(u)$ be the M-state of the configuration that is associated with the tree node u. Then, the leaf string $\mathrm{leaf}(M, x)$ is the string $\alpha(q(v_1)) \cdots \alpha(q(v_k)) \in \Sigma^+$, where $v_1, \ldots, v_k$ are all leaves of $T(x)$ listed in lexicographic order.

A *complete binary tree* is a rooted tree, where every non-leaf node has a left and a right child, and every path from the root to a leaf has the same length. An adequate NTM M is called *balanced*, if for every input $x \in \Gamma^*$, the computation $T(x)$ (produced by M on input x) is a complete binary tree. In Section 3 we defined the complexity class $\mathsf{LEAF}(K) = \{\mathsf{LEAF}(M, K) \mid M \text{ is an adequate polynomial time NTM}\}$ for a language $K \subseteq \Sigma^*$. Let us define the subclass

$$\mathsf{bLEAF}(K) = \{\mathsf{LEAF}(M, K) \mid M \text{ is a balanced polynomial time NTM}\}.$$

Both classes $\mathsf{LEAF}(K)$ and $\mathsf{bLEAF}(K)$ are closed under polynomial time reductions. We clearly have $\mathsf{bLEAF}(K) \subseteq \mathsf{LEAF}(K)$. The following result was shown in [29] by padding computation trees to complete binary trees.

▶ **Lemma 28.** *Assume that $K \subseteq \Sigma^*$ is a language such that Σ contains a symbol 1 with the following property: if $uv \in K$ for $u, v \in \Sigma^*$ then $u1v \in K$. Then $\mathsf{LEAF}(K) = \mathsf{bLEAF}(K)$.*

In particular, we obtain the following lemma:

▶ **Lemma 29.** *Let G be a finitely generated group and Σ a finite standard generating set for G. Then $\mathsf{LEAF}(WP(G, \Sigma)) = \mathsf{bLEAF}(WP(G, \Sigma))$.*

Moreover, we have:

▶ **Lemma 30.** *Let G be finitely generated group and Σ, Γ finite standard generating sets for G. Then $\mathsf{LEAF}(WP(G, \Sigma)) = \mathsf{LEAF}(WP(G, \Gamma))$.*

Proof. Consider a language $L \in \mathsf{LEAF}(\mathrm{WP}(G, \Sigma))$. Thus, there exists an adequate polynomial time NTM M such that $L = \mathsf{LEAF}(M, \mathrm{WP}(G, \Sigma))$. We modify M as follows: If M terminates and prints the symbol $a \in \Sigma$, it enters a small nondeterministic subcomputation that produces the leaf string w_a, where $w_a \in \Gamma^*$ is a word that evaluates to the same group element as a. Let M' be the resulting adequate polynomial time NTM. It follows that $\mathsf{LEAF}(M, \mathrm{WP}(G, \Sigma)) = \mathsf{LEAF}(M', \mathrm{WP}(G, \Gamma))$. ◀

As remarked in the main part, Lemma 29 allows to just write $\mathsf{LEAF}(\mathrm{WP}(G))$ (as well as $\mathsf{bLEAF}(\mathrm{WP}(G))$). In [25] it was shown that $\mathsf{PSPACE} = \mathsf{LEAF}(\mathrm{WP}(G))$ for every finite non-solvable group.

B Compressed words and the circuit value problem

We mentioned in the introduction that the circuit value problem for a f.g. group G can be seen as a succinct version of the word problem, where the input word is succinctly represented by a context-free grammar that produces exactly one word. Such a context-free grammar is

also called a straight-line program in the literature on string compression. This alternative viewpoint on the circuit value problem turns out to be convenient for our proofs. In the following we will elaborate this viewpoint.

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 $\{(A, B) \in V \times V : B \text{ occurs in } \rho(A)\}$ 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 on terminal word, which will be denoted by $\mathrm{val}(\mathcal{G})$. Formally, one can extend ρ to a morphism $\rho : (V \cup \Sigma)^* \to (V \cup \Sigma)^*$ by setting $\rho(a) = a$ for all $a \in \Sigma$. The above acyclicity condition on ρ implies that for $m = |V|$ we have $\rho^m(w) \in \Sigma^*$ for all $w \in (V \cup \Sigma)^*$. We then define $\mathrm{val}_{\mathcal{G}}(w) = \rho^m(w)$ (the string derived from the sentential form w) and $\mathrm{val}(\mathcal{G}) = \mathrm{val}_{\mathcal{G}}(S)$. We define the size of the SLP $\mathcal{G}$ 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 $0 \leq i \leq n-1$.

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 $\mathrm{val}(\mathcal{G}) \neq \varepsilon$ into an SLP $\mathcal{G}'$ in Chomsky normal form with $\mathrm{val}(\mathcal{G}) = \mathrm{val}(\mathcal{G}')$, see e.g. [38, Proposition 3.8].

The circuit value problem $\mathrm{CVP}(G)$ for a f.g. group G (as defined in the introduction) is equivalent to the following problem, where Σ is a finite standard generating set for G:
Input: an SLP $\mathcal{G}$ over the alphabet Σ.
Question: does $\mathrm{val}(\mathcal{G}) = 1$ hold in G?
It is an easy observation that the computational complexity of the circuit value problem for G does not depend on the chosen generating set Σ: More precisely, if we denote for a moment with $\mathrm{CVP}(G, \Sigma)$ the circuit value problem for the specific generating set Σ, then for all generating sets Σ and Σ' for G, $\mathrm{CVP}(G, \Sigma)$ is LOGSPACE-reducible to $\mathrm{CVP}(G, \Sigma')$ [38, Lemma 4.2]. The circuit value problem for G is also known as the compressed word problem for G [38].

We need a couple of known results for SLPs. For this we use the following notations on strings (which will be also needed in Appendix C). Take a word $w = a_0 \cdots a_{n-1} \in \Sigma^*$ over the alphabet Σ ($n \geq 0$, $a_0, \ldots, a_{n-1} \in \Sigma$). For $0 \leq i < n$ let $w[i] = a_i$ and for $0 \leq i \leq j < n$ let $w[i : j] = a_i a_{i+1} \cdots a_j$. Moreover $w[: i] = w[0 : i]$. Note that in the notations $w[i]$ and $w[i : j]$ we take 0 as the first position in w. This will be convenient in Appendix C. Finally, with $|w|_a = |\{i \mid a_i = a\}|$ we denote the number of occurrences of a in w.

▶ **Lemma 31** (c.f. [13]). *For every SLP $\mathcal{G}$ we have $|\mathrm{val}(\mathcal{G})| \leq 3^{|\mathcal{G}|/3}$.*

▶ **Lemma 32** ([38, Chapter 3]). *The following problems can be solved in polynomial time, where $\mathcal{G}$ is an SLP over a terminal alphabet Σ, $a \in \Sigma$, and $p, q \in \mathbb{N}$ are numbers given in binary notation:*

- *Given $\mathcal{G}$, compute the length $|\mathrm{val}(\mathcal{G})|$ of the word $\mathrm{val}(\mathcal{G})$.*
- *Given $\mathcal{G}$ and a, compute the number $|\mathrm{val}(\mathcal{G})|_a$ of occurrences of a in $\mathrm{val}(\mathcal{G})$.*
- *Given $\mathcal{G}$ and p, compute the symbol $\mathrm{val}(\mathcal{G})[p] \in \Sigma$ (in case $0 \leq p < |\mathrm{val}(\mathcal{G})|$ does not hold, the algorithm outputs a special symbol).*
- *Given $\mathcal{G}$ and p, q, compute an SLP for the string $\mathrm{val}(\mathcal{G})[p : q]$ (in case $0 \leq p \leq q < |\mathrm{val}(\mathcal{G})|$ does not hold, the algorithm outputs a special symbol).*

▶ **Lemma 33** (c.f. [38, Lemma 3.12]). *Given a symbol $a_0 \in \Sigma$ and a sequence of morphisms $\varphi_1, \ldots, \varphi_n : \Sigma^* \to \Sigma^*$, where every φ_i is given by a list of the words $\varphi_i(a)$ for $a \in \Sigma$, one can compute in* LOGSPACE *an SLP for the word $\varphi_1(\varphi_2(\cdots \varphi_n(a_0) \cdots))$.*

The next lemma follows easily by evaluating a given SLP and then running an algorithm for the "ordinary" word problem:

▶ **Lemma 34.** *If G is a finitely generated group such that $WP(G)$ can be solved in polylogarithmic space, then $CVP(G)$ belongs to* PSPACE.

C Additional details for Section 5

This section presents a detailed proof of Theorem 6. Instead of circuits over groups we will use the equivalent framework of SLPs from Appendix B. The proof of the lower bound in Theorem 6 uses some of the techniques from the paper [37], where a connection between leaf strings and SLPs was established. In Sections C.1–C.3 we will introduce these techniques. The proof of Theorem 6 will be given in Appendix C.4.

Let us first fix some general notations. For integers $i \leq j$ we write $[i..j]$ for the interval $\{z \in \mathbb{Z} \mid i \leq z \leq j\}$. In the following, we will identify a bit string $\alpha = a_1 a_2 \cdots a_n$ $(a_1, \ldots, a_n \in \{0,1\})$ with the vector $(a_1, a_2, \ldots, a_n)$. In particular, for another vector $\overline{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{N}^n$ we will write $\alpha \cdot \overline{s} = \sum_{i=1}^{n} a_i \cdot s_i$ for the scalar product. Moreover, we write $\sum \overline{s}$ for the sum $s_1 + s_2 + \cdots + s_n$.

C.1 Subsetsum problems

A sequence $(s_1, \ldots, s_n)$ of natural numbers is *super-decreasing* if $s_i > s_{i+1} + \cdots + s_n$ for all $1 \leq i \leq n$. For example, $(s_1, \ldots, s_n)$ with $s_i = 2^{n-i}$ is super-decreasing. An instance of the *subsetsum problem* is a tuple $(t, s_1, \ldots, s_k)$ of binary encoded natural numbers. It is a positive instance if there are $a_1, \ldots, a_k \in \{0,1\}$ such that $t = a_1 s_1 + \cdots + a_k s_k$. Subsetsum is a classical NP-complete problem, see e.g. [15]. The *super-decreasing subsetsum* problem is the restriction of subsetsum to instances $(t, s_1, \ldots, s_k)$, where $(s_1, \ldots, s_k)$ is super-decreasing. In [30] it was shown that super-decreasing subsetsum is P-complete.[1] We need a slightly generalized version of the construction showing P-hardness that we discuss in Appendix C.2.

C.2 From Boolean circuits to super-decreasing subsetsum

For this section, we have to fix some details on Boolean circuits. Let us consider a Boolean circuit C with input gates $x_1, \ldots, x_m$ and output gates $y_0, \ldots, y_{n-1}$.[2] We view C as a directed acyclic graph with multi-edges (there can be two edges between two nodes); the nodes are the gates of the circuit. The number of incoming edges of a gate is called its *fan-in* and the number of outgoing edges is the *fan-out*. Every input gate x_i has fan-in zero and every output gate y_i has fan-out zero. Besides the input gates there are two more gates c_0 and c_1 of fan-in zero, where c_i carries the constant truth value $i \in \{0,1\}$. Besides $x_1, \ldots, x_m, c_0, c_1$ every other gate has fan-in two and computes the nand of its two input gates. Moreover, we assume that every output gate y_i is a nand-gate. For a bit string $\alpha = b_1 \cdots b_m$ $(b_1, \ldots, b_m \in \{0,1\})$ and $0 \leq i \leq n-1$ we denote with $C(\alpha)_i$ the value of the output gate y_i when every input gate x_j $(1 \leq j \leq m)$ is set to b_j. Thus, C defines a map $\{0,1\}^m \to \{0,1\}^n$.

[1] In fact, [30] deals with the *super-increasing* subsetsum problem. But this is only a nonessential detail. For our purpose, super-decreasing sequences are more convenient.

[2] It will be convenient for us to number the input gates from 1 and the output gates from 0.

We assume now that C is a Boolean circuit as above with the following additional property that will be satisfied later: For all input bit strings $\alpha \in \{0,1\}^m$ there is exactly one $i \in [0..n-1]$ such that $C(\alpha)_i = 1$. Using a refinement of the construction from [30] we compute in LOGSPACE $q_0, \ldots, q_{n-1} \in \mathbb{N}$ and two super-decreasing sequences $\overline{r} = (r_1, \ldots r_m)$ and $\overline{s} = (s_1, \ldots, s_k)$ for some k (all numbers are represented in binary notation) with the following properties:

- The $r_1, \ldots, r_m$ are pairwise distinct powers of 4.
- For all $0 \leq i \leq n-1$ and all $\alpha \in \{0,1\}^m$: $C(\alpha)_i = 1$ if and only if there exists $\delta \in \{0,1\}^k$ such that $\delta \cdot \overline{s} = q_i + \alpha \cdot \overline{r}$.

Let us first add for every input gate x_i two new nand-gates $\bar{x}_i$ and $\bar{\bar{x}}_i$, where $\bar{\bar{x}}_i$ has the same outgoing edges as x_i. Moreover we remove the old outgoing edges of x_i and replace them by the edges $(x_i, \bar{x}_i)$, $(c_1, \bar{x}_i)$ and two edges from $\bar{x}_i$ to $\bar{\bar{x}}_i$. This has the effect that every input gate x_i has a unique outgoing edge. Clearly, the new circuit computes the same Boolean function (basically, we introduce two negation gates for every input gate). Let $g_1, \ldots, g_p$ be the nand-gates of the circuit enumerated in reverse topological order, i.e., if there is an edge from gate g_i to gate g_j then $i > j$. We denote the two edges entering gate g_i with e_{2i+n-2} and e_{2i+n-1}. Moreover, we write e_i $(0 \leq i \leq n-1)$ for an imaginary edge that leaves the output gate y_i and whose target gate is unspecified. Thus, the edges of the circuit are $e_0, \ldots, e_{2p+n-1}$. We now define the natural numbers $q_0, \ldots, q_{n-1}, r_1, \ldots r_m, s_1, \ldots, s_k$ with $k = 3p$:

- Let $I = \{j \mid e_j \text{ is an outgoing edge of the constant gate } c_1 \text{ or a nand-gate}\}$. For $0 \leq i \leq n-1$ we define the number q_i as

$$q_i = \sum_{j \in I \setminus \{i\}} 4^j.$$

  Recall that e_i is the unique outgoing edge of the output gate y_i.
- If e_j is the unique outgoing edge of the input gate x_i then we set $r_i = 4^j$. We can choose the reverse topological sorting of the nand-gates in such a way that $r_1 > r_2 > \cdots > r_m$ (we only have to ensure that the target gates $\overline{x}_1, \ldots, \overline{x}_m$ of the input gates appear in the order $\overline{x}_m, \ldots, \overline{x}_1$ in the reverse topological sorting of the nand-gates).
- To define the numbers $s_1, \ldots, s_k$ we first define for every nand-gate g_i three numbers t_{3i}, t_{3i-1} and t_{3i-2} as follows, where $I_i = \{j \mid e_j \text{ is an outgoing edge of gate } g_i\}$:

$$\begin{aligned} t_{3i} &= 4^{2i+n-1} + 4^{2i+n-2} + \sum_{j \in I_i} 4^j \\ t_{3i-1} &= 4^{2i+n-1} - 4^{2i+n-2} = 3 \cdot 4^{2i+n-2} \\ t_{3i-2} &= 4^{2i+n-2} \end{aligned}$$

  Then, the tuple $(s_1, \ldots, s_k)$ is $(t_{3p}, t_{3p-1}, t_{3p-2}, \ldots, t_3, t_2, t_1)$, which is indeed super-decreasing (see also [30]). In fact, we have $s_i - (s_{i+1} + \cdots + s_k) \geq 4^{n-1}$ for all $i \in [1..k]$. To see this, note that the sets $I_{i+1}, \ldots, I_k$ are pairwise disjoint. This implies that the $n-1$ low-order digits in the base-4 expansion of $s_{i+1} + \cdots + s_k$ are zero or one.

In order to understand this construction, one should think of the edges of the circuit carrying truth values. Recall that there are $2p+n$ edges in the circuit (including the imaginary outgoing edges of the output gates $y_0, \ldots, y_{n-1}$). A number in base-4 representation with $2p+n$ digits that are either 0 or 1 represents a truth assignment to the $2p+n$ edges, where a 1-digit represents the truth value 1 and a 0-digit represents the truth value 0. Consider an input string $\alpha = b_1 \cdots b_m \in \{0,1\}^m$ and consider an output gate y_i, $i \in [0..n-1]$. Then the number $N := 4^i + q_i + b_1 r_1 + \cdots + b_m r_m$ encodes the truth assignment for the circuit edges, where:

- all outgoing edges of the constant gate c_1 carry the truth value 1,
- all outgoing edges of the constant gate c_0 carry the truth value 0,
- the unique outgoing edge of an input gate x_i carries the truth value b_i,
- all outgoing edges of nand-gates carry the truth value 1.

We have to show that $C(\alpha)_i = 1$ if and only if there exists $\delta \in \{0,1\}^k$ such that $\delta \cdot \overline{s} = N - 4^i$. For this we apply the canonical algorithm for super-decreasing subsetsum with input $(N, s_1, \ldots, s_k)$. This algorithm initializes a counter A to N and then goes over the sequence $s_1, \ldots, s_k$ in that order. In the j-th step $(1 \leq j \leq k)$ we set A to $A - s_j$ if $A \geq s_j$. If $A < s_j$ then we do not modify A. After that we proceed with s_{j+1}. The point is that this process simulates the evaluation of the circuit on the input values $b_1, \ldots, b_m$. Thereby the nand-gates are evaluated in the topological order $g_p, g_{p-1}, \ldots, g_1$. Assume that g_j is the gate that we want to evaluate next. In the above algorithm for super-decreasing subsetsum the evaluation of g_j is simulated by the three numbers t_{3j}, t_{3j-1}, and t_{3j-2}. At the point where the algorithm checks whether t_{3j} can be subtracted from A, the base-4 digits at positions $2j+n, \ldots, 2p+n-1$ in the counter value A have been already set to zero. If the digits at the next two high-order positions $2j+n-1$ and $2j+n-2$ are still 1 (i.e., the input edges e_{2j+n-2} and e_{2j+n-1} for gate g_j carry the truth value 1), then we can subtract t_{3j} from A. Thereby we subtract all powers 4^{2j+n-1}, 4^{2j+n-2} and 4^h, where e_h is an outgoing edge for gate g_j. Since gate g_j evaluates to zero (both input edges carry 1), this subtraction correctly simulates the evaluation of gate g_j: all outgoing edges e_h of g_j (that were initially set to the truth value 1) are set to the truth value 0. On the other hand, if one of the two digits at positions $2j+n-1$ and $2j+n-2$ in A is 0 (which means that gate g_j evaluates to 1), then we cannot subtract t_{3j} from A. If both digits at positions $2j+n-1$ and $2j+n-2$ in A are 0, then also t_{3j-1} and t_{3j-2} cannot be subtracted. On the other hand, if exactly one of the two digits at positions $2j+n-1$ and $2j+n-2$ is 1, then with t_{3j-1} and t_{3j-2} we can set these two digits to 0 (thereby digits at positions $< 2j+n-2$ are not modified).

Assume now that y_j $(j \in [0..n-1])$ is the unique output gate that evaluates to 1, i.e., all output gates $y_{j'}$ with $j' \neq j$ evaluate to zero. Then after processing all weights $s_1, \ldots, s_k$ we have $A = 4^j$ (we will never subtract 4^j). We have shown that there exists $\delta \in \{0,1\}^k$ such that $\delta \cdot \overline{s} + 4^j = N$. Hence, if $i = j$ (i.e., $C(\alpha)_i = 1$) then $\delta \cdot \overline{s} = N - 4^i$. Now assume that $i \neq j$. In order to get a contradiction, assume that there is $\delta' \in \{0,1\}^k$ such that $\delta' \cdot \overline{s} + 4^i = N$. We have $\delta \neq \delta'$ and $\delta \cdot \overline{s} + 4^j = \delta' \cdot \overline{s} + 4^i$, i.e, $\delta \cdot \overline{s} - \delta' \cdot \overline{s} = 4^i - 4^j$. Since $i, j \in [0..n-1]$, we get $|\delta \cdot \overline{s} - \delta' \cdot \overline{s}| < 4^{n-1}$. But $s_i - (s_{i+1} + \cdots s_k) \geq 4^{n-1}$ for all $i \in [1..k]$ implies that $|\delta \cdot \overline{s} - \delta' \cdot \overline{s}| \geq 4^{n-1}$.

C.3 From super-decreasing subsetsum to straight-line programs

In [35] a super-decreasing sequence $\overline{t} = (t_1, \ldots, t_k)$ of natural numbers is encoded by the string $S(\overline{t}) \in \{0,1\}^*$ of length $\sum \overline{t} + 1$ such that for all $0 \leq p \leq \sum \overline{t}$:

$$S(\overline{t})[p] = \begin{cases} 1 & \text{if } p = \alpha \cdot \overline{t} \text{ for some } \alpha \in \{0,1\}^k, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Note that in the first case α is unique. Since $\overline{t}$ is a super-decreasing sequence, the number of 1's in the string $S(\overline{t})$ is 2^k. Also note that $S(\overline{t})$ starts and ends with 1. In [35] it was shown that from a super-decreasing sequence $\overline{t}$ of binary encoded numbers one can construct in LOGSPACE an SLP for the word $S(\overline{t})$.

C.4 Proof of Theorem 6

Let us fix a regular wreath product of the form $G \wr \mathbb{Z}$ for a finitely generated group G. Such groups are also known as generalized lamplighter groups (the lamplighter group arises for $G = \mathbb{Z}_2$). Throughout this section, we fix a set of standard generators Σ for G and let $\tau = 1$ be the generator for $\mathbb{Z}$. Then $\Sigma \cup \{\tau, \tau^{-1}\}$ is a standard generating set for the wreath product $G \wr \mathbb{Z}$. In $G \wr \mathbb{Z}$ the G-generator $a \in \Sigma$ represents the mapping $f_a \in G^{(\mathbb{Z})}$ with $f_a(0) = a$ and $f_a(z) = 1$ for $z \neq 0$. For a word $w \in (\Sigma \cup \{\tau, \tau^{-1}\})^*$ we define $\eta(w) := |w|_\tau - |w|_{\tau^{-1}}$. Thus, the element of $G \wr \mathbb{Z}$ represented by w is of the form $f\tau^{\eta(w)}$ for some $f \in G^{(\mathbb{Z})}$. Recall the definition of the left action of $\mathbb{Z}$ on $G^{(\mathbb{Z})}$ from Section 2 (where we take $H = Y = \mathbb{Z}$). For better readability, we write $c \circ f$ for ${}^c f$ ($c \in \mathbb{Z}$, $f \in G^{(\mathbb{Z})}$). Hence, we have $(c \circ f)(z) = f(z+c)$. If one thinks of f as a bi-infinite word over the alphabet G, then $c \circ f$ is the same word but shifted by $-c$.

The following intuition might be helpful: Consider a word $w \in (\Sigma \cup \{\tau, \tau^{-1}\})^*$. In $G \wr \mathbb{Z}$ we can simplify w to a word of the form $\tau^{z_0} a_1 \tau^{z_1} a_2 \cdots \tau^{z_{k-1}} a_k \tau^{z_k}$ (with $z_j \in \mathbb{Z}$, $a_j \in \Sigma$), which in $G \wr \mathbb{Z}$ can be rewritten as

$$\tau^{z_0} a_1 \tau^{z_1} a_2 \cdots \tau^{z_{k-1}} a_k \tau^{z_k} = \Big(\prod_{j=1}^{k} \tau^{z_0+\cdots+z_{j-1}} a_j \tau^{-(z_0+\cdots+z_{j-1})} \Big)\, \tau^{z_0+\cdots+z_k}.$$

Hence, the word w represents the group element

$$\Big(\prod_{j=1}^{k} (z_0 + \cdots + z_{j-1}) \circ f_{a_j} \Big)\, \tau^{z_0+\cdots+z_k}.$$

This gives the following intuition for evaluating $\tau^{z_0} a_1 \tau^{z_1} a_2 \cdots \tau^{z_{k-1}} a_k \tau^{z_k}$: In the beginning, every $\mathbb{Z}$-position carries the G-value 1. First, go to the $\mathbb{Z}$-position $-z_0$ and multiply the G-element at this position with a_1 (on the right), then go to the $\mathbb{Z}$-position $-z_0 - z_1$ and multiply the G-element at this position with a_2, and so on.

Proof of Theorem 6. The easy part is to show that the circuit value problem for $G \wr \mathbb{Z}$ belongs to $\forall\mathsf{LEAF}(\mathrm{WP}(G))$. In the following, we make use of the statements from Lemma 32. Let $\mathcal{G}$ be an SLP over the alphabet $\Sigma \cup \{\tau, \tau^{-1}\}$ and let $f\tau^{\eta(\mathrm{val}(\mathcal{G}))} \in G \wr \mathbb{Z}$ be the group element represented by $\mathrm{val}(\mathcal{G})$. By Lemma 32 we can compute $\eta(\mathrm{val}(\mathcal{G}))$ in polynomial time. If $\eta(\mathrm{val}(\mathcal{G})) \neq 0$ then the Turing-machine rejects by printing a non-trivial generator of G (here we need the assumption that G is non-trivial). So, let us assume that $\eta(\mathrm{val}(\mathcal{G})) = 0$. We can also compute in polynomial time two integers $b, c \in \mathbb{Z}$ such that $\mathrm{supp}(f) \subseteq [b..c]$. We can take for instance $b = -|\mathrm{val}(\mathcal{G})|$ and $c = |\mathrm{val}(\mathcal{G})|$. It suffices to check whether for all $x \in [b..c]$ we have $f(x) = 1$. For this, the Turing-machine branches universally to all binary encoded integers $x \in [b..c]$ (this yields the $\forall$-part in $\forall\mathsf{LEAF}(\mathrm{WP}(G))$). Consider a specific branch that leads to the integer $x \in [b..c]$. From x and the input SLP $\mathcal{G}$ the Turing-machine then produces a leaf string over the standard generating set Σ of G such that this leaf string represents the group element $f(x) \in G$. For this, the machine branches to all positions $p \in [0..|\mathrm{val}(\mathcal{G})|-1]$ (if $p < q < |\mathrm{val}(\mathcal{G})|$ then the branch for p is to the left of the branch for q). For a specific position p, the machine computes in polynomial time the symbol $a = \mathrm{val}(\mathcal{G})[p]$. If a is τ or τ^{-1} then the machine prints $1 \in \Sigma$. On the other hand, if $a \in \Sigma$ then the machine computes in polynomial time $d = \eta(\mathrm{val}(\mathcal{G})[:p])$. This is possible by first computing an SLP for the prefix $\mathrm{val}(\mathcal{G})[:p]$. If $d = -x$ then the machine prints the symbol a, otherwise the machine prints the trivial generator 1. It is easy to observe that the leaf string produced in this way represents the group element $f(x)$.

Let us now prove hardness for $\forall\mathsf{LEAF}(\mathrm{WP}(G/Z(G)))$ with respect to $\mathsf{LOGSPACE}$-reductions. By Lemma 29 it suffices to show that $\mathrm{CVP}(G \wr \mathbb{Z})$ is hard for the balanced class $\forall\mathsf{bLEAF}(\mathrm{WP}(G/Z(G)))$. Let $a_0, \ldots, a_{n-1}$ be an arbitrary enumeration of the standard generators in Σ. Fix a language $L \in \forall\mathsf{bLEAF}(\mathrm{WP}(G/Z(G)))$. From the definition of the class $\forall\mathsf{bLEAF}(\mathrm{WP}(G/Z(G)))$ it follows that there exist two polynomials p_1 and p_2 and a balanced polynomial time NTM M running in time $p_1 + p_2$ that outputs a symbol from Σ after termination and such that the following holds: Consider an input word z for M and let $m_1 = p_1(|z|)$, $m_2 = p_2(|z|)$, $m = m_1 + m_2$, and $T(z)$ be the corresponding computation tree of M on input z. It is a complete binary tree of height m. Its node set can be identified with the bit strings in $\{0,1\}^{\leq m}$, where $v0$ (resp., $v1$) is the left (resp., right) child of $v \in \{0,1\}^{<m}$. For every leaf $\alpha \in \{0,1\}^m$ let us denote with $\lambda(\alpha)$ the symbol from Σ that M prints when reaching the leaf α. Then we have: $z \in L$ if and only if for all $\beta \in \{0,1\}^{m_1}$ the string

$$\lambda_\beta := \prod_{\gamma \in \{0,1\}^{m_2}} \lambda(\beta\gamma) \tag{4}$$

represents a group element from the center $Z(G)$. Here (and in the following), the product in the right-hand side of (4) goes over all bit strings of length m_2 in lexicographic order. Our construction consists of five steps:

Step 1. Note that given a bit string $\alpha \in \{0,1\}^m$, we can compute in polynomial time the symbol $\lambda(\alpha) \in \Sigma$ by following the computation path specified by α. Using the classical Cook-Levin construction (see e.g. [2]), we can compute from the input z and $a \in \Sigma$ in $\mathsf{LOGSPACE}$ a Boolean circuit $C_{z,a}$ with m input gates $x_1, \ldots, x_m$ and a single output gate y_0 such that for all $\alpha \in \{0,1\}^m$: $C_{z,a}(\alpha)_0 = 1$ if and only if $\lambda(\alpha) = a$. By taking the disjoint union of these circuits and merging the input gates, we can build a single circuit C_z with m input gates $x_1, \ldots, x_m$ and $n = |\Sigma|$ output gates $y_0, \ldots, y_{n-1}$. For every $\alpha \in \{0,1\}^m$ and every $0 \leq i \leq n-1$ the following holds: $C_z(\alpha)_i = 1$ if and only if $\lambda(\alpha) = a_i$.

Step 2. Using the construction from Appendix C.2 we can compute from the circuit C_z in $\mathsf{LOGSPACE}$ numbers $q_0, \ldots, q_{n-1} \in \mathbb{N}$ and two super-decreasing sequences $\overline{r} = (r_1, \ldots, r_m)$ and $\overline{s} = (s_1, \ldots, s_k)$ with the following properties:

- The $r_1, \ldots, r_m$ are pairwise distinct powers of 4.
- For all $0 \leq i \leq n-1$ and all $\alpha \in \{0,1\}^m$ we have: $\lambda(\alpha) = a_i$ if and only if $C_z(\alpha)_i = 1$ if and only if there is $\delta \in \{0,1\}^k$ such that $\delta \cdot \overline{s} = q_i + \alpha \cdot \overline{r}$.

Note that for all $\alpha \in \{0,1\}^m$ there is a unique i such that $C_z(\alpha)_i = 1$. Hence, for all $\alpha \in \{0,1\}^m$ there is a unique i such that $q_i + \alpha \cdot \overline{r}$ is of the form $\delta \cdot \overline{s}$ for some $\delta \in \{0,1\}^k$. For this unique i we have $\lambda(\alpha) = a_i$.

We split the super-decreasing sequence $\overline{r} = (r_1, \ldots, r_m)$ into the two sequences $\overline{r}_1 = (r_1, \ldots, r_{m_1})$ and $\overline{r}_2 = (r_{m_1+1}, \ldots, r_m)$. For the following consideration, we need the following numbers:

$$\ell = \max\left\{ \textstyle\sum \overline{r}_1 + \max\{q_0, \ldots, q_{n-1}\} + 1,\ \sum \overline{s} - \sum \overline{r}_2 - \min\{q_0, \ldots, q_{n-1}\} + 1 \right\} \tag{5}$$

$$\pi = \ell + \textstyle\sum \overline{r}_2 \tag{6}$$

The binary encodings of these numbers can be computed in $\mathsf{LOGSPACE}$ (since iterated addition, max, and min can be computed in $\mathsf{LOGSPACE}$). The precise value of ℓ will be only relevant at the end of step 4.

Step 3. By the result from [35] (see Appendix C.3) we can construct in LOGSPACE from the three super-decreasing sequences $\overline{r}_1, \overline{r}_2$ and $\overline{s}$ three SLPs $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{H}$ over the alphabet $\{0,1\}$ such that $\mathrm{val}(\mathcal{G}_1) = S(\overline{r}_1)$, $\mathrm{val}(\mathcal{G}_2) = S(\overline{r}_2)$ and $\mathrm{val}(\mathcal{H}) = S(\overline{s})$ (see (3)). For all positions $p \geq 0$ (in the suitable range) we have:

$$\begin{aligned}
\mathrm{val}(\mathcal{G}_1)[p] = 1 &\iff \exists \beta \in \{0,1\}^{m_1} : p = \beta \cdot \overline{r}_1 \\
\mathrm{val}(\mathcal{G}_2)[p] = 1 &\iff \exists \gamma \in \{0,1\}^{m_2} : p = \gamma \cdot \overline{r}_2 \\
\mathrm{val}(\mathcal{H})[p] = 1 &\iff \exists \delta \in \{0,1\}^{k} : p = \delta \cdot \overline{s}
\end{aligned}$$

Note that $|\mathrm{val}(\mathcal{G}_1)| = \sum \overline{r}_1 + 1$, $|\mathrm{val}(\mathcal{G}_2)| = \sum \overline{r}_2 + 1$, and $|\mathrm{val}(\mathcal{H})| = \sum \overline{s} + 1$.

Step 4. We build in LOGSPACE for every $0 \leq i \leq n-1$ an SLP $\mathcal{H}_i$ from the SLP $\mathcal{H}$ by replacing in every right-hand side of $\mathcal{H}$ every occurrence of 0 by τ^{-1} and every occurrence of 1 by $a_i\tau^{-1}$. Let T_i be the start variable of $\mathcal{H}_i$, let S_1 be the start variable of $\mathcal{G}_1$, and let S_2 be the start variable of $\mathcal{G}_2$. We can assume that the variable sets of the SLPs $\mathcal{G}_1, \mathcal{G}_2, \mathcal{H}_0, \ldots, \mathcal{H}_{n-1}$ are pairwise disjoint. We next combine these SLPs into a single SLP $\mathcal{I}$. The variables of $\mathcal{I}$ are the variables of the SLPs $\mathcal{G}_1, \mathcal{G}_2, \mathcal{H}_0, \ldots, \mathcal{H}_{n-1}$ plus a fresh variable S which is the start variable of $\mathcal{I}$. The right-hand sides for the variables are defined below. In the right-hand sides we write powers τ^p for integers p whose binary encodings can be computed in LOGSPACE. Such powers can be produced by small subSLPs that can be constructed in LOGSPACE too.

- In all right-hand sides of $\mathcal{G}_1$ and $\mathcal{G}_2$ we replace all occurrences of the terminal symbol 0 by the $\mathbb{Z}$-generator τ.
- We replace every occurrence of the terminal symbol 1 in a right-hand side of $\mathcal{G}_1$ by $S_2\tau^{\ell}$, where ℓ is from (5).
- We replace every occurrence of the terminal symbol 1 in a right-hand side of $\mathcal{G}_2$ by $\sigma\tau$, where

$$\sigma = \tau^{q_0} T_0 \tau^{h-q_0} \tau^{q_1} T_1 \tau^{h-q_1} \cdots \tau^{q_{n-1}} T_{n-1} \tau^{h-q_{n-1}} \tag{7}$$

  and $h = \sum \overline{s} + 1$ is the length of the word $\mathrm{val}(\mathcal{H})$ (which is $-\eta(\mathrm{val}_{\mathcal{I}}(T_i))$ for every $i \in [0..n-1]$). Note that $\eta(\mathrm{val}_{\mathcal{I}}(\sigma)) = 0$.
- Finally, the right-hand side of the start variable S is $S_1\tau^{-d}$ where $d := \sum \overline{r}_1 + 1 + 2^{m_1} \cdot \pi$. (note that $d = \eta(\mathrm{val}_{\mathcal{I}}(S_1))$).

Before we explain this construction, let us first introduce some notations.

- Let $u := \mathrm{val}_{\mathcal{I}}(S_2)$. We have $\eta(u) = |\mathrm{val}(\mathcal{G}_2)|$. Hence, the group element represented by u can be written as $f_u\tau^{|\mathrm{val}(\mathcal{G}_2)|}$ for a mapping $f_u \in G^{(\mathbb{Z})}$.
- Let $v := \mathrm{val}_{\mathcal{I}}(\sigma)$, where σ is from (7). Note that $\eta(v) = 0$. Hence, the group element represented by v is a mapping $f_v \in G^{(\mathbb{Z})}$. Its support is a subset of the interval from position $-\max\{q_0, \ldots, q_{n-1}\}$ to position $\sum \overline{s} - \min\{q_0, \ldots, q_{n-1}\}$.
- For $\beta \in \{0,1\}^{m_1}$ let $\mathsf{bin}(\beta)$ be the number represented by β in binary notation (thus, $\mathsf{bin}(0^{m_1}) = 0$, $\mathsf{bin}(0^{m_1-1}1) = 1$, …, $\mathsf{bin}(1^{m_1}) = 2^{m_1} - 1$). Moreover, let

$$p_\beta := -\mathsf{bin}(\beta) \cdot \pi.$$

First, note that $\eta(\mathrm{val}(\mathcal{I})) = 0$. This is due to the factor τ^{-d} in the right-hand side of the start variable S of $\mathcal{I}$. Hence, the group element represented by $\mathrm{val}(\mathcal{I})$ is a mapping $f \in G^{(\mathbb{Z})}$. The crucial claim is the following:

▷ Claim. For every $\beta \in \{0,1\}^{m_1}$, $f(p_\beta)$ is the group element represented by the leaf string λ_β from (4).

Proof of the claim. In the following, we compute in the restricted direct product $G^{(\mathbb{Z})}$. Recall that the multiplication in this group is defined by the pointwise multiplication of mappings.

Since we replaced in $\mathcal{G}_1$ every 1 in a right-hand side by $S_2\tau^\ell$, which produces $u\tau^\ell$ in $\mathcal{I}$ (which evaluates to $f_u\tau^{\pi+1}$), the mapping f is a product (in the restricted direct product $G^{(\mathbb{Z})}$) of shifted copies of f_u. More precisely, for every $\beta' \in \{0,1\}^{m_1}$ we get the shifted copy

$$\big(\beta' \cdot \overline{r}_1 + \mathsf{bin}(\beta') \cdot \pi\big) \circ f_u \tag{8}$$

of f_u. The shift distance $\beta' \cdot \overline{r}_1 + \mathsf{bin}(\beta') \cdot \pi$ can be explained as follows: The 1 in $\mathrm{val}(\mathcal{G}_1)$ that corresponds to $\beta' \in \{0,1\}^{m_1}$ occurs at position $\beta' \cdot \overline{r}_1$ (the first position is 0) and to the left of this position we find $\mathsf{bin}(\beta')$ many 1's and $\beta' \cdot \overline{r}_1 - \mathsf{bin}(\beta')$ many 0's in $\mathrm{val}(\mathcal{G}_1)$. Moreover, every 0 in $\mathrm{val}(\mathcal{G}_1)$ was replaced by τ (shift by 1) and every 1 in $\mathrm{val}(\mathcal{G}_1)$ was replaced by $u\tau^\ell$ (shift by $\ell + |\mathrm{val}(\mathcal{G}_2)| = \pi + 1$). Hence, the total shift distance is indeed (8). Also note that if $\beta' \in \{0,1\}^{m_1}$ is lexicographically smaller than $\beta'' \in \{0,1\}^{m_1}$ then $\beta' \cdot \overline{r}_1 < \beta'' \cdot \overline{r}_1$. This implies that

$$f = \prod_{\beta' \in \{0,1\}^{m_1}} \big(\beta' \cdot \overline{r}_1 + \mathsf{bin}(\beta') \cdot \pi\big) \circ f_u = \prod_{\beta' \in \{0,1\}^{m_1}} \big(\beta' \cdot \overline{r}_1 - p_{\beta'}\big) \circ f_u.$$

Let us now compute the mapping f_u. Recall that we replaced in $\mathcal{G}_2$ every occurrence of 1 by $\sigma\tau$, where σ is from (7) and derives to v. The 1's in $\mathrm{val}(\mathcal{G}_2)$ occur at positions of the form $\gamma \cdot \overline{r}_2$ for $\gamma \in \{0,1\}^{m_2}$ and if $\gamma \in \{0,1\}^{m_2}$ is lexicographically smaller than $\gamma' \in \{0,1\}^{m_2}$ then $\gamma \cdot \overline{r}_2 < \gamma' \cdot \overline{r}_2$. We therefore get

$$f_u = \prod_{\gamma \in \{0,1\}^{m_2}} (\gamma \cdot \overline{r}_2) \circ f_v.$$

We obtain

$$\begin{aligned} f &= \prod_{\beta' \in \{0,1\}^{m_1}} \big(\beta' \cdot \overline{r}_1 - p_{\beta'}\big) \circ f_u \\ &= \prod_{\beta' \in \{0,1\}^{m_1}} \big(\beta' \cdot \overline{r}_1 - p_{\beta'}\big) \circ \prod_{\gamma \in \{0,1\}^{m_2}} (\gamma \cdot \overline{r}_2 \circ f_v) \\ &= \prod_{\beta' \in \{0,1\}^{m_1}} \prod_{\gamma \in \{0,1\}^{m_2}} \big(\beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 - p_{\beta'}\big) \circ f_v \end{aligned}$$

and hence

$$f(p_\beta) = \prod_{\beta' \in \{0,1\}^{m_1}} \prod_{\gamma \in \{0,1\}^{m_2}} f_v(p_\beta - p_{\beta'} + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2).$$

We claim that for all $\beta \neq \beta'$ and all $\gamma \in \{0,1\}^{m_2}$ we have

$$f_v(p_\beta - p_{\beta'} + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2) = 1. \tag{9}$$

Let us postpone the proof of this for a moment. From (9) we get

$$f(p_\beta) = \prod_{\gamma \in \{0,1\}^{m_2}} f_v(\beta \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2).$$

Consider a specific $\gamma \in \{0,1\}^{m_2}$ and let $\alpha = \beta\gamma$ and $p = \beta \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 = \alpha \cdot \overline{r}$. From the definition of $v = \mathrm{val}_{\mathcal{I}}(\sigma)$ it follows that for all $x \in \mathbb{Z}$, $f_v(x)$ is a product of those group generators a_i such that $x = -q_i + \delta \cdot \overline{s}$ for some $\delta \in \{0,1\}^k$. For the position p this means

that $q_i + \alpha \cdot \overline{r} = \delta \cdot \overline{s}$. By our previous remarks, there is a unique such $i \in [0..n-1]$ and for this i we have $\lambda(\alpha) = a_i$. Hence, we obtain $f_v(p) = \lambda(\alpha) = \lambda(\beta\gamma)$ and thus

$$f(p_\beta) = \prod_{\gamma \in \{0,1\}^{m_2}} \lambda(\beta\gamma) = \lambda_\beta.$$

It remains to show (9). To get this identity, we need the precise value of ℓ from (5) (so far, the value of ℓ was not relevant). Assume now that $\beta \neq \beta'$, which implies

$$|p_\beta - p_{\beta'}| \geq \pi = \ell + \sum \overline{r}_2.$$

Hence, we either have

$$\begin{aligned} p_\beta - p_{\beta'} + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 &\geq \ell + \sum \overline{r}_2 + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 \\ &\geq \ell + \sum \overline{r}_2 \\ &> \sum \overline{s} - \min\{q_0, \ldots, q_{n-1}\} \end{aligned}$$

or

$$\begin{aligned} p_\beta - p_{\beta'} + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 &\leq -\ell - \sum \overline{r}_2 + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 \\ &\leq -\ell + \sum \overline{r}_1 \\ &< -\max\{q_0, \ldots, q_{n-1}\}, \end{aligned}$$

where the strict inequalities follow from our choice of ℓ. Recall that the support of the mapping f_v is contained in $[-\max\{q_0, \ldots, q_{n-1}\}.. \sum \overline{s} - \min\{q_0, \ldots, q_{n-1}\}]$. This shows (9) and hence the claim. ◁

Step 5. By the above claim, we have $f(p_\beta) \in Z(G)$ for all $\beta \in \{0,1\}^{m_1}$ if and only if $\lambda_\beta \in Z(G)$ for all $\beta \in \{0,1\}^{m_1}$, which is equivalent to $z \in L$. The only remaining problem is that the word $\mathrm{val}(\mathcal{I})$ produces some "garbage" group elements $f(x)$ on positions x that are not of the form p_β. Note that for every $g \in G \setminus Z(G)$, there is a generator $a_i \in \Sigma$ such that the commutator $[g, a_i]$ is non-trivial. We now produce from $\mathcal{I}$ an SLP $\mathcal{I}^{-1}$ such that $\mathrm{val}(\mathcal{I}^{-1})$ represents the inverse element of $f \in G^{(\mathbb{Z})}$, which is the mapping g with $g(x) = f(x)^{-1}$ for all $x \in \mathbb{Z}$. To construct $\mathcal{I}^{-1}$, we have to reverse every right-hand side of $\mathcal{I}$ and replace every occurrence of a symbol $a_0, \ldots, a_{n-1}, \tau, \tau^{-1}$ by its inverse.

It is easy to compute in LOGSPACE for every $i \in [0..n-1]$ an SLP for the word

$$w_i := \left(a_i \tau^\pi\right)^{2^{m_1}} \tau^{-2^{m_1} \cdot \pi}.$$

Then the group element represented by w_i is the mapping $f_i \in G^{(\mathbb{Z})}$ whose support is the set of positions p_β for $\beta \in \{0,1\}^{m_1}$ and $f_i(p_\beta) = a_i$ for all $\beta \in \{0,1\}^{m_1}$. We can also compute in LOGSPACE an SLP for the word w_i^{-1}. We then built in LOGSPACE SLPs $\mathcal{J}_0, \ldots, \mathcal{J}_{n-1}$ such that $\mathrm{val}(\mathcal{J}_i) = \mathrm{val}(\mathcal{I}^{-1}) w_i^{-1} \mathrm{val}(\mathcal{I}) w_i$. Hence, the word $\mathrm{val}(\mathcal{J}_i)$ represents the group element $g_i \in G^{(\mathbb{Z})}$, where $g_i(x) = 1$ for all $x \in \mathbb{Z} \setminus \{p_\beta \mid \beta \in \{0,1\}^{m_1}\}$ and $g_i(p_\beta) = f(p_\beta)^{-1} a_i^{-1} f(p_\beta) a_i = [f(p_\beta), a_i]$.

Finally, we construct in LOGSPACE an SLP $\mathcal{J}$ such that

$$\mathrm{val}(\mathcal{J}) = \mathrm{val}(\mathcal{J}_0)\, \tau\, \mathrm{val}(\mathcal{J}_1)\, \tau\, \mathrm{val}(\mathcal{J}_2) \cdots \tau\, \mathrm{val}(\mathcal{J}_{n-1})\, \tau^{-n+1}.$$

We can assume that $n \leq \ell + \sum \overline{r}_2 = \pi$ (n is a constant and we can always make ℓ bigger). Then $\mathrm{val}(\mathcal{J})$ evaluates to the group element $g \in G^{(\mathbb{Z})}$ with $g(x) = 1$ for $x \in \mathbb{Z} \setminus \{p_\beta - i \mid \beta \in \{0,1\}^{m_1}, 0 \leq i \leq n-1\}$ and $g(p_\beta - i) = g_i(p_\beta) = [f(p_\beta), a_i]$ for $0 \leq i \leq n-1$. Hence, if $f(p_\beta) \in Z(G)$ for all $\beta \in \{0,1\}^{m_1}$ then $\mathrm{val}(\mathcal{J}) = 1$ in $G \wr \mathbb{Z}$. On the other hand, if there is a $\beta \in \{0,1\}^{m_1}$ such that $f(p_\beta) \in G \setminus Z(G)$ then there is an a_i such that $[f(p_\beta), a_i] \neq 1$. Hence $g(p_\beta - i) \neq 1$ and $\mathrm{val}(\mathcal{J}) \neq 1$ in $G \wr \mathbb{Z}$. This concludes the proof of Theorem 6. ◀