The Power Word Problem

Markus Lohrey
Universität Siegen, Germany

Armin Weiß
Universität Stuttgart, Germany

Abstract

In this work we introduce a new succinct variant of the word problem in a finitely generated group G, which we call the power word problem: the input word may contain powers p^x, where p is a finite word over generators of G and x is a binary encoded integer. The power word problem is a restriction of the compressed word problem, where the input word is represented by a straight-line program (i.e., an algebraic circuit over G). The main result of the paper states that the power word problem for a finitely generated free group F is AC^0-Turing-reducible to the word problem for F. Moreover, the following hardness result is shown: For a wreath product $G \wr \mathbb{Z}$, where G is either free of rank at least two or finite non-solvable, the power word problem is complete for coNP. This contrasts with the situation where G is abelian: then the power word problem is shown to be in TC^0.

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

Keywords and phrases word problem, compressed word problem, free groups

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.43

Related Version A full version [27] of the paper is available on arXiv https://arxiv.org/abs/1904.08343.

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

Acknowledgements We thank Laurent Bartholdi for pointing out the result [5, Theorem 6.6] on the bound of the order of elements in the Grigorchuk group, which allowed us to establish Theorem 10.

1 Introduction

Algorithmic problems in group theory have a long tradition, going back to the work of Dehn from 1911 [9]. One of the fundamental group theoretic decision problems introduced by Dehn is the *word problem* for a finitely generated group G (with a fixed finite generating set Σ): does a given word $w \in \Sigma^*$ evaluate to the group identity? Novikov [34] and Boone [8] independently proved in the 1950's the existence of finitely presented groups with undecidable word problem. On the positive side, in many important classes of groups the word problem is decidable, and in many cases also the computational complexity is quite low. Famous examples are finitely generated linear groups, where the word problem belongs to deterministic logarithmic space (L for short) [22] and hyperbolic groups where the word problem can be solved in linear time [17] as well as in LOGCFL [23].

In recent years, also compressed versions of group theoretical decision problems, where input words are represented in a succinct form, have attracted attention. One such succinct representation are so-called straight-line programs, which are context-free grammars that produce exactly one word. The size of such a grammar can be much smaller than the word it produces. For instance, the word a^n can be produced by a straight-line program of size $\mathcal{O}(\log n)$. For the *compressed word problem* for the group G the input consists of a straight-line program that produces a word w over the generators of G and it is asked whether w evaluates to the identity element of G. This problem is a reformulation of the

44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019).
Editors: Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen; Article No. 43; pp. 43:1–43:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

circuit evaluation problem for G. The compressed word problem naturally appears when one tries to solve the word problem in automorphism groups or semidirect products [25, Section 4.2]. For the following classes of groups, the compressed word problem is known to be solvable in polynomial time: finite groups (where the compressed word problem is either P-complete or in NC^2 [6]), finitely generated nilpotent groups [20] (where the complexity is even in NC^2), hyperbolic groups [18] (in particular, free groups), and virtually special groups (i.e, finite extensions of subgroups of right-angled Artin groups) [25]. The latter class covers for instance Coxeter groups, one-relator groups with torsion, fully residually free groups and fundamental groups of hyperbolic 3-manifolds. For finitely generated linear groups there is still a randomized polynomial time algorithm for the compressed word problem [26, 25]. Simple examples of groups where the compressed word problem is intractable are wreath products $G \wr \mathbb{Z}$ with G a non-abelian group: for every such group the compressed word problem is coNP-hard [25] (this includes for instance Thompson's group F); on the other hand, if, in addition, G is finite, then the (ordinary) word problem for $G \wr \mathbb{Z}$ is in NC^1 [37].

In this paper, we study a natural variant of the compressed word problem, called the *power word problem*. An input for the power word problem for the group G is a tuple $(p_1, x_1, p_2, x_2, \ldots, p_n, x_n)$ where every p_i is a word over the group generators and every x_i is a binary encoded integer (such a tuple is called a *power word*); the question is whether $p_1^{x_1} p_2^{x_2} \cdots p_n^{x_n}$ evaluates to the group identity of G.

From a power word $(p_1, x_1, p_2, x_2, \ldots, p_n, x_n)$ one can easily (e.g. by an AC^0-reduction) compute a straight-line program for the word $p_1^{x_1} p_2^{x_2} \cdots p_n^{x_n}$. In this sense, the power word problem is at most as difficult as the compressed word problem. On the other hand, both power words and straight-line programs achieve exponential compression in the best case; so the additional difficulty of the the compressed word problem does not come from a higher compression rate but rather because straight-line programs can generate more "complex" words.

Our main results for the power word problem are the following; in each case we compare our results with the corresponding results for the compressed word problem:

- The power word problem for every finitely generated nilpotent group is in $\mathsf{DLOGTIME}$-uniform TC^0 and hence has the same complexity as the word problem (or the problem of multiplying binary encoded integers). The proof is a straightforward adaption of a proof from [33]. There, the special case, where all words p_i in the input power word are single generators, was shown to be in $\mathsf{DLOGTIME}$-uniform TC^0. The compressed word problem for every finitely generated nilpotent group belongs to the class $\mathsf{DET} \subseteq \mathsf{NC}^2$ and is hard for the counting class $\mathsf{C_{=}L}$ in case of a torsion-free nilpotent group [20].
- The power word problem for a finitely generated group G is NC^1-many-one-reducible to the power word problem for any finite index subgroup of G. An analogous result holds for the compressed word problem as well [20].
- The power word problem for a finitely generated free group is AC^0-Turing-reducible to the word problem for F_2 (the free group of rank two) and therefore belongs to L. In contrast, it was shown in [24] that the compressed word problem for a finitely generated free group of rank at least two is P-complete.
- The power word problem for a wreath product $G \wr \mathbb{Z}$ with G finitely generated abelian belongs to $\mathsf{DLOGTIME}$-uniform TC^0. For the compressed word problem for $G \wr \mathbb{Z}$ with G finitely generated abelian only the existence of a randomized polynomial time algorithm for the complement is known [21].
- The power word problem for the wreath products $F_2 \wr \mathbb{Z}$ and every wreath product $G \wr \mathbb{Z}$, where G is finite and non-solvable, is coNP-complete. For these groups this sharpens the corresponding coNP-hardness result for the compressed word problem [25].

Table 1 Our results on the power word problem compared to previous results on the (compressed) word problem. Here WP stands for "word problem".

class of groups	PowerWP	CompressedWP	WP
nilpotent groups	TC^0	DET, $\mathsf{C_=L}$-hard [20]	TC^0 [35]
Grigorchuk group G	$\mathsf{L}^{a)}$	PSPACE	L [13]
non-abelian f.g. free	$\mathsf{L}^{b)}$	P-complete [24]	L [22]
$G \wr \mathbb{Z}$ for G f.g. abelian	TC^0	coRP [21]	TC^0 [30]
$G \wr \mathbb{Z}$ for G finite non-solvable	coNP-complete	PSPACE, coNP-hard [25]	NC^1 [37]
$F_2 \wr \mathbb{Z}$	coNP-complete	PSPACE, coNP-hard [25]	$\mathsf{L}^{b)}$ [37]
finite extension of a f.g. group H	NC^1-many-one-reducible to PowerWP(H) (resp. CompressedWP(H) [20], resp. WP(H) [37])		

a) AC^0-many-one-reducible to the word problem of G.
b) AC^0-Turing-reducible to the word problem of F_2.

- The power word problem for the Grigorchuk group is uAC^0-many-one-reducible to the word problem. The word problem for the Grigorchuk group is in L [13], which implies that the compressed word problem is in PSPACE. However, there is no non-trivial lower-bound known for the compressed word problem for the Grigorchuk group.

Table 1 summarizes the above results. Due to space constraints we present only short proof skteches for our main theorems; proofs of all lemmas can be found in the full version [27].

Related work. Implicitly, (variants of) the power word problem have been studied before. In the commutative setting, Ge [14] has shown that one can verify in polynomial time an identity $\alpha_1^{x_1}\alpha_2^{x_2}\cdots\alpha_n^{x_n} = 1$, where the α_i are elements of an algebraic number field and the x_i are binary encoded integers.

Another problem related to the power word problem is the knapsack problem [12, 28, 31] for a finitely generated group G (with generating set Σ): for a given sequence of words $w, w_1, \ldots, w_n \in \Sigma^*$, the question is whether there exist $x_1, \ldots, x_n \in \mathbb{N}$ such that $w = w_1^{x_1} \cdots w_n^{x_n}$ holds in G. For many groups G one can show that if such $x_1, \ldots, x_n \in \mathbb{N}$ exist, then there exist such numbers of size $2^{\mathrm{poly}(N)}$, where $N = |w| + |w_1| + \cdots + |w_n|$ is the input length. This holds for instance for right-angled Artin groups (also known as graph groups). In this case, one nondeterministically guesses the binary encodings of numbers $x_1, \ldots, x_n$ and then verifies, using an algorithm for the power word problem, whether $w_1^{x_1} \cdots w_n^{x_n} w^{-1} = 1$ holds. In this way, it was shown in [28] that for every right-angled Artin group the knapsack problem belongs to NP (using the fact that the compressed word problem and hence the power word problem for a right-angled Artin group belongs to P).

In [16], Gurevich and Schupp present a polynomial time algorithm for a compressed form of the subgroup membership problem for a free group F, where group elements are represented in the form $a_1^{x_1}a_2^{x_2}\cdots a_n^{x_n}$ with binary encoded integers x_i. The a_i must be standard generators of the free group F. This is the same input representation as in [33] and is more restrictive then our setting, where we allow powers of the form w^x for w an arbitrary word over the group generators (on the other hand, Gurevich and Schupp consider the subgroup membership problem, which is more general than the word problem).

2 Preliminaries

Words. An *alphabet* is a (finite or infinite) set Σ; an element $a \in \Sigma$ is called a *letter*. The free monoid over Σ is denoted by Σ^*, its elements are called *words*. The multiplication of the monoid is concatenation of words. The identity element is the empty word 1. The length of a word w is denoted by $|w|$. If w, p, x, q are words such that $w = pxq$, then we call x a *factor* of w, p a *prefix* of w, and q a *suffix* of w. We write $v \leq_{\mathrm{pref}} w$ (resp. $v <_{\mathrm{pref}} w$) if v is a (strict) prefix of w and $v \leq_{\mathrm{suff}} w$ (resp. $v <_{\mathrm{suff}} w$) if v is a (strict) suffix of w.

String rewriting systems. Let Σ be an alphabet and $S \subseteq \Sigma^* \times \Sigma^*$ be a set of pairs, called a *string rewriting system*. We write $\ell \to r$ if $(\ell, r) \in S$. The corresponding *rewriting relation* $\underset{S}{\Longrightarrow}$ over Σ^* is defined by: $u \underset{S}{\Longrightarrow} v$ if and only if there exist $\ell \to r \in S$ and words $s, t \in \Sigma^*$ such that $u = s\ell t$ and $v = srt$. We also say that u can be rewritten to v in one step. We write $u \overset{k}{\underset{S}{\Longrightarrow}} v$ if u can be rewritten to v in exactly k steps, i.e., if there are $u_0, \ldots, u_k$ with $u = u_0$, $v = u_k$ and $u_i \underset{S}{\Longrightarrow} u_{i+1}$ for $0 \leq i \leq k-1$. We denote the transitive closure of $\underset{S}{\Longrightarrow}$ by $\overset{+}{\underset{S}{\Longrightarrow}} = \bigcup_{k \geq 1} \overset{k}{\underset{S}{\Longrightarrow}}$ and the reflexive and transitive closure by $\overset{*}{\underset{S}{\Longrightarrow}} = \bigcup_{k \geq 0} \overset{k}{\underset{S}{\Longrightarrow}}$. Moreover $\overset{*}{\underset{S}{\Longleftrightarrow}}$ is the reflexive, transitive, and symmetric closure of $\underset{S}{\Longrightarrow}$; it is the smallest congruence containing S. The set of *irreducible word* with respect to S is $\mathrm{IRR}(S) = \{w \in \Sigma^* \mid \text{there is no } v \text{ with } w \underset{S}{\Longrightarrow} v\}$.

Free groups. Let X be a set and $X^{-1} = \left\{ a^{-1} \;\middle|\; a \in X \right\}$ be a disjoint copy of X. We extend the mapping $a \mapsto a^{-1}$ to an involution without fixed points on $\Sigma = X \cup X^{-1}$ by $(a^{-1})^{-1} = a$ and finally to an involution without fixed points on Σ^* by $(a_1 a_2 \cdots a_n)^{-1} = a_n^{-1} \cdots a_2^{-1} a_1^{-1}$. For an integer $z < 0$ and $w \in \Sigma^*$ we write w^z for $(w^{-1})^{-z}$. The string rewriting system $S = \left\{ aa^{-1} \to 1 \;\middle|\; a \in \Sigma \right\}$ is strongly confluent and terminating meaning that for every word $w \in \Sigma^*$ there exists a unique word $\mathrm{red}(w) \in \mathrm{IRR}(S)$ with $w \overset{*}{\underset{S}{\Longrightarrow}} \mathrm{red}(w)$ (for precise definitions see e.g. [7, 19]). Words from $\mathrm{IRR}(S)$ are called *freely reduced*. The system S defines the free group $F_X = \Sigma^*/S$ with basis X. More concretely, elements of F_X can be identified with freely reduced words, and the group product of $u, v \in \mathrm{IRR}(S)$ is defined by $\mathrm{red}(uv)$. With this definition $\mathrm{red} : \Sigma^* \to F_X$ becomes a monoid homomorphism that commutes with the involution $\cdot^{-1}$: $\mathrm{red}(w)^{-1} = \mathrm{red}(w^{-1})$ for all words $w \in \Sigma^*$. If $|X| = 2$, we write F_2 for F_X. It is known that for every countable set X, F_2 contains an isomorphic copy of F_X.

Finitely generated groups and the power word problem. A group G is called *finitely generated* if there exist a finite a finite set X and a surjective group homomorphism $h : F_X \to G$. In this situation, the set $\Sigma = X \cup X^{-1}$ is called a finite (symmetric) generating set for G. For words $u, v \in \Sigma^*$ we usually say that $u = v$ in G or $u =_G v$ in case $h(\mathrm{red}(u)) = h(\mathrm{red}(v))$. The *word problem* for the finitely generated group G, $\mathrm{WP}(G)$ for short, is defined as follows:
- input: a word $w \in \Sigma^*$.
- question: does $w =_G 1$ hold?

A *power word* (over Σ) is a tuple $(p_1, x_1, p_2, x_2, \ldots, p_n, x_n)$ where $p_1, \ldots, p_n \in \Sigma^*$ are words over the group generators (called the periods of the power word) and $x_1, \ldots, x_n \in \mathbb{Z}$ are integers that are given in binary notation. Such a power word represents the word $p_1^{x_1} p_2^{x_2} \cdots p_n^{x_n}$. Quite often, we will identify the power word $(p_1, x_1, p_2, x_2, \ldots, p_n, x_n)$ with the word $p_1^{x_1} p_2^{x_2} \cdots p_n^{x_n}$. Moreover, if $x_i = 1$, then we usually omit the exponent 1 in a power word. The *power word problem* for the finitely generated group G, PowerWP(G) for short, is defined as follows:

- input: a power word $(p_1, x_1, p_2, x_2, \ldots, p_n, x_n)$.
- question: does $p_1^{x_1} p_2^{x_2} \cdots p_n^{x_n} =_G 1$ hold?

Due to the binary encoded exponents, a power word can be seen as a succinct description of an ordinary word. Hence, a priori, the power word problem for a group G could be computationally more difficult than the word problem. We will see examples of groups G, where PowerWP(G) is indeed more difficult than WP(G) (under standard assumptions from complexity theory), as well as examples of groups G, where PowerWP(G) and WP(G) are equally difficult.

Wreath products. Let G and H be groups. Consider the direct sum $K = \bigoplus_{h \in H} G_h$, where G_h is a copy of G. We view K as the set $G^{(H)}$ of all mappings $f \colon H \to G$ such that $\mathrm{supp}(f) := \{h \in H \mid f(h) \neq 1\}$ is finite, together with pointwise multiplication as the group operation. The set $\mathrm{supp}(f) \subseteq H$ is called the *support* of f. The group H has a natural left action on $G^{(H)}$ given by $hf(a) = f(h^{-1}a)$, where $f \in G^{(H)}$ and $h, a \in H$. The corresponding semidirect product $G^{(H)} \rtimes H$ is the (restricted) *wreath product* $G \wr H$. In other words:

- Elements of $G \wr H$ are pairs (f, h), where $h \in H$ and $f \in G^{(H)}$.
- The multiplication in $G \wr H$ is defined as follows: Let $(f_1, h_1), (f_2, h_2) \in G \wr H$. Then $(f_1, h_1)(f_2, h_2) = (f, h_1 h_2)$, where $f(a) = f_1(a) f_2(h_1^{-1} a)$.

Complexity. We assume that the reader is familiar with the complexity classes P, NP, and coNP and many-one reductions; see e.g. [2] for details. We use circuit complexity for classes below deterministic logspace (L for short).

A language $L \subseteq \{0,1\}^*$ is AC^0-Turing-reducible to $K \subseteq \{0,1\}^*$ if there is a family of constant-depth, polynomial-size Boolean circuits with oracle gates for K deciding L. More precisely, $L \subseteq \{0,1\}^*$ belongs to $\mathsf{AC}^0(K)$ if there exists a family $(C_n)_{n \geq 0}$ of circuits which, apart from the input gates $x_1, \ldots, x_n$ are built up from *not*, *and*, *or*, and *oracle gates* for K (which output 1 if and only if their input is in K). All gates may have unbounded fan-in, but there is a polynomial bound on the number of gates and wires and a constant bound on the depth (length of a longest path from an input gate x_i to the output gate o). Finally, C_n accepts exactly the words from $L \cap \{0,1\}^n$, i.e., if each input gate x_i receives the input $a_i \in \{0,1\}$, then a distinguished output gate evaluates to 1 if and only if $a_1 a_2 \cdots a_n \in L$.

In the following, we only consider $\mathsf{DLOGTIME}$-uniform $\mathsf{AC}^0(K)$ for which we write $\mathsf{uAC}^0(K)$. $\mathsf{DLOGTIME}$-uniform means that there is a deterministic Turing machine which decides in time $\mathcal{O}(\log n)$ on input of two gate numbers (given in binary) and the string 1^n whether there is a wire between the two gates in the n-input circuit and also computes the type of a given gate. For more details on these definitions we refer to [36]. If the languages K and L in the above definition of $\mathsf{uAC}^0(K)$ are defined over a non-binary alphabet Σ, then one first has to fix a binary encoding of words over Σ.

The class uTC^0 is defined as uAC^0(Majority) where Majority is the problem to determine whether the input contains more 1s than 0s. The class NC^1 is the class of languages accepted by Boolean circuits of bounded fan-in and logarithmic depth. When talking about hardness for uTC^0 or NC^1 we use uAC^0-Turing reductions unless stated otherwise. As a consequence of Barrington's theorem [3], we have $\mathsf{NC}^1 = \mathsf{uAC}^0(\mathrm{WP}(A_5))$ where A_5 is the alternating group over 5 elements [36, Corollary 4.54]. Moreover, the word problem for any finite group G is in NC^1. Robinson proved that the word problem for the free group F_2 is NC^1-hard [35], i.e., $\mathsf{NC}^1 \subseteq \mathsf{uAC}^0(\mathrm{WP}(F_2))$.

3 Results

In this section we state our (and prove the easy) results on the power word problem. Outlines of the proofs of Theorems 2, 8 and 9 can be found in Sections 4 and 5, respectively.

▶ **Theorem 1.** *If G is a finitely generated nilpotent group, then* $\textsc{PowerWP}(G)$ *is in* uTC^0.

Proof. In [33], the so-called word problem with binary exponents was shown to be in uTC^0 for finitely generated nilpotent groups. We can apply the same techniques as in [33]: we compute Mal'cev normal forms of all p_i [33, Theorem 5], then use the power polynomials from [33, Lemma 2] to compute Mal'cev normal forms with binary exponents of all $p_i^{x_i}$. Finally, we compute the Mal'cev normal form of $p_1^{x_1} \cdots p_n^{x_n}$ again using [33]. ◀

▶ **Theorem 2.** *The power word problem for a finitely generated free group is* AC^0*-Turing-reducible to the word problem for the free group* F_2.

Notice that if the free group has rank one, then the power word problem is in uTC^0 because iterated addition is in uTC^0.

▶ Remark 3. If the input is of the form $(p_1, x_1, p_2, x_2, \ldots, p_n, x_n)$ where all p_i are freely reduced, then the reduction in Theorem 2 is a uTC^0-many-one reduction.

▶ Remark 4. One can consider variants of the power word problem, where the exponents are not given in binary representation but in even more compact forms. *Power circuits* as defined in [32] are such a representation that allow non-elementary compression for some integers. The proof of Theorem 2 involves iterated addition and comparison of exponents. For power circuits iterated addition is in uAC^0 (just putting the power circuits next to each other), but comparison (even for equality) is P-complete [38]. Hence, the variant of the power word problem, where exponents are encoded with power circuits is P-complete for free groups.

▶ Remark 5. The proof of Theorem 2 can be easily generalized to free products. However, in order to have a simpler presentation we only state and prove the result for free groups and postpone the free product case to a future full version.

It is easy to see that the power word problem for every finite group belongs to NC^1. The following result generalizes this fact:

▶ **Theorem 6.** *Let G be finitely generated and let $H \leq G$ have finite index. Then* $\textsc{PowerWP}(G)$ *is* NC^1*-many-one-reducible to* $\textsc{PowerWP}(H)$.

Proof sketch. W.l.o.g. we can assume that H is a finitely generated normal subgroup and R is a finite set of representatives of $Q := G/H$ with $1 \in R$. As a first step we replace in the input power word every $p_i^{x_i}$ by $h_i^{y_i} p_i^{z_i}$ where $x_i = y_i\,|Q| + z_i$, $0 \leq z_i < |Q|$ and h_i is a word over the generators of H with $p_i^{|Q|} =_G h_i$. Moreover, we write $p_i^{z_i}$ as a word without exponents. Using the conjugate collection process from [35, Theorem 5.2], the result can be rewritten in the form hr where h is a power word in the subgroup H and $r \in R$. ◀

As an immediate consequence of Theorem 2, Theorem 6 and the NC^1-hardness of the word problem for F_2 [35, Theorem 6.3] we obtain:

▶ **Corollary 7.** *The power word problem for every finitely generated virtually free group is* AC^0*-Turing-reducible to the word problem for the free group* F_2.

▶ **Theorem 8.** *For every finitely generated abelian group G,* $\textsc{PowerWP}(G \wr \mathbb{Z})$ *is in* uTC^0.

▶ **Theorem 9.** *Let G be either a finite non-solvable group or a finitely generated free group of rank at least two. Then* $\textsc{PowerWP}(G \wr \mathbb{Z})$ *is* coNP*-complete.*

▶ **Theorem 10.** *The power word problem for the Grigorchuk group (as defined in [15] and also known as* first Grigorchuk group*) is* uAC^0*-many-one-reducible to its word problem.*

Theorem 10 applies only if the generating set contains a neutral letter. Otherwise, the reduction is in uTC^0. It is well-know that the word problem for the Grigorchuk group is in L (see e.g. [13]). Thus, also the power word problem is in L.

Proof sketch of Theorem 10. By [5, Theorem 6.6], every element of length N in the Grigorchuk group has order at most $CN^{3/2}$ for some constant C. Since the order of every element is a power of two, we can reduce all exponents modulo the smallest power of two $\geq CN^{3/2}$ where N is the length of the longest period p_i. After that the words are short and can be written without exponents. ◀

4 Proof of Theorem 2

The proof of Theorem 2 consists of two main steps: first we do some preprocessing leading to a particularly nice instance of the power word problem. While this preprocessing is simple from a theoretical point of view, it is where the main part of the workload is performed during the execution of the algorithm. Then, in the second step, all exponents are reduced to polynomial size. After this shortening process, the power word problem can be solved by the ordinary word problem. The most difficult part is to prove correctness of the shortening process. For this, we introduce a rewriting system over an extended alphabet of words with exponents. We outline the proof in a sequence of lemmas which all follow rather easily from the previous ones and we give some small hints how to prove the lemmas.

Preprocessing. We use the notations from the paragraph on free groups in Section 2. In particular, recall that $S = \left\{ aa^{-1} \to 1 \;\middle|\; a \in \Sigma \right\}$. Fix an arbitrary order on the input alphabet Σ. This gives us the lexicographic order on Σ^*, which is denoted by $\preceq$. Let $\Omega \subseteq \mathrm{IRR}(S) \subseteq \Sigma^*$ denote the set of words w such that

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

Notice that Ω consists of Lyndon words [29, Chapter 5.1] with the stronger requirement of being freely reduced, cyclically reduced and also minimal among the conjugacy class of the inverse. The first aim is to rewrite the input power word in the form

$$w = s_0 p_1^{x_1} s_1 \cdots p_n^{x_n} s_n \qquad \text{with } p_i \in \Omega \text{ and } s_i \in \mathrm{IRR}(S). \tag{1}$$

The reason for this lies in the following crucial lemma which essentially says that, if a long factor of $p_i^{x_i}$ cancels with some $p_j^{x_j}$, then already $p_i = p_j$. Thus, only the same p_i can cancel implying that we can make the exponents of the different p_i independently smaller.

▶ **Lemma 11.** *Let $p, q \in \Omega$, $x, y \in \mathbb{Z}$ and let v be a factor of p^x and w a factor of q^y. If $vw \xRightarrow[S]{*} 1$ and $|v| = |w| \geq |p| + |q| - 1$, then $p = q$.*

Proof. Since p and q are cyclically reduced, v and w are freely reduced, i.e., $v = w^{-1}$ as words. Thus, v has two periods $|p|$ and $|q|$. Since v is long enough, by the theorem of Fine and Wilf [10] it has also a period of $\gcd(|p|, |q|)$. This means that also p and q have period $\gcd(|p|, |q|)$ (since cyclic permutations of p and q are factors of v). Assuming $\gcd(|p|, |q|) < |p|$, would mean that p is a proper power contradicting the fact that p is primitive. Hence, $|p| = |q|$. Since $|v| \geq |p| + |q| - 1 = 2|p| - 1$, p is a factor of v, which itself is a factor of q^{-y}. Thus, p is a cyclic permutation of q or of q^{-1}. By the last condition on Ω, this implies $p = q$. ◀

▶ **Lemma 12.** *The following is in* $\mathsf{uAC}^0(\mathrm{WP}(F_2))$*: given a power word v, compute a power word w of the form* (1) *such that $v =_{F_X} w$.*

The proof of this lemma is straightforward using [39, Proposition 20] in order to compute freely reduced words. We call these steps the *preprocessing steps*. Henceforth, we will assume that the inputs for the power word problem are given in the form (1).

The symbolic reduction system. We define the infinite alphabet $\Delta = \Delta' \cup \Delta''$ with $\Delta' = \Omega \times (\mathbb{Z} \setminus \{0\})$ and $\Delta'' = \mathrm{IRR}(S) \setminus \{1\}$. We write p^x for $(p, x) \in \Delta'$. A word over Δ can be read as a word over Σ in the natural way. Formally, we can define a canonical projection $\pi : \Delta^* \to \Sigma^*$ that maps a symbol $a \in \Delta$ to the corresponding word over Σ, but most of the times we will not write π explicitly.

Whenever there is the risk of confusion, we write $|v|_\Sigma$ to denote the length of $v \in \Delta^*$ read over Σ (i.e., $|v|_\Sigma = |\pi(v)|$) whereas $|v|_\Delta$ is the length over Δ. Moreover, we denote the number of occurrences of letters from Δ' in w with $|w|_{\Delta'}$. Finally, for a symbol $s \in \Delta''$ define $\lambda(s) = |s|_\Sigma$ and for $p^x \in \Delta'$ set $\lambda(p^x) = |p|_\Sigma$. For $u = a_1 \cdots a_m \in \Delta^*$ with $a_i \in \Delta$ for $1 \leq i \leq m$ we define $\lambda(u) = \sum_{i=1}^{m} \lambda(a_i)$. Thus, $\lambda(u)$ is the number of letters from Σ required to write down u ignoring the binary exponents.

A word w as in (1), which has been preprocessed as in the previous section, can be viewed as word over Δ with $w \in ((\Delta'' \cup \{1\})\Delta')^*(\Delta'' \cup \{1\})$, $|w|_{\Delta'} = n$ and $|w|_\Delta \leq 2n + 1$ (we only have $\leq$ because some s_i might be empty).

We define the infinite string rewriting system T over Δ^* by the following rewrite rules, where $p^x, p^y, q^y \in \Delta'$, $s, t \in \Delta''$, $r \in \Delta'' \cup \{1\}$, and $d, e \in \mathbb{Z}$. Here, p^0 is identified with the empty word. Note that the strings in the rewrite rules are over the alphabet Δ, whereas the strings in the if-conditions are over the alphabet Σ.

$$p^x p^y \to p^{x+y} \tag{2}$$

$$p^x q^y \to p^{x-d} r q^{y-e} \qquad \text{if } p \neq q, p^x q^y \xRightarrow[S]{+} p^{x-d} r q^{y-e} \in \mathrm{IRR}(S) \text{ for } r = p'q' \text{ with } p' <_{\mathrm{pref}} p^{\mathrm{sign}(x)} \text{ and } q' <_{\mathrm{suff}} q^{\mathrm{sign}(y)} \tag{3}$$

$$st \to r \qquad \text{if } st \xRightarrow[S]{+} r \in \mathrm{IRR}(S) \tag{4}$$

$$p^x s \to p^{x-d} r \qquad \text{if } p^x s \xRightarrow[S]{+} p^{x-d} r \in \mathrm{IRR}(S) \text{ for } r = p's' \text{ with } p' <_{\mathrm{pref}} p^{\mathrm{sign}(x)} \text{ and } s' <_{\mathrm{suff}} s \tag{5}$$

$$s p^x \to r p^{x-d} \qquad \text{if } s p^x \xRightarrow[S]{+} r p^{x-d} \in \mathrm{IRR}(S) \text{ for } r = s'p' \text{ with } s' <_{\mathrm{pref}} s \text{ and } p' <_{\mathrm{suff}} p^{\mathrm{sign}(x)} \tag{6}$$

▶ **Lemma 13.** *The following length bounds hold in the above rules:*
- *in rule* (3)*:* $0 < |r|_\Sigma \leq |p|_\Sigma + |q|_\Sigma$, $|d| \leq |q|_\Sigma$*, and* $|e| \leq |p|_\Sigma$
- *in rules* (5) *and* (6)*:* $|d| \leq |s|_\Sigma$.

The inequalities $|d| \le |q|_\Sigma$ and $|e| \le |p|_\Sigma$ follow from Lemma 11. The other inequalities are obvious. The next lemma is also straightforward from the definition.

▶ **Lemma 14.** *For $u \in \Delta^*$ we have $u =_{F_X} 1$ if and only if $u \xRightarrow[T]{*} 1$.*

▶ **Lemma 15.** *Let $u \in \Delta^*$. If $u \xRightarrow[T]{*} v$, then $u \xRightarrow[T]{\le k} v$ for $k = 2\,|u|_\Delta + 4\,|u|_{\Delta'} \le 6\,|u|_\Delta$.*

Proof sketch. The proof is based on the fact that at most $2|u|_{\Delta'} - 3$ applications of rules of the form (3) can occur. These are the only length increasing rules. All other rules either decrease the number of non-reduced two-letter factors of u (this can happen at most $|u|_\Delta - 1$ times) or decrease the length of u (this can happen at most $|u|_\Delta + 2|u|_{\Delta'} - 3$ times). ◀

Consider a word $u \in \Delta^*$ and $p \in \Omega$. Let $\Delta_p = \{p^x \mid x \in \mathbb{Z} \setminus \{0\}\}$. We can write u uniquely as $u = u_0 p^{y_1} u_1 \cdots p^{y_m} u_m$ with $u_i \in (\Delta \setminus \Delta_p)^*$. We define $\eta_p^i(u) = \sum_{j=1}^{i} y_j$ and $\eta_p(u) = \eta_p^m(u)$. By Lemma 13 we know that all rules of T change $\eta_p(\cdot)$ by at most $\lambda(u)$. We can use this observation in order to show the next lemma by induction on k.

▶ **Lemma 16.** *Let $u \xRightarrow[T]{k} v$. Then for all $v' \le_{pref} v$ with $v' \in \Delta^*$ there is some $u' \in \Delta^*$ with $u' \le_{pref} u$ and $|\eta_p(u') - \eta_p(v')| \le (k+1)^2\,\lambda(u)$.*

The shortened version of a word. Take a word $u \in \Delta^*$ and $p \in \Omega$ and write u as $u = u_0 p^{y_1} u_1 \cdots p^{y_m} u_m$ with $u_i \in (\Delta \setminus \Delta_p)^*$ (we are only interested in the case that p^x appears as a letter in u for some $x \in \mathbb{Z}$). Let $\mathcal{C}$ be a finite set of finite, non-empty, non-overlapping intervals of integers, i.e., we can write $\mathcal{C} = \{\, [\ell_j, r_j] \mid 1 \le j \le k \,\}$ for $k = |\mathcal{C}|$ and $\ell_j \le r_j$ for all j. We can assume that the intervals are ordered increasingly, i.e., we have $r_j < \ell_{j+1}$. We set $d_j = r_j - \ell_j + 1 > 0$. We say that u is *compatible* with $\mathcal{C}$ if $\eta_p^i(u) \notin [\ell_j, r_j]$ for any i, j. If w is compatible with $\mathcal{C}$, we define the *shortened version* $\mathcal{S}_\mathcal{C}(u)$ of u: for $i \in \{1, \dots, m\}$ we set

$$C_i = C_i(u) = \begin{cases} \{\, j \mid 1 \le j \le k, \eta_p^{i-1}(u) < \ell_j \le r_j < \eta_p^i(u) \,\} & \text{if } y_i > 0 \\ \{\, j \mid 1 \le j \le k, \eta_p^{i}(u) < \ell_j \le r_j < \eta_p^{i-1}(u) \,\} & \text{if } y_i < 0, \end{cases}$$

i.e., C_i collects all intervals between $\eta_p^{i-1}(u)$ and $\eta_p^i(u)$. Then $\mathcal{S}_\mathcal{C}(u)$ is defined by

$$\mathcal{S}_\mathcal{C}(u) = u_0 p^{z_1} u_1 \cdots p^{z_m} u_m \qquad \text{where}$$
$$z_i = y_i - \operatorname{sign}(y_i) \cdot \sum_{j \in C_i} d_j = \begin{cases} y_i - \sum_{j \in C_i} d_j & \text{if } y_i > 0, \\ y_i + \sum_{j \in C_i} d_j & \text{if } y_i < 0. \end{cases}$$

A straightforward computation yields the next lemma:

▶ **Lemma 17.** *For all i we have $z_i \ne 0$ and $\operatorname{sign}(z_i) = \operatorname{sign}(y_i)$. In particular, if $u \in \mathrm{IRR}(T)$, then also $\mathcal{S}_\mathcal{C}(u) \in \mathrm{IRR}(T)$.*

Furthermore, we define $\operatorname{dist}_p(u, \mathcal{C}) = \min\left\{ \left|\eta_p^i(u) - x\right| \;\middle|\; 0 \le i \le m, x \in [\ell, r] \in \mathcal{C} \right\}$. Note that $\operatorname{dist}_p(u, \mathcal{C}) > 0$ if and only if u is compatible with $\mathcal{C}$. Moreover, if $\operatorname{dist}_p(u, \mathcal{C}) = a$, $v = v_0 p^{z_1} v_1 \cdots p^{z_m} v_m$, and $\left|\eta_p^i(u) - \eta_p^i(v)\right| \le b$ for all $i \le m$, then $\operatorname{dist}_p(v, \mathcal{C}) \ge a - b$.

▶ **Lemma 18.** *If $\operatorname{dist}_p(u, \mathcal{C}) > (k+1)^2\,\lambda(u)$ and $u \xRightarrow[T]{k} v$, then $S_\mathcal{C}(u) \xRightarrow[T]{k} S_\mathcal{C}(v)$.*

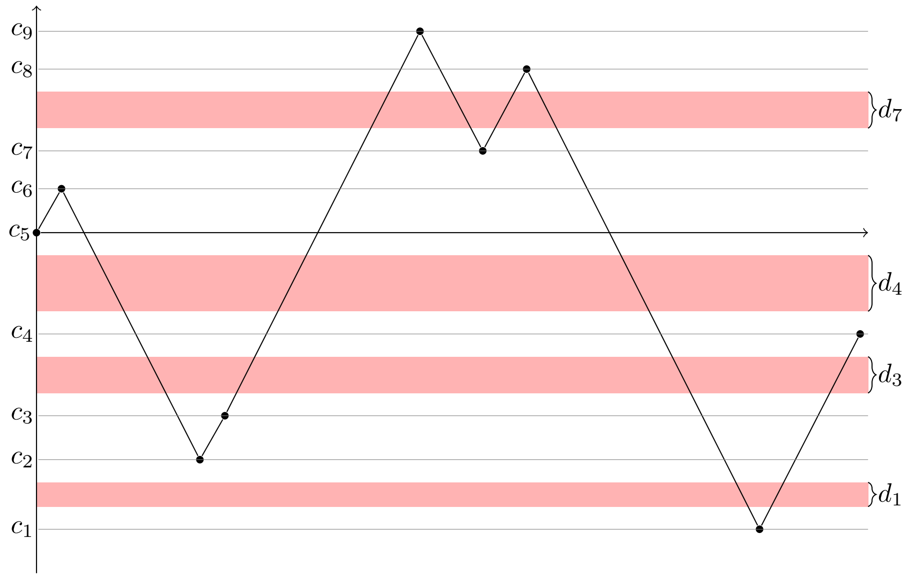

Figure 1 The red shaded parts represent the intervals from the set $\mathcal{C}^K_{u,p}$ in (7). The differences $c_3 - c_2$, $c_6 - c_5$, $c_7 - c_6$ and $c_9 - c_8$ are strictly smaller than $2K$.

Proof sketch. The first step for proving this lemma is to show that if $\mathrm{dist}_p(u, \mathcal{C}) > \lambda(u)$ and $u \underset{T}{\Longrightarrow} v$, then $S_{\mathcal{C}}(u) \underset{T}{\Longrightarrow} S_{\mathcal{C}}(v)$. To see this this, we distinguish between the rules applied: When applying one of the rules (3)–(6), we have $C_i(u) = C_i(v)$ for all i since the exponents are only changed slightly. Thus, the shortening process does the same on v as on u. When applying a rule (2), the exponents are added, which is compatible with the shortening process. Now we obtain the lemma by induction on k. In order to see that $\mathrm{dist}_p(u, \mathcal{C}) > \lambda(u)$ is satisfied in the inductive step, we use Lemma 16. ◀

We define a set of intervals which should be "cut out" from u as follows: We write $\{ c_1, \dots, c_l \} = \{ \eta^i_p(u) \mid 0 \le i \le m \}$ with $c_1 < \cdots < c_l$ and we set

$$\mathcal{C}^K_{u,p} = \{ [c_j + K, c_{j+1} - K] \mid 1 \le j \le l-1, c_{j+1} - c_j \ge 2K \} . \tag{7}$$

Notice that $\mathrm{dist}_p(u, \mathcal{C}^K_{u,p}) = K$ (given that $\mathcal{C}^K_{u,p} \ne \emptyset$). The situation is shown in Figure 1.

▶ **Proposition 19.** *Let $p \in \Omega$, $u = u_0 p^{y_1} u_1 \cdots p^{y_m} u_m \in \Delta^*$ with $u_i \in (\Delta \setminus \Delta_p)^*$, and $K = (6\,|u|_\Delta + 1)^2\,\lambda(u) + 1$. Then $u =_{F_X} 1$ if and only if $S_{\mathcal{C}}(u) =_{F_X} 1$ for $\mathcal{C} = \mathcal{C}^K_{u,p}$.*

Proof. By Lemma 14 we have $u =_{F_X} 1$ if and only if $u \overset{*}{\underset{T}{\Longrightarrow}} 1$. Let $k = 6\,|u|_\Delta$. By Lemma 15, for all $u \overset{*}{\underset{T}{\Longrightarrow}} v$ we have $u \overset{\le k}{\underset{T}{\Longrightarrow}} v$. By the choice of $\mathcal{C}$, we have $\mathrm{dist}_p(u, \mathcal{C}) > (k+1)^2\,\lambda(u)$. Hence, we can apply Lemma 18, which implies that $S_{\mathcal{C}}(u) \overset{*}{\underset{T}{\Longrightarrow}} S_{\mathcal{C}}(v)$ where v is a T-reduced (thus freely reduced) word for u. Clearly, if v is the empty word, then $S_{\mathcal{C}}(v)$ will be the empty word. On the other hand, if v is non-empty, then $S_{\mathcal{C}}(v)$ is non-empty and T-reduced by Lemma 17. Hence, we have $u =_{F_X} 1$ if and only if $S_{\mathcal{C}}(u) =_{F_X} 1$. ◀

▶ **Lemma 20.** *Let p, u, K, and $\mathcal{C}$ be as in Proposition 19 and $S_{\mathcal{C}}(u) = u_0 p^{z_1} u_1 \cdots p^{z_m} u_m$. Then $|z_i| \le m \cdot (2 \cdot (6\,|u|_\Delta + 1)^2 \cdot \lambda(u) + 1)$ for all $1 \le i \le m$.*

Proof of Theorem 2. We start with the preprocessing as described in Lemma 12 leading to a word $w = s_0 p_1^{x_1} s_1 \cdots p_n^{x_n} s_n$ with $p_i \in \Omega$ and $s_i \in \mathrm{IRR}(S)$ as in (1). After that we apply the shortening procedure for all $p \in \{ p_i \mid 1 \leq i \leq n \}$. This can be done in parallel for all p, as the outcome of the shortening only depends on the p-exponents. By Lemma 20 this leads to a word $\hat{w}$ of polynomial length. Finally, we can test whether $\hat{w} =_{F_X} 1$ using one oracle gate for $\mathrm{WP}(F_2)$ (recall that F_2 contains a copy of F_X). The computations for shortening only involve iterated addition (and comparisons of integers), which is in uTC^0 and, thus, can be solved in uAC^0 with oracle gates for $\mathrm{WP}(F_2)$. ◀

5 The power word problem in wreath products

The goal of this section is to prove Theorems 8 and 9. We first fix some notation. Let G be a finitely generated group with the finite symmetric generating set Σ. For $\mathbb{Z}$ we fix the generator a. Hence $\Sigma \cup \{a, a^{-1}\}$ is a symmetric generating set for the wreath product $G \wr \mathbb{Z}$. For a word $w = v_0 a^{e_1} v_1 \cdots a^{e_n} v_n$ with $e_i \in \{-1, 1\}$ and $v_i \in \Sigma^*$ let $\sigma(w) = e_1 + \cdots + e_n$. With $I(w)$ we denote the interval $[b, c] \subseteq \mathbb{Z}$, where b (resp., c) is the minimal (resp., maximal) integer of the form $e_1 + \cdots + e_i$ for $0 \leq i \leq n$. Note that if w represents $(f, d) \in G \wr \mathbb{Z}$, then $d = \sigma(w)$, $\mathrm{supp}(f) \subseteq I(w)$ and $0, d \in I(w)$.

Periodic words over groups. We recall a construction from [12]. With G^+ we denote the set of all tuples $(g_0, \dots, g_{q-1})$ over G of arbitrary length $q \geq 1$. With G^ω we denote the set of all mappings $f : \mathbb{N} \to G$. Elements of G^ω can be seen as infinite sequences (or words) over the set G. We define the binary operation $\otimes$ on G^ω by pointwise multiplication: $(f \otimes g)(n) = f(n)g(n)$. The identity element is the mapping id with $\mathrm{id}(n) = 1$ for all $n \in \mathbb{N}$. For $f_1, f_2, \dots, f_n \in G^\omega$ we write $\bigotimes_{i=1}^n f_i$ for $f_1 \otimes f_2 \otimes \cdots \otimes f_n$. If G is abelian, we write $\sum_{i=1}^n f_i$ for $\bigotimes_{i=1}^n f_i$. A function $f \in G^\omega$ is periodic with period $q \geq 1$ if $f(k) = f(k+q)$ for all $k \geq 0$. In this case, f can specified by the tuple $(f(0), \dots, f(q-1))$. Vice versa, a tuple $u = (g_0, \dots, g_{q-1}) \in G^+$ defines the periodic function $f_u \in G^\omega$ with $f_u(n \cdot q + r) = g_r$ for $n \geq 0$ and $0 \leq r < q$. One can view this mapping as the sequence u^ω obtained by taking infinitely many repetitions of u. Let G^ρ be the set of all periodic functions from G^ω. If f_1 is periodic with period q_1 and f_2 is periodic with period q_2, then $f_1 \otimes f_2$ is periodic with period $q_1 q_2$ (in fact, $\mathrm{lcm}(q_1, q_2)$). Hence, G^ρ forms a countable subgroup of G^ω. Note that G^ρ is not finitely generated: The subgroup generated by elements $f_i \in G^\rho$ with period q_i ($1 \leq i \leq n$) contains only functions with period $\mathrm{lcm}(q_1, \dots, q_n)$. For $n \geq 0$ we define the subgroup G^ρ_n of all $f \in G^\rho$ with $f(k) = 1$ for all $0 \leq k \leq n-1$. We consider the uniform membership problem for subgroups G^ρ_n, $\textsc{Membership}(G^\rho_*)$ for short:

- input: tuples $u_1, \dots, u_n \in G^+$ (elements of G are represented by finite words over Σ) and a binary encoded number m.
- question: does $\bigotimes_{i=1}^n f_{u_i}$ belong to G^ρ_m?

The following result was shown in [12]:

▶ **Theorem 21.** *For every finitely generated abelian group G, $\textsc{Membership}(G^\rho_*)$ is in uTC^0.*

▶ **Lemma 22.** *Let $w \in (\Sigma \cup \{a, a^{-1}\})^*$ with $\sigma(w) \neq 0$, $n \geq 1$, and $I(w^n) = [b, c]$. Moreover, let $s = c - b + 1$ be the size of the interval $I(w)$ and let $(g, n \cdot \sigma(w)) \in G \wr \mathbb{Z}$ be the group element represented by w^n. Then g is periodic on the interval $[b + s, c - s]$ with period $|\sigma(w)|$.*

▶ **Example 23.** Let us consider the wreath product $\mathbb{Z} \wr \mathbb{Z}$ and let the left copy of $\mathbb{Z}$ in the wreath product be generated by b. Consider the word $w = ba^{-1}babab^3ab^3ab^5a^{-1}b$ and let $n = 8$. We have $\sigma(w) = 2$ and $I(w) = [-1, 3]$. Moreover, w represents the group element

$(f,2)$ with $f(-1)=1$, $f(0)=2$, $f(1)=3$, $f(2)=4$, and $f(3)=5$. Let us now consider the word w^8. The following diagram shows how to obtain the corresponding element of $\mathbb{Z} \wr \mathbb{Z}$:

-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5														
		1	2	3	4	5												
				1	2	3	4	5										
						1	2	3	4	5								
								1	2	3	4	5						
										1	2	3	4	5				
												1	2	3	4	5		
														1	2	3	4	5
1	2	4	6	9	6	9	6	9	6	9	6	9	6	9	6	8	4	5

We have $I(w^8) = [-1, 17]$ and $\sigma(w^8) = 8\sigma(w) = 16$. If $(g, 16)$ is the group element represented by w^8, then the function g is periodic on the interval $[2, 14]$ (which includes the interval $[-1+s, 17-s]$, where $s = |I(w)| = 5$) with period 2.

Proofs of Theorem 8 and 9. A conjunctive truth-table reduction is a Turing reduction where the output is the conjunction over the outputs of all oracle gates.

▶ **Proposition 24.** *For every finitely generated group G, $\textsc{PowerWP}(G \wr \mathbb{Z})$ is conjunctive truth-table uTC^0-reducible to $\textsc{Membership}(G_*^\rho)$ and $\textsc{PowerWP}(G)$.*

Proof sketch. Let $w = u_1^{x_1} u_2^{x_2} \cdots u_k^{x_k}$ be the input power word and let $(f, d) \in G \wr \mathbb{Z}$ be the element represented by w. We can check in uTC^0 whether $d = 0$. The difficult part is to check whether f is the zero-mapping. For this we compute an interval I (of exponential size) that contains the support of f. We then partition I into two sets C and $I \setminus C$. The set C has polynomial size and we can check whether f is the zero-mapping on C using polynomially many oracle calls to $\textsc{PowerWP}(G)$. The complement $I \setminus C$ can be written as a union of polynomially many intervals. The crucial property of C is that on each of these intervals f can be written as a sum of periodic sequences; for this we use Lemma 22. Using oracle calls to $\textsc{Membership}(G_*^\rho)$ allows us to check whether f is the zero mapping on $I \setminus C$. ◀

Since for a finitely generated abelian group G, one can solve $\textsc{PowerWP}(G)$ in uTC^0, Theorem 8 is a consequence of Proposition 24 and Theorem 21.

We split the proof of Theorem 9 into three propositions: one for the upper bound and two for the lower bounds. It is straightforward to show that if the word problem for the finitely generated group G belongs to coNP, then also $\textsc{Membership}(G_*^\rho)$ belongs to coNP. Since coNP is closed under conjunctive truth-table uTC^0-reducibility, Proposition 24 yields:

▶ **Proposition 25.** *Let G be a finitely generated group such that $\textsc{PowerWP}(G)$ belongs to coNP. Then also $\textsc{PowerWP}(G \wr \mathbb{Z})$ belongs to coNP.*

▶ **Proposition 26.** *If G is a finite, non-solvable group, $\textsc{PowerWP}(G \wr \mathbb{Z})$ is coNP-hard.*

Proof sketch. Barrington [4] proved the following result: Let C be a fan-in two boolean circuit of depth d with n input gates $x_1, \ldots, x_n$. From C one can compute a sequence of triples (a so-called G-program) $P_C = (k_1, g_1, h_1)(k_2, g_2, h_2) \cdots (k_\ell, g_\ell, h_\ell) \in ([1,n] \times G \times G)^*$ of length $\ell \leq (4|G|)^d$ such that for every input valuation $v : \{x_1, \ldots, x_n\} \to \{0,1\}$ the following two statements are equivalent:

(a) C evaluates to 0 under the input valuation v.

(b) $c_1 c_2 \cdots c_\ell = 1$ in G, where $c_i = g_i$ if $v(x_{k_i}) = 0$ and $c_i = h_i$ if $v(x_{k_i}) = 1$.

This G-program is constructed as a sequence of iterated commutators, based on the observation that $[g, h] = 1$ if and only if $g = 1$ or $h = 1$ (given some reasonable assumptions on g and h). Every formula C in conjunctive normal form can be written as a circuit of depth $\mathcal{O}(\log |C|)$. Hence the G-program P_C has length polynomial in $|C|$. From [4] it is easy to see that on input of the formula C, the G-program P_C can be computed in logspace.

Let $P_C = (k_1, g_1, h_1) \cdots (k_\ell, g_\ell, h_\ell)$ and $x_1, \ldots, x_n$ be the variables in C. We compute in logspace the n first primes $p_1, \ldots, p_n$ and $M = \prod_{i=1}^{n} p_i$ (the latter in binary notation). Let a denote the generator of $\mathbb{Z}$ in the wreath product $G \wr \mathbb{Z}$. We now compute for every $1 \leq i \leq \ell$ the power word $w_i = (h_i(ag_i)^{p_{k_i}-1}a)^{M/p_{k_i}} a^{-M}$ and set $w_C = w_1 w_2 \cdots w_\ell$. The group element of $G \wr \mathbb{Z}$ represented by w_C is of the form $(f, 0)$.

We claim that $w_C = 1$ in $G \wr \mathbb{Z}$ if and only if C is unsatisfiable: For a number $z \in [0, M-1]$ we define the valuation $v_z : \{x_1, \ldots, x_n\} \to \{0, 1\}$ by $v_z(x_i) = 1$ if $z \equiv 0 \mod p_i$ and $v_z(x_i) = 0$ otherwise. By the Chinese remainder theorem, for every valuation $v : \{x_1, \ldots, x_n\} \to \{0, 1\}$ there exists $z \in [0, M-1]$ with $v = v_z$. Based on the above statements (a) and (b), the final step of the proof checks that $f(z) = 1$ if and only if C evaluates to 0 under v_z. ◀

▶ **Proposition 27.** *Let F be a finitely generated free group of rank at least two. Then* PowerWP$(F \wr \mathbb{Z})$ *is* coNP*-hard.*

The proof is almost the same as for Proposition 26. The difference is that we mimic Robinson's proof that the word problem for F_2 is NC^1-hard [35] instead of Barrington's result.

6 Further Research

We conjecture that the method of Section 4 can be generalized to right-angled Artin groups (RAAGs – also known as graph groups) and hyperbolic groups, and hence that the power word problem for a RAAG (resp., hyperbolic group) G is AC^0-Turing-reducible to the word problem for G. One may also try to prove transfer results for the power word problem with respect to group theoretical constructions, e.g., graph products, HNN extensions and amalgamated products over finite subgroups. For finitely generated linear groups, the power word problem leads to the problem of computing matrix powers with binary encoded exponents. The complexity of this problem is open; variants of this problem have been studied in [1, 11].

Another open question is what happens if we allow nested exponents. We conjecture that in the free group for any nesting depth bounded by a constant the problem is still in $\mathsf{uAC}^0(\mathrm{WP}(F_2))$. However, for unbounded nesting depth it is not clear what happens: we only know that it is in P since it is a special case of the compressed word problem; but it still could be in $\mathsf{uAC}^0(\mathrm{WP}(F_2))$ or it could be P-complete or somewhere in between.

References

1 Eric Allender, Nikhil Balaji, and Samir Datta. Low-Depth Uniform Threshold Circuits and the Bit-Complexity of Straight Line Programs. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science, MFCS 2014, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 13–24. Springer-Verlag, 2014. doi:10.1007/978-3-662-44465-8_2.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

3 David A. Mix Barrington. Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 1–5. ACM, 1986. doi:10.1145/12130.12131.

4 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.

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

6 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.

7 Ron Book and Friedrich Otto. *String-Rewriting Systems*. Springer-Verlag, 1993.

8 William W. Boone. The word problem. *Annals of Mathematics*, 70(2):207–265, 1959.

9 Max Dehn. Ueber unendliche diskontinuierliche Gruppen. *Mathematische Annalen*, 71:116–144, 1911.

10 Nathan J. Fine and Herbert S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16:109–114, 1965.

11 Esther Galby, Joël Ouaknine, and James Worrell. On Matrix Powering in Low Dimensions. In *Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015*, volume 30 of *LIPIcs*, pages 329–340. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.329.

12 Moses Ganardi, Daniel König, Markus Lohrey, and Georg Zetzsche. Knapsack Problems for Wreath Products. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science, STACS 2018*, volume 96 of *LIPIcs*, pages 32:1–32:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018. URL: http://www.dagstuhl.de/dagpub/978-3-95977-062-0.

13 Max Garzon and Yechezkel Zalcstein. The complexity of Grigorchuk groups with application to cryptography. *Theoretical Computer Science*, 88(1):83–98, 1991.

14 Guoqiang Ge. Testing Equalities of Multiplicative Representations in Polynomial Time (Extended Abstract). In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science, FOCS 1993*, pages 422–426. IEEE Computer Society, 1993.

15 Rostislaw I. Grigorchuk. Burnside's problem on periodic groups. *Functional Analysis and Its Applications*, 14:41–43, 1980.

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

17 Derek Holt. Word-hyperbolic groups have real-time word problem. *International Journal of Algebra and Computation*, 10:221–227, 200.

18 Derek Holt, Markus Lohrey, and Saul Schleimer. Compressed Decision Problems in Hyperbolic Groups. In *Proceedings of the 36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019*, volume 126 of *LIPIcs*, pages 37:1–37:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.STACS.2019.37.

19 Matthias Jantzen. *Confluent String Rewriting*, volume 14 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

20 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.

21 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.

22 Richard J. Lipton and Yechezkel Zalcstein. Word Problems Solvable in Logspace. *Journal of the ACM*, 24:522–526, 1977.

23 Markus Lohrey. Decidability and complexity in automatic monoids. *International Journal of Foundations of Computer Science*, 16(4):707–722, 2005.

24 Markus Lohrey. Word Problems and Membership Problems on Compressed Words. *SIAM Journal on Computing*, 35(5):1210–1240, 2006. doi:10.1137/S0097539704445950.

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

26 Markus Lohrey and Saul Schleimer. Efficient computation in groups via compression. In *Proceedings of the 2nd International Symposium on Computer Science in Russia, CSR 2007*, volume 4649 of *Lecture Notes in Computer Science*, pages 249–258. Springer-Verlag, 2007.

27 Markus Lohrey and Armin Weiß. The power word problem. *CoRR*, abs/1904.08343, 2019. URL: `https://arxiv.org/abs/1904.08343`.

28 Markus Lohrey and Georg Zetzsche. Knapsack in Graph Groups. *Theory of Computing Systems*, 62(1):192–246, 2018. `doi:10.1007/s00224-017-9808-3`.

29 M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and Its Applications*. Addison-Wesley, 1983. Reprinted by *Cambridge University Press*, 1997.

30 Alexei Miasnikov, Svetla Vassileva, and Armin Weiß. The Conjugacy Problem in Free Solvable Groups and Wreath Products of Abelian Groups is in TC^0. *Theory of Computing Systems*, 63(4):809–832, 2018. `doi:10.1007/s00224-018-9849-2`.

31 Alexei Myasnikov, Andrey Nikolaev, and Alexander Ushakov. Knapsack Problems in Groups. *Mathematics of Computation*, 84(292):987–1016, 2015.

32 Alexei Myasnikov, Alexander Ushakov, and Won Dong-Wook. Power Circuits, exponential Algebra, and Time Complexity. *International Journal of Algebra and Computation*, 22(6):3–53, 2012.

33 Alexei Myasnikov and Armin Weiß. TC^0 circuits for algorithmic problems in nilpotent groups. In *Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017*, volume 83 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPIcs.MFCS.2017.23`.

34 Pyotr S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov*, pages 1–143, 1955. In Russian.

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

36 Heribert Vollmer. *Introduction to Circuit Complexity*. Springer-Verlag, 1999.

37 Stephan Waack. The parallel complexity of some constructions in combinatorial group theory. *Journal of Information Processing and Cybernetics*, 26(5-6):265–281, 1990.

38 Armin Weiß. *On the Complexity of Conjugacy in Amalgamated Products and HNN Extensions*. Dissertation, Institut für Formale Methoden der Informatik, Universität Stuttgart, 2015.

39 Armin Weiß. A Logspace Solution to the Word and Conjugacy problem of Generalized Baumslag-Solitar Groups. In *Algebra and Computer Science*, volume 677 of *Contemporary Mathematics*, pages 185–212. American Mathematical Society, 2016.