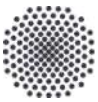


University of Stuttgart

Institute of Formal Methods in Computer Science

Universitätsstraße 38
70569 Stuttgart

Master Thesis

The Power Word Problem in Graph Groups

Florian Stober

Study progam:	Informatik
Examiner:	Prof. Dr. Volker Diekert
Advisor:	Dr. Armin Weiß
start date	02.01.2021
end date:	02.07.2021

Abstract

This thesis studies the complexity of the power word problem in graph groups. The power word problem is a variant of the word problem, where the input is a power word. A power word is a compact representation of a word. It may contain powers p^x, where p is a finite word and x is a binary encoded integer. A graph group, also known as right-angled Artin group or partially commutative group is a free group augmented with commutation relations. We show that the power word problem in graph groups can be decided in polynomial time, and more precisely it is AC^0-Turing-reducible to the word problem of the free group with two generators F_2. Being a generalization of graph groups, we also look into the power word problem in graph products. The power word problem in a fixed graph product is AC^0-Turing-reducible to the word problem of the free group F_2 and the power word problem of the base groups. Furthermore, we look into the uniform power word problem in a graph product, where the dependence graph and the base groups are part of the input. Given a class of finitely generated groups C, the uniform power word problem in a graph product is $\mathrm{C}_=\mathrm{L}$-Turing-reducible to the word problem in the free group F_2 and the uniform power word problem in C. Finally, we show that as a consequence of our results on the power word problem the uniform knapsack problem in graph groups is NP-complete.

Kurzfassung

In dieser Arbeit untersuchen wir die Komplexität des Power-Wort Problems in Graph Gruppen. Das Power-Wort Problem (engl. power word problem) ist eine Variante des Wort Problems (engl. word problem), bei der die Eingabe als Power-Wort (engl. power word) gegeben ist. Ein Power-Wort ist eine kompakte Darstellung eines Wortes. Dieses kann Potenzen p^x enthalten, wobei p ein Wort und x eine binär kodierte ganze Zahl ist. Eine Graph Gruppe (engl. graph group), auch bekannt als Rechtwinklige Artin-Gruppe oder Freie Partiell Kommutative Gruppe, ist eine Freie Gruppe mit Kommutationsrelationen. Wir zeigen, dass das Power-Wort Problem in Graph Gruppen in polynomieller Zeit gelöst werden kann. Noch genauer zeigen wir, dass es AC^0-Turing-reduzierbar auf das Wort Problem der Freien Gruppe mit zwei Erzeugenden F_2 ist. Zusätzlich werden Graph Produkte (engl. graph products) betrachtet, dabei handelt es sich um eine Verallgemeinerung von Graph Gruppen. Im Fall von Graph Produkten zeigen wir, dass das Power-Wort Problem AC^0-Turing-reduzierbar auf das Wort Problem der Freien Gruppe F_2 und das Power-Wort Problem der zugrundeliegenden Gruppen ist. Weiterhin betrachten wir das uniforme Power-Wort Problem in Graph Produkten. Bei diesem sind der Abhängigkeitsgraph und die zugrundeliegenden Gruppen Teil der Eingabe. Wir zeigen, dass für eine Klasse von endlich erzeugten Gruppen C das uniforme Power-Wort Problem in einem Graph Produkt $\mathrm{C}_=\mathrm{L}$-Turing-reduzierbar auf das Wort Problem der Freien Gruppe F_2 und das uniforme Power-Wort Problem in C ist. Aus unseren Ergebnissen schließen wir, dass das uniforme Rucksack Problem (engl. uniform knapsack problem) in Graph Gruppen NP-vollständig ist.

Contents

1 Introduction

In the field of algorithmic group theory, the word problem is an important area of study. More than a century ago, in 1911, Max Dehn [Deh11] recognized the importance of the word problem. Given a word in the generators of a finitely generated group, the word problem is to decide whether that word is equal to the identity in the group. An important discovery in the field has been the existence of finitely presented groups with undecidable word problem [Boo58; Nov55]. However, there are many groups with solvable word problem. For these groups, the algorithmic complexity of the word problem is studied.

In this thesis we study a variant of the word problem, called the power word problem, where the input is represented in a more compact form. The input is given as two lists $[p_1, \ldots, p_n]$ and $[x_1, \ldots, x_n]$, where p_i is a word in the generators of the group and x_i is a binary encoded integer for each i. Then, the problem is to decide whether $p_1^{x_1} \ldots p_n^{x_n}$ is equal to the identity in the group. The power word problem has been proposed by Markus Lohrey and Armin Weiß [LW19]. One of their results is that the power word problem for a finitely generated free group is AC^0-Turing reducible to the word problem in the free group with two generators F_2.

A graph group, also known as right-angled Artin group or partially commutative group is a free group augmented with commutation relations. Often, the notion of a dependence graph is used to describe which generators commute. The nodes of the dependence graph are the generators of the group. Two generators commute if they are not connected in the dependence graph.

A concept closely related to graph groups are partially commutative monoids, also referred to as trace monoids. A partially commutative monoid is a free monoid augmented with a commutation relation. Similar to graph groups, a dependence graph is used to describe the commutation relation. Elements of a partially commutative monoid are often represented as a directed acyclic graph.

Graph products are a generalization of graph groups. The concept has been introduced by Elisabeth R. Green in 1990 [Gre90]. Given a set of groups, a graph product is a free product of those groups augmented with commutation relations. As with graph groups, we use a dependence graph. Each node of the dependence graph corresponds to a group. Generators of two different groups commute, if the nodes corresponding to those groups are not connected in the dependence graph.

There are other approaches to representing the input to the word problem in a compact form. One of them is the compressed word problem introduced by Markus Lohrey in 2004 [Loh04]. The input is given as a straight-line program, that is a context free grammar that produces a single word. The compressed word problem of a graph group can be solved in polynomial time [HLM12].

The knapsack problem is a classical optimization problem. Informally, it can be described as follows. There are n different items, each with a weight and a value. The goal is to choose how many of each item to take, without exceeding the weight limit, while maximizing the value. When formulated as a decision problem, the question is whether a specific value can be achieved, while

staying within the weight limit. Karp has shown in 1972, that the decision variant of the knapsack problem is NP-complete [Kar72]. In 2015, Myasnikov et al. have formulated the knapsack problem for arbitrary groups [MNU15].

The knapsack problem for fixed graph groups has been studied by M. Lohrey and G. Zetzsche. They have shown that for certain graph groups the knapsack problem is NP-complete [LZ16].

1.1 Outline

In Chapter 2 we define the notation used throughout this thesis, we give formal definitions for partially commutative monoids and graph groups, and we introduce some basic tools for working with graph groups. Following that, Chapter 3 introduces different variants of the word problem discussed in this thesis, and we give an introduction to circuit complexity and counting complexity classes. In Chapter 4 we formally define graph products. Then, we present various known results, that will be used in this thesis. We conclude the chapter, by defining a cyclic normal form for graph products, and discussing how to encode the input for the power word problem.

Chapter 5 states conditions under which two elements of a graph group (respectively graph product) are conjugate. The results presented in this chapter are important tools to solving the power word problem in the next two chapters. In Chapter 6 and Chapter 7, we present a solution to the power word problem in graph groups and graph products. Each chapter begins by describing the preprocessing steps required for the algorithm. Then a symbolic rewriting system in defined which is used to prove the correctness of the following shortening process. There is some redundancy between the two chapters. Parts of the proof are identical or very similar. The chapter on graph groups may seem superfluous as the results presented there are implied by our results on graph products. However, solving the power word problem in graph products is more involved as there are more details to take care of. We hope that including the proof for the simpler case of graph groups, will aid in understanding the proof for graph products.

Finally, in Chapter 8, we discuss the consequences of our results to the uniform knapsack problem in graph groups. Concluding the thesis, we summarize our results in Chapter 9.

2 Preliminaries

The free monoid over Σ is the set $M(\Sigma) = \Sigma^*$ together with the concatenation operation. An element of the free monoid is called a *word*[1]. The identity is the empty word which is denoted by 1. The set Σ is called an *alphabet*. The size of the alphabet is $\sigma = |\Sigma|$. An element of Σ is referred to as a *character*, a *letter* or a *symbol*.

Let $x = uwv$ for some $x, u, w, v \in \Sigma^*$. We say u is a *prefix*, w is a *factor* and v is a *suffix* of x. We call u a *proper prefix* if $u \neq x$. Similarly, w is a *proper factor* if $w \neq x$ and v is a *proper suffix* if $v \neq x$.

Let G be a group. We write 1_G to identify the identity in G. If the group is obvious from the context, we may omit the index G. Let $S \subseteq G$. We say that S is a generating set, if every group element can be expressed as a combination (w. r. t. the group operation) of elements of S and their inverses. We say that G is *finitely generated* (f.g.), if there is a finite generating set S.

Let X be an arbitrary set. Let $X^{-1} = \{x^{-1} \mid x \in X\}$ be a disjoint copy of X which acts as a set of symbolic inverses. Let $\Sigma = X \cup X^{-1}$. We have an involution on Σ by $x \mapsto x^{-1}$ which is extended to an involution on Σ^* by $(x_1 \cdots x_n)^{-1} \mapsto x_n^{-1} \cdots x_1^{-1}$. The free group with generating set X is defined as $F(X) = M(\Sigma)/\{aa^{-1} = 1 \mid a \in \Sigma\}$. By F_2 we denote the free group with two generators. A word $w \in \Sigma^*$ is called *reduced*, if for all $v \in \Sigma^*$ with $w =_G v$ it holds that $|w| \leq |v|$. This is equivalent to saying that there are no factors aa^{-1} in w, where $a \in \Sigma$.

Let G be a finitely generated group with generating set X. We express elements of G as words over $\Sigma = X \cup X^{-1}$. We write $w_1 =_G w_2$ to denote that two words $w_1, w_2 \in \Sigma^*$ represent the same group element. In contrast, if we write $w_1 = w_2$, then w_1 and w_2 are equal as words. Note that $w_1 = w_2$ implies $w_1 =_G w_2$, but the implication does not hold the other way.

A subgroup H of a group G is denoted by $H \leq G$. Let G and H be groups. Then $G \times H$ denotes the direct product of those groups, $G * H$ denotes the free product of those groups, $G \rtimes H$ denotes the semi-direct product. We write $G \simeq H$ if there is an isomorphism from G to H.

[1] Also known as *string* in the literature.

2.1 Rewriting Systems

A relation $S \subseteq \Sigma^* \times \Sigma^*$, where Σ is an alphabet, is called a rewriting system. We use the notation $l \to r$ to denote that $(l, r) \in S$. Based on the set S, the rewriting relation $\underset{S}{\Longrightarrow}$ is defined as

$$u \underset{S}{\Longrightarrow} v \quad \Longleftrightarrow \quad \exists l, r, p, q \in \Sigma^* : \; u = plq, v = prq, l \to r.$$

In addition we define the following:

- $\overset{+}{\underset{S}{\Longrightarrow}}$ is the transitive closure of $\underset{S}{\Longrightarrow}$.
- $\overset{*}{\underset{S}{\Longrightarrow}}$ is the reflexive, transitive closure of $\underset{S}{\Longrightarrow}$.
- We write $u \overset{k}{\underset{S}{\Longrightarrow}} v$ to denote that u can be rewritten to v in exactly k steps.
- We write $u \overset{\leq k}{\underset{S}{\Longrightarrow}} v$ to denote that u can be rewritten to v using at most k steps.

We say that a word $w \in \Sigma^*$ is *irreducible* w. r. t. S, if there is no $v \in \Sigma^*$ with $w \underset{S}{\Longrightarrow} v$. The set of irreducible words is defined as $\mathrm{IRR}(S) = \{w \in \Sigma^* \mid w \text{ is irreducible}\}$.

2.2 Partially Commutative Monoids

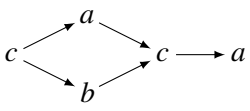
Let $M(\Sigma) = \Sigma^*$ be the free monoid over the set of generators Σ. Let $I \subseteq \Sigma \times \Sigma$ be symmetric and irreflexive. The relation I describes which generators commute. Hence, I is called the commutation relation. It is also known as independence relation. The relation $D = (\Sigma \times \Sigma) \setminus I$ is called dependence relation. It is often visualized as dependence graph.

We obtain a partially commutative monoid[2], by amending a commutation relation. It is defined as

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

Elements of a partially commutative monoid can be visually represented as a directed acyclic graph. Each distinguished letter in the word is a node in the graph. Let $w = u_1 \dots u_n$. There is an edge $u_i \to u_j$, if $i < j$ and $(u_i, u_j) \in D$. Often, transitive edges are omitted for better visibility.

Example 2.1 Let $\Sigma = \{a, b, c\}$. Let I be the symmetric closure of $\{(a, b)\}$. Let $w = cbaca$. Then w is represented by the following graph.



□

[2] Also known as trace monoid in literature.

The partially commutative monoid M is said to be *connected* if the dependence graph D is connected. A word $v \in M$ is said to be connected if the induced subgraph of D, consisting only of the letters occurring in v, is connected.

We extend the definitions of the terms prefix, suffix and factor to partially commutative monoids. Let $x =_M uwv$ for some $x, u, w, v \in \Sigma^*$. We say u is a *prefix*, w is a *factor* and v is a *suffix* of x.

Example 2.2 Let $\Sigma = \{a, b, c\}$. Let I be the symmetric closure of $\{(a, b), (b, c)\}$. Let $w = cbacabca$. Now c is a prefix of w, b is a prefix of w, but a is not. An example for a factor is bbc as w can be written as $w =_M caca\, bbc\, a$. □

In this thesis we frequently work with a power of some word. The following lemma describes the shape of a prefix, suffix or factor of such a power.

Lemma 2.3 *Given an element $p \in M$, where p is connected, and an integer $k \in \mathbb{N}$, the following is true for the shape of prefixes, suffixes and factors of p^k.*

(i) A prefix w of p^k can be written as $w = p^x w_s \cdots w_1$ where $x \in \mathbb{N}$, $s < \sigma$ and w_i is a proper prefix of p for each i.

(ii) A suffix w of p^k can be written as $w = w_1 \cdots w_t p^x$ where $x \in \mathbb{N}$, $t < \sigma$ and w_i is a proper suffix of p for each i.

(iii) Given a factor w of p^k at least one of the following is true.

- *w is a factor of p.*
- *We can write $w = u_1 \cdots u_t p^x v_s \cdots v_1$ where $x \in \mathbb{N}$, $s, t < \sigma$, u_i is a proper suffix of p for $1 \leq i \leq s$ and v_i is a proper prefix of p for $1 \leq i \leq t$.* □

Proof We look at three statements.

(i) Let w be a prefix of p^k. We can rewrite w by grouping the letters belonging to the same instance of p in p^k.

$$w = \tilde{w}_1 \tilde{w}_2 \ldots \tilde{w}_r$$

$\tilde{w}_r$ contains at least one letter. $\tilde{w}_{i-1}$ contains all instances of all letters in p that do not commute with some letter in $\tilde{w}_i$. It follows that $\tilde{w}_{r-j}$ contains all instances of all letters that are reachable from a letter in $\tilde{w}_r$ in the dependence graph restricted to letters in p with at most j steps. For $j \geq \sigma - 1$ we obtain that $\tilde{w}_{r-j}$ contains all instances of all letters in p, thus $\tilde{w}_{r-j} = p$. Hence, there is an $s \leq \sigma - 1$, and the lemma follows with $w_i = \tilde{w}_{r-i}$ and $x = r - s$.

(ii) Let w be a suffix of p^k. The proof is symmetric to (i).

(iii) Let w be a factor of p^k. We distinguish two cases.

- w is a factor of p. The lemma follows immediately.
- w is not a factor of p. There are u, v, r, s with $w = uv$ and $k = r + s$ such that u is a suffix of p^r and v is a prefix of p^s. The lemma follows from the results in (i) and (ii). ■

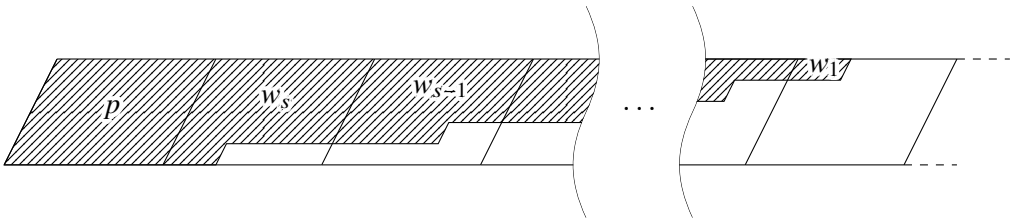


Figure 2.1: Prefix of some p^x.

Figure 2.1 illustrates case (i) of Lemma 2.3. The lemma not only applies to elements of partially commutative monoids, but also to freely reduced elements of graph groups and graph products, which is how it will be used most often in this thesis. The relation between graph groups, graph products and partially commutative monoids will be discussed further in the subsequent sections.

2.3 Graph Groups

Graph groups are defined similarly to partially commutative monoids. Again we have the symmetric commutation relation $I \subseteq \Sigma \times \Sigma$. We require that $(a, b) \in I$ if and only if $(a^{-1}, b) \in I$. The set $G = F(X)/\{ab = ba \mid (a, b) \in I\}$ is a graph group, also known as free partially commutative group or right-angled Artin group. The name graph group is due to the commutation relation being commonly visualized as an undirected graph.

Similar to free groups, we call a word $w \in \Sigma^*$ *freely reduced*, if for all $v \in \Sigma^*$ with $w =_G v$ it holds that $|w| \leq |v|$.

Graph groups and partially commutative monoids are tightly connected. The following lemma shows, how equality in a graph group and the corresponding partially commutative monoid are related.

Lemma 2.4 *Let $G = F(X)/\{ab = ba \mid (a, b) \in I\}$ and let the corresponding monoid be $M = M(\Sigma)/\{ab = ba \mid (a, b) \in I\}$. Given $u, v \in \Sigma^*$ the following holds.*

(i) $u =_M v \implies u =_G v$.

(ii) If u and v are freely reduced, then $u =_M v \iff u =_G v$. □

Proof We look at two statements.

(i) It follows directly from the definition of M and G, that $u =_M v \implies u =_G v$.

(ii) We only need to show that for freely reduced u and v, we have $u =_G v \implies u =_M v$. As u and v are freely reduced, they do not contain factors of the form aa^{-1}. Therefore, they are equal up to the commutation of letters, which is an operation that is allowed in M, and thus $u =_M v$. ■

2.4 Projection to Free Monoids

A graph group can be projected to a product of free monoids [Wra88]. Let $A_i \subseteq \Sigma$ be a family of sets fulfilling the following properties.

1. $\forall_i : a^{-1} \in A_i$ if and only if $a \in A_i$.
2. $(a, b) \in D$ if and only if $\exists_j : a, b \in A_j$.

The family A_i could consist of all sets $\{a, a^{-1}, b, b^{-1}\}$ of non-commuting elements, i. e., where $(a, b) \in D$ [Dub85]. An alternative is to use the maximal cliques in the dependence graph [Dub86].

Let $\pi_i : \Sigma^* \to A_i^*$ be the projection to the free monoid A_i^*, defined by $\pi_i(a) = a$ for $a \in A_i$, $\pi_i(a) = 1$ otherwise. We define a projection to a direct product of free monoids, $\Pi : \Sigma^* \to A_1^* \times \cdots \times A_k^*, w \mapsto (\pi_1(w), \ldots, \pi_k(w))$.

Using Lemma 2.4, we adapt two lemmata for partially commutative monoids presented in [Dub86] to hold for graph groups, by requiring that all elements are freely reduced.

Lemma 2.5 *[Dub86, Proposition 1.2] For freely reduced $u, v \in G$ we have $u =_G v$ if and only if $\Pi(u) = \Pi(v)$.* □

Lemma 2.6 *[Dub86, Proposition 1.7] Let $w \in \Sigma^*$ be freely reduced in G. There are $u \in \Sigma^*$ and $t > 1$ with $w =_G u^t$ if and only if there is a family F with $\Pi(w) = F^t$.* □

3 Algorithmic Complexity and Problems

In this chapter we formally define the decision problems discussed in this thesis. Then, we introduce circuit complexity and counting complexity classes.

Definition 3.1 (word problem) Let G be a group with a presentation over the alphabet Σ. The word problem $\mathrm{WP}(G)$ is to decide for $w \in \Sigma^*$ whether $w =_G 1$. □

Definition 3.2 (power word problem) Let G be a group with a presentation over the alphabet Σ. The input consists of a list of words $w_1, \ldots, w_n \in \Sigma^*$ and a list of binary encoded integers $x_1, \ldots, x_n \in \mathbb{Z}$. We interpret the input as $w = w_1^{x_1} \ldots w_n^{x_n}$. The power word problem $\mathrm{PowWP}(G)$ is to decide whether $w =_G 1$. □

Definition 3.3 (generalized word problem) Let G be a group with a presentation over the alphabet Σ. Let $H \leq G$. The generalized word problem $\mathrm{GWP}(G, H)$ is to decide for $w \in \Sigma^*$ whether $w \in H$.□

3.1 Circuit Complexity

The idea behind circuit complexity classes is to characterize the complexity of a function f by the size or depth of a boolean circuit that computes f. Circuit complexity classes can be seen as language classes or as function classes. The boolean circuit consists of *input* and *output* gates as well as the usual *and*, *or* and *not* gates. The gates of the circuit with their connections form a directed acyclic graph.

A single circuit has a fixed input size. Thus, we consider a family of circuits $C = (C_i)_{i \in \mathbb{N}}$. We say that C computes a function f, if on the input of w, C_n produces the output $f(w)$, where n is the length of the binary encoding of w.

In this thesis we use the following circuit complexity classes:

- NC^i: The class of functions computed by a family of boolean circuits of depth $O(\log^i(n))$, where the fan-in of each gate is bounded by a constant.
- AC^i: The class of functions computed by a family of boolean circuits of depth $O(\log^i(n))$. There is no bound on the fan-in of an individual gate.
- TC^i: The class of functions computed by a family of boolean circuits of depth $O(\log^i(n))$, with the addition of *majority* gates. Again, there is no bound on the fan-in of an individual gate.

Note that the above definition does not impose any restrictions on the computability and computational complexity of the mapping $n \mapsto C_n$. If we add such restrictions to a circuit complexity class, then we call it *uniform*. We only consider the DLOGTIME-uniform variant of the circuit complexity classes. A family of circuits C is DLOGTIME-uniform, if the following conditions are fulfilled.

- The type of a distinguished gate can be decided in DLOGTIME.
- Whether two gates are connected, can be decided in DLOGTIME.

To emphasize that we refer to the DLOGTIME-uniform variant of the circuit complexity class, we add the prefix u, e. g., we write uAC^0 to denote DLOGTIME-uniform AC^0.

Throughout this thesis we use Turing reductions to characterize the computational complexity of different decision problems. In the context of circuit complexity classes, Turing reductions are realized by allowing for oracle gates in the boolean circuit. An oracle gate can compute an arbitrary function or decide an arbitrary language. A problem A is AC^0-Turing reducible to a problem B, if A can be decided by a family of boolean circuits in AC^0, that may also contain oracle gates which decide B. We write $A \in \mathsf{uAC}^0(B)$ to denote that A is uAC^0-Turing reducible to B. We may use oracle gates from a set of languages C. In that case we write $A \in \mathsf{uAC}^0(C)$ to indicate that A can be decided in uAC^0 with oracle gates for the problems in C.

3.2 Counting Complexity Classes

Counting complexity classes are built on the idea of counting the number of accepting and rejecting paths of a Turing machine. For a non-deterministic Turing machine M, let accept_M denote the number of accepting paths and let reject_M denote the number of rejecting paths. We define $\text{gap}_M = \text{accept}_M - \text{reject}_M$.

In this thesis we make use of two counting complexity classes. The first is GapL, which contains all functions that can be computed as the difference of accepting and rejecting paths of a logarithmic space-bounded Turing machine. The second is $\mathrm{C}_=\mathrm{L}$, which contains a language L, if there is a function f in GapL, such that f evaluates to zero for exactly the words contained in L. These classes are formally defined as follows.

$$\mathsf{GapL} = \left\{\text{gap}_M \mid M \text{ is a non-deterministic, logarithmic space-bounded Turing machine}\right\}$$
$$\mathrm{C}_=\mathrm{L} = \{L \mid \text{There is } f \in \mathsf{GapL} \text{ with } \forall_{w \in \Sigma^*} : w \in L \iff f(w) = 0\}$$

4 Graph Products

Graph products are a generalization of graph groups. The concept has been introduced by Elisabeth R. Green in 1990 [Gre90]. Just like a graph group is a free group augmented with commutation relations, a graph product is a free product extended with commutation relations. Any graph group can be written as a graph product of single generator free groups. It follows that the results presented in this chapter can also be applied to graph groups.

We begin with a formal definition of graph products. Then we have a look at geodesics, which are the graph product equivalent to a freely reduced representative. Ee have a look at normal forms, introducing a cyclic normal form for graph products. Finally, we discuss how the input to the power word problem can be encoded in a way suitable to AC^0-Turing reductions.

Definition 4.1 Given a set $\mathcal{L}$, let $(G_\alpha)_{\alpha\in\mathcal{L}}$ be a family of groups. Let $I \subseteq \mathcal{L}\times\mathcal{L}$ be an irreflexive, symmetric relation. We call I the independence relation. The alphabet of the graph product is

$$\Gamma = \bigcup_{\alpha\in\mathcal{L}} (G_\alpha \setminus \{1\}).$$

For $w \in \Gamma$ we define $\mathrm{alph}(w) = \alpha$ where $w \in G_\alpha$. We extend the independence relation over $\Gamma\times\Gamma$ by $I = \{(u,v) \mid (\mathrm{alph}(u), \mathrm{alph}(v)) \in I\}$. The graph product is the group

$$\mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}}) = \langle \Gamma \mid uv = [uv] \text{ for } u,v \in G_\alpha; uv = vu \text{ for } (u,v) \in I\rangle \qquad \square$$

The equivalent to a freely reduced word in a graph product is a geodesic. A geodesic representative of a group element is a representative of minimum length. We use the terms geodesic, freely reduced and irreducible synonymously.

Let $w = w_1 \dots w_n$ be freely reduced. Then there are no $i \le j < k$ such that $\mathrm{alph}(w_i) = \mathrm{alph}(w_k)$ and $(w_i, w_{i+1}\dots w_j) \in I$, $(w_k, w_{j+1}\dots w_{k-1}) \in I$. Otherwise there would be a contradiction as $w =_G w_1 \dots w_{i-1} w_{i+1} \dots w_j [w_i w_k] w_{j+1} \dots w_{k-1} w_{k+1} \dots w_n$, which is shorter than w.

When talking about prefixes, suffixes or factors of a freely reduced element of a graph product, we mean a prefix (respectively suffix, factor) in the corresponding partially commutative monoid $M(\Gamma)$. By $\le_{\mathcal{L}}$ we denote a linear order on the set $\mathcal{L}$.

Example 4.2 Let $G_1 = \langle a, b\rangle$, $G_2 = \langle c\rangle$. Let $\mathcal{L} = \{1,2\}$ and $I = \emptyset$. Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$.

Consider the reduced word $w = [ab][c] \in G$. Here $[ab]$ is a prefix of w in the partially commutative monoid $M(\Gamma)$, but $[a]$ is not. $\square$

The geodesic problem geo_G, is the problem of computing a geodesic representative of an arbitrary group element $w \in G$. Jonathan Kausch [Kau17] has shown that the geodesic problem in graph products is uAC^0-Turing reducible to the word problem.

Lemma 4.3 *[Kau17, Theorem 6.3.3]*

- *Given a graph product* $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ *of f.g. groups, we have*

  $$\mathrm{geo}_G \in \mathsf{uAC}^0(\mathrm{WP}(G)).$$

  That is a geodesic of $w \in G$ *can be computed in* uAC^0 *with oracle gates for the word problem of* G*.*

- *Given a non-trivial class of f.g. groups* $\mathcal{C}$*, we have*

  $$\mathrm{geo}_{\mathrm{GP}\,\mathcal{C}} \in \mathsf{uAC}^0(\mathrm{WP}(\mathrm{GP}\,\mathcal{C})).$$

  That is a geodesic of $w \in G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$*, where* $G_1, \dots G_n$ *and* I *are part of the input, can be computed in* uAC^0 *with oracle gates for the uniform word problem of* $\mathrm{GP}\,\mathcal{C}$*.* □

We introduce the notion of a cyclically reduced word. A word $w \in \Gamma^*$ is cyclically reduced if for all $u, v \in \Gamma^*$ with $w =_G uv$ the transposed word vu is a geodesic. Jonathan Kausch [Kau17] obtained the following results on the complexity of computing a cyclically reduced conjugate in graph products.

Lemma 4.4 *[Kau17, Theorem 7.3.4]*

- *Given a graph product* $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ *of f.g. groups, a cyclically reduced conjugate of* $w \in G$ *can be computed in* uAC^0 *with oracle gates for the word problem of* G*.*

- *Given a non-trivial class of f.g. groups* $\mathcal{C}$*, a cyclically reduced conjugate of* $w \in G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$*, where* $G_1, \dots G_n$ *and* I *are part of the input, can be computed in* uAC^0 *with oracle gates for the uniform word problem of* $\mathrm{GP}\,\mathcal{C}$*.* □

Similar to graph groups, there is a tight connection between graph products and partially commutative monoids. We reformulate Lemma 2.4 for graph groups.

Lemma 4.5 *Let* $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ *and let the corresponding partially commutative monoid be* $M = M(\Gamma)/\{ab = ba \mid (a, b) \in I\}$*. Given* $u, v \in \Gamma^*$ *the following holds.*

(i) $u =_M v \implies u =_G v$*.*

(ii) If u *and* v *are freely reduced, then* $u =_M v \iff u =_G v$*.* □

Proof We look at two statements.

(i) It follows directly from the definition of M and G, that $u =_M v \implies u =_G v$.

(ii) We only need to show that for freely reduced u, v, we have $u =_G v \implies u =_M v$. As u and v are freely reduced, they do not contain factors of the form ab, where a and b are from the same base group and thus $ab =_G [ab]$. Therefore, they are equal up to the commutation of letters, which is an operation that is allowed in M, and thus $u =_M v$. ■

We define the family of free monoids $\mathcal{A} = \{\Gamma_\alpha \cup \Gamma_\beta \mid (\alpha, \beta) \in D\}$. This fulfills the requirements laid out in Section 2.4. Let Π be defined as in Section 2.4. Similar to graph groups, we adapt two lemmata for partially commutative monoids presented in [Dub86] to hold for graph products. Here, we require that all elements are freely reduced. For the second lemma this means that we additionally require w to not be from a single base group.

Lemma 4.6 *[Dub86, Proposition 1.2] Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. For freely reduced $u, v \in G$ we have $u =_G v$ if and only if $\Pi(u) = \Pi(v)$.* □

Lemma 4.7 *[Dub86, Proposition 1.7] Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. Let w be freely reduced in G, where w is not from a single base group G_α. Then, there are $u \in \Gamma^*$ and $t > 1$ with $w =_G u^t$ if and only if there is a family F with $\Pi(w) = F^t$.* □

A normal form is a unique representative of a group element. The geodesic presented above is not necessarily unique. However, a normal form can be obtained by choosing the geodesic representative that is lexicographically smallest. For this purpose assume we have a linear order on $\mathcal{L}$. This is also called the length lexicographical normal form. Let nf_G be the problem of computing the length lexicographical normal form of a group element of G. Jonathan Kausch [Kau17] obtained the following results on the complexity of the normal form problem in graph products.

Lemma 4.8 *[Kau17, Theorem 6.3.9] Given a graph product $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ of f.g. groups, it holds that*

$$\mathrm{nf}_G \in \mathsf{uAC}^0(\{\mathrm{nf}_{G_\alpha} \mid \alpha \in \mathcal{L}\} \cup \{\mathrm{WP}(G)\}).$$

That is the length lexicographical normal form of a group element $w \in G$ can be computed in uAC^0 with oracle gates for the normal form problem in each G_α and the word problem of G. □

Lemma 4.9 *[Kau17, Theorem 6.3.15] Given a non-trivial class of f.g. groups $\mathcal{C}$, we have*

$$\mathrm{nf}_{\mathrm{GP}\,\mathcal{C}} \in \mathsf{uAC}^0(\mathrm{C}_= \mathrm{L}^{\mathrm{nf}_\mathcal{C}}).$$

That is the length lexicographical normal form of $w \in G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$, where $G_1, \dots G_n$ and I are part of the input, can be computed in uAC^0 with oracle gates for the uniform word problem in $\mathrm{C}_= \mathrm{L}^{\mathrm{nf}_\mathcal{C}}$. □

4.1 Cyclic Normal Form

In this section we adapt the cyclic normal form for graph groups presented in [CGW09] to graph products. The key property of this cyclic normal form is that, given a word w, all cyclic normal forms conjugate to w are cyclic permutations of each other. The authors of [CGW09] have shown that their cyclic normal form can be computed in linear time for graph groups. We show that for a graph products G of f.g. groups it can be computed in uAC^0 with oracle gates for the word problem in G and for the normal form problem in each base group. We also look at the uniform case, where the groups and independence graph are part of the input.

Definition 4.10 Let $w \in \Gamma^*$ be cyclically reduced. We say w is a cyclic normal form if w is a length lexicographical normal form and all its cyclic permutations are length lexicographic normal forms.□

Let $u, v \in \Sigma^*$ be two elements of a partially commutative monoid M. We say that u and v are related by commutation if $u =_M v$ and we can write $u = w_1 \dots w_n$, $v = w_1 \dots w_{i-1} w_{i+1} w_i w_{i+2} \dots w_n$. We say that u and v are related by cycling if they are conjugate and we can write $u = w_1 \dots w_n$ and $v = w_n w_1 \dots w_{n-1}$.

Lemma 4.11 *Let u, v be cyclic normal forms. Then u and v are conjugate if and only if u is a cyclic permutation of v.* □

Proof This proof follows the proof given in [CGW09] for graph groups. If u is a cyclic permutation of v, then u and v are obviously conjugate. We only need to show the other direction. Assume u is conjugate to v. Then u and v are related by a sequence of commutations and cyclings.

$$u = w_1 \to w_2 \to \dots \to w_r = v$$

We show by induction on $\mathcal{L}$ that there is a sequence consisting of cyclings only, that relates u and v. If there are no commutations then we are done. Otherwise, let $\gamma \in \mathcal{L}$ be the largest element of $\mathcal{L}$ (with respect to $\leq_{\mathcal{L}}$), such that there is a distinguished letter $c \in G_\gamma$ in w that is involved in a commutation. We track the letter c during the sequence of cyclings and commutations. We write each w_i as $w_i = w_i' c w_i''$.

Let $w_j \to w_{j+1}$ be the first commutation involving c. Since w is a cyclic normal form and there is no element larger than c involved in a commutation, this commutation must have the form $w_j = w_{j+1}' bcw_j'' \to w_{j+1}' cbw_j'' = w_{j+1}$, where $b \in G_\beta$ with $(\beta, \gamma) \in I$ and $\beta < \gamma$. The takeaway from looking at the first commutation involving the distinguished letter c is that there is at least one commutation involving c where c commutes to the left of a smaller letter.

Let $w_p \to w_{p+1}$ be the last commutation where c commutes to the left of a smaller letter. This commutation has the form $w_p = w_{p+1}' acw_p'' \to w_{p+1}' caw_p'' = w_{p+1}$, where $a \in G_\alpha$ with $(\alpha, \gamma) \in I$ and $\alpha < \gamma$. We also track the distinguished letter a.

Assume that the distinguished letters a and c do not commute again. Recall that we can write each w_q with $q > p$ as $w_q = w_q' c w_q''$. We look at the cyclic permutation $cw_q'' w_q'$ and consider the position of a. We have $cw_q'' w_q' = cy_q az_q$ where $(cy_q, a) \in I$. Thus, $cy_q az_q$ represents the same group element as $acy_q z_q$ and therefore we obtain the contradiction that any w_q, especially $w_r = v$ is not a cyclic normal form.

It follows, that the distinguished letters a and c will commute again in the opposite direction. Let $w_s = c$ denote this commutation in the opposite direction. We have the following sequence of commutations and cycling from w_p to w_{s+1}.

$$w_p = w_{p+1}' acw_p'' \to w_{p+1}' caw_p'' \to \dots \to w_s' caw_{s+1}'' \to w_s' acw_{s+1}'' = w_{s+1}$$

We can replace it with a new sequence

$$w_p = w_{p+1}' acw_p'' \to v_{p+1}' acv_{p+1}'' \to \dots \to v_{s-1}' acv_{s-1}'' \to v_s' acv_s'' = w_{s+1},$$

where $v_q'' v_q'$ is a cyclic permutation of $y_q z_q$.

The new sequence has two commutations involving the distinguished letter c less then the original sequence. Applying this argument repeatedly we conclude that there is a sequence with no commutations involving the distinguished letter c, or any other letter from G_γ. By induction, there is a sequence from u to v that does not involve any commutations. ■

Theorem 4.12 *In the following, w is irreducible and connected.*

- *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. Then a cyclic normal form conjugate to a group element $w \in G$ can be computed in uAC^0 with oracle gates for the normal form problem in each G_α and the word problem of G.*
- *Given a non-trivial class of f.g. groups C, a cyclic normal form conjugate of $w \in G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$, where $G_1, \dots G_n$ and I are part of the input, can be computed in uAC^0 with oracle gates for the uniform word problem in $\mathrm{C}_= \mathrm{L}^{\mathrm{nf}_C}$.* □

Proof Let $w \in \Gamma^*$ be the input. Let $\sigma = |\mathcal{L}|$. We can assume w to be cyclically reduced, as a cyclically reduced conjugate can be computed in uAC^0 with oracle gates for the word problem in G by Lemma 4.4. A cyclic normal form of w can be computed with the following steps.

(i) Compute the length lexicographical normal form $\tilde{w} = \mathrm{nf}_G(w^\sigma)$.

(ii) Let $\tilde{w} = ydz$, where $d \in G_\alpha$ such that α is maximal w. r. t. $\leq_{\mathcal{L}}$, $y \in (\Gamma \setminus G_\alpha)^*$ and $z \in \Gamma^*$. Compute the cyclic permutation dzy. That is, we rotate the first occurrence of d to the front

(iii) Compute the length lexicographical normal form $\mathrm{nf}_G(dzy)$. We have $\mathrm{nf}_G(dzy) = u^\sigma$, where u is a cyclic normal form conjugate to w.

First, we look at the complexity in the non-uniform case. Our algorithm requires solving the normal form problem twice, which by Lemma 4.8 can be done in uAC^0 with oracle gates for the normal form problem in each G_α and the word problem of G. In addition, we need to compute a cyclic permutation which can be done in uAC^0.

Second, we look at the complexity in the uniform case. By Lemma 4.9, solving the normal form problem can be done in uAC^0 with oracle gates for the uniform word problem in $\mathrm{C}_= \mathrm{L}^{\mathrm{nf}_C}$. Again, the cyclic permutation can be computed in uAC^0.

Third, we show that our algorithm is correct, i. e., it computes a cyclic normal form. Observe that there can be no prefix of z that commutes with d. That is because $\tilde{w}$ is a length lexicographic normal form and d is chosen such that there is no larger letter that commutes with d. Additionally, there can be no prefix of y that commutes with dz. Assume there is such a prefix $\tilde{y}$. Then $y\tilde{y}$ is a prefix of $w^{2\sigma}$, and we can write $y\tilde{y} =_G w_1 \dots w_k$ where each w_i is a prefix of w and $k \geq \sigma$. As $y\tilde{y}$ commutes with d, w_1 is not equivalent to w, a contradiction to Lemma 2.3.

From these facts it follows that if $dzy =_G ez'y'$, then $d = e$. The second normal form computation does not alter the letters, it only rearranges them, as the individual letters are in normal form as a result of the previous normal form computation. Thus, $\mathrm{nf}_G(dzy)$ must start with the letter d.

Let $\hat{u} = y^{-1}wy$. Obviously $\mathrm{nf}_G(dzy) =_G \hat{u}^\sigma$. We show that $\mathrm{nf}_G(dzy) = \mathrm{nf}_G(\hat{u})^\sigma$ by showing that $\mathrm{nf}_G(\hat{u})^\sigma$ is normal. Observe that $\mathrm{nf}_G(\hat{u})$ is cyclically reduced, as w is cyclically reduced and y is a prefix of w^σ. It follows that $\mathrm{nf}_G(\hat{u})^\sigma$ is cyclically reduced.

Assume that $\mathrm{nf}_G(\hat{u})^\sigma$ is not the length lexicographic normal form. Then we have $k < \sigma$, $a \in \Gamma_\alpha$ and $b \in \Gamma_\beta$ with $(\alpha, \beta) \in I, \alpha <_{\mathcal{L}} \beta$ and $\mathrm{nf}_G(\hat{u})^k = x_1 b x_2$, $\mathrm{nf}_G(\hat{u}) = y_1 a y_2$ such that bx_2y_1 is reduced, but bx_2y_1a is not. It follows that a commutes with bx_2y_1. Most importantly a commutes with y_1. Hence, a is a prefix of y_1 and therefore also a prefix of $\mathrm{nf}_G(\hat{u})$. It follows $a = d$, which is a contradiction as d is the largest letter in w, but a is smaller than some letter b in w.

We conclude that $\mathrm{nf}_G(\hat{u})^\sigma$ is the length lexicographic normal form and therefore $\mathrm{nf}_G(dzy) = \mathrm{nf}_G(\hat{u})^\sigma$. It remains to show that $u = \mathrm{nf}_G(\hat{u})$ is a cyclic normal form. Let v be a cyclic permutation of u. Then v is a factor of $\mathrm{nf}_G(\hat{u})^\sigma$ which is the length lexicographic normal form. Therefore, v is also in length lexicographic normal form. We conclude that u is a cyclic normal form. ■

In a fixed graph group the normal form problem and the word problem can be solved in uAC^0 with oracle for the word problem of the free group F_2 [Kau17, Corollary 5.6.7, Theorem 6.3.9]. We derive the following special case of the above lemma for graph groups.

Corollary 4.13 *Let G be a graph group. Then a cyclic normal form conjugate of some irreducible, connected $w \in G$ can be computed in* $\mathsf{uAC}^0(\mathrm{WP}(F_2))$. □

4.2 Input Encoding

In this section we look at how to encode the input for the power word problem in graph groups and graph products. In the following we will use blocks of constant size to encode the different parts of the input. This type of encoding is most suitable to the AC^0-Turing reductions used in this thesis.

Elements of the graph group G are represented by words over Σ. Let the input for the power word problem be $p_1^{x_1} \dots p_n^{x_n}$. Let ℓ be the maximal number of bits required to encode any p_i and let r be the maximum number of bits required to encode any x_i. We may assume that $n \geq \ell$ and $n \geq r$, as we can pad the input with the identity element to achieve this. Now the input is encoded in $2n$ blocks of n bits each as follows.

p_1	x_1	$\cdots$	p_n	x_n

When looking at graph products we need to spend some more time thinking about how we encode a group element p_i. The group element is represented as a word over Σ, the disjoint union of the alphabets of the base groups. Let $p_i = p_{i,1} \dots p_{i,m}$. Let k be the number of bits required to encode the id of the base group (some $\alpha \in \mathcal{L}$). Let ℓ be the maximum number of bits required to encode any character of any Σ_α. Again we can assume $k \geq n$, $\ell \geq n$ and $n \geq m$, and pad with zeros/ the identity element as necessary. This leads to the following binary encoding of a single p_i requiring $2n$ blocks of n bits each.

$\mathrm{alph}(p_{i,1})$	$p_{i,1}$	$\cdots$	$\mathrm{alph}(p_{i,n})$	$p_{i,n}$

We encode the input to the power word problem similar to graph groups, the difference is in the encoding of a p_i, which now requires $2n^2$ bits, and therefore we need $2n^3 + n^2$ bits in total.

p_1	x_1	$\cdots$	p_n	x_n

5 Conjugacy in Graph Groups and Graph Products

In this chapter we look at specific conditions under which two elements of a graph group or graph product are conjugate. More precisely we are interested in conditions for two powers p^x and q^y under which p is conjugate to q. The results presented in this chapter are essential to solving the power word problem in the subsequent chapters.

We begin with an intermediate lemma on conjugacy in partially commutative monoids. On top of that we build our theorems giving conditions under which two elements of a graph group (respectively graph product) are conjugate or even equal.

Lemma 5.1 *Let M be a partially commutative monoid. Let $p, q, v \in M$, $x, y \in \mathbb{N}$ such that p and q are primitive and connected and v is a common factor of p^x and q^y. For each projection π_i we can write $\pi_i(p) = \tilde{p}_i^{s_i}$ and $\pi_i(q) = \tilde{q}_i^{r_i}$ where $\tilde{p}_i$ and $\tilde{q}_i$ are primitive. If p^2 is a factor of v and q^2 is a factor of v then for all projections $\tilde{p}_i$ and $\tilde{q}_i$ are conjugate as words and $r_i = s_i$.* □

Proof We look at a projection π_i. Since v is a factor of p^x and q^y, it holds that $\pi_i(v)$ is a factor of $\pi_i(p^x) = \tilde{p}_i^{s_i}$ and $\pi_i(q^y) = \tilde{q}_i^{r_i}$. Thus $\pi_i(v)$ has periods $|\tilde{p}_i|$ and $|\tilde{q}_i|$. As p^2 is a factor of v, $\pi_i(p^2)$ is a factor of $\pi_i(v)$. This yields the lower bound $2|\tilde{p}_i|$ on the length of $\pi_i(v)$. By symmetry, we obtain the lower bound $2|\tilde{q}_i|$ on the length of $\pi_i(v)$. Combining those we have

$$|\pi_i(v)| \geq \max\{2|\tilde{p}_i|, 2|\tilde{q}_i|\} \geq |\tilde{p}_i| + |\tilde{q}_i| \geq |\tilde{p}_i| + |\tilde{q}_i| - 1$$

Thus, by the theorem of Fine and Wilf [FW65], we have that $|\tilde{p}_i| = |\tilde{q}_i|$. As p is a factor of v, we have that $\tilde{p}_i$ is a factor of $\pi_i(v)$ and by transitivity also of $\pi_i(q^y) = \tilde{q}_i^{r_i \cdot y}$. Therefore $\tilde{p}_i$ and $\tilde{q}_i$ are conjugate.

Assume that for some i we have $s_i \neq r_i$. Then there are α and β such that $\alpha s_i = \beta r_i$. W. l. o. g. let $\beta \neq 1$ and $\gcd\{\alpha, \beta\} = 1$. Now β divides s_i.

Let J be the set of indices j such that $\alpha s_j = \beta r_j$. Clearly $i \in J$. Let l be an index such that $A_l \cap A_j \neq \emptyset$ for some j in J. Let $a \in A_l \cap A_j$ We have $s_l|\tilde{p}_l|_a = |p|_a = s_j|\tilde{p}_j|_a$ as the number of a's in each projection is $|p|_a$. Similarly, we have $r_l|\tilde{q}_l|_a = |q|_a = r_j|\tilde{q}_j|_a$, which is equivalent to $r_l|\tilde{p}_l|_a = r_j|\tilde{p}_j|_a$ (as $\tilde{p}_i$ and $\tilde{q}_i$ are conjugate for all i). Thus, we obtain

$$\alpha s_l|\tilde{p}_l|_a = \alpha s_j|\tilde{p}_j|_a = \beta r_j|\tilde{p}_j|_a = \beta r_l|\tilde{p}_l|_a$$

Comparing coefficients, the result $\alpha s_l = \beta r_l$ is obtained. By induction, we get $J = \{1, \ldots, k\}$ thus every s_i is divisible by β, and we can write $p = u^\beta$ contradicting p being primitive. ■

When applying the lemma to a graph group we need p, q and v to be freely reduced, so that we can apply our knowledge about partially commutative monoids. Additionally, we require p and q to be cyclically reduced, as p^x and q^x need to be freely reduced.

Corollary 5.2 *Let G be a graph group. Let $p, q, v \in G$, $x, y \in \mathbb{N}$ such that p and q are primitive, connected and cyclically reduced, v is freely reduced and v is a common factor of p^x and q^y. For each projection π_i we can write $\pi_i(p) = \tilde{p}_i^{s_i}$ and $\pi_i(q) = \tilde{q}_i^{r_i}$ where $\tilde{p}_i$ and $\tilde{q}_i$ are primitive. If p^2 is a factor of v and q^2 is a factor of v then for all projections $\tilde{p}_i$ and $\tilde{q}_i$ are conjugate as words and $r_i = s_i$.* □

To apply the lemma to graph products we additionally require that neither p nor q are from a single base group, as otherwise a power of p (respectively q) would not be freely reduced, which is a requirement to apply the lemmata about partially commutative monoids. The term freely reduced refers to a geodesic in the context of graph products.

Corollary 5.3 *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. Let $p, q, v \in G$, $x, y \in \mathbb{N}$ such that p and q are primitive, connected and cyclically reduced, v is freely reduced, v is a common factor of p^x and q^y, and there is no $\alpha \in \mathcal{L}$ such that $p \in G_\alpha$ or $q \in G_\alpha$. For each projection π_i we can write $\pi_i(p) = \tilde{p}_i^{s_i}$ and $\pi_i(q) = \tilde{q}_i^{r_i}$ where $\tilde{p}_i$ and $\tilde{q}_i$ are primitive. If p^2 is a factor of v and q^2 is a factor of v then for all projections $\tilde{p}_i$ and $\tilde{q}_i$ are conjugate as words and $r_i = s_i$.* □

5.1 Graph Groups

We begin with a theorem giving conditions under which for two powers p^x and q^y we have that p is equal to q in a graph group. The proof follows almost directly from the previous lemmata.

Theorem 5.4 *Let G be a graph group. Given $p, q \in G$ with p and q being primitive, connected and cyclically reduced, positive integers x and y, a suffix β of p^x and a prefix α of q^y, where β and α are freely reduced and $\beta\alpha =_G 1$, if $|\beta| = |\alpha| > (\sigma + 1) \cdot \max\{|p|, |q|\}$ then $p =_G q^{-1}$.* □

Proof Using Lemma 2.3 and the length bound on α and β we obtain that p^2 is a suffix of β and q^2 is a prefix of α. Observe that $\beta =_G \alpha^{-1}$. Thus β is a common suffix of p^x and q^{-y}. p^2 and q^{-2} are both suffixes of β. By Corollary 5.2 the projections $\pi_i(p)$ and $\pi_i(p^{-1})$ are conjugate. As both are suffixes of $\pi_i(\beta)$ they are equal, i. e., $\pi_i(p) = \pi_i(q^{-1})$. Using Lemma 2.5 we obtain $p =_G q^{-1}$. ■

We continue with a more intricate result on the conjugacy of two group elements.

Theorem 5.5 *Let G be a graph group. Let Ω be a set of words $w \in G$ with the following properties.*

- *$w \neq_G 1$,*
- *w is primitive,*
- *w is connected,*
- *w is cyclically reduced,*
- *w uniquely represents its conjugacy class and the conjugacy class of its inverse.*

Given $\hat{p}, \hat{q} \in \Omega$, $\hat{x}, \hat{y} \in \mathbb{Z}$, a factor u of $\hat{p}^x$ and a factor v of $\hat{q}^y$, where u and v are freely reduced and $uv =_G 1$, if $|u| = |v| > 2(\sigma + 1)(|\hat{p}| + |\hat{q}|)$ then $\hat{p} =_G \hat{q}$. □

Proof We have $u =_G v^{-1}$, thus v^{-1} is a factor of $\hat{p}^x$ and therefore v is a factor of $\hat{p}^{-x}$.

Let $\Omega^{\pm} = \Omega \cup \Omega^{-1}$ be the extension of Ω that includes the inverse of each element. Let $x = |\hat{x}|$, $y = |\hat{y}|$. There are $p, q \in \Omega^{\pm}$ such that $p \in \{\hat{p}, \hat{p}^{-1}\}$, $q \in \{\hat{q}, \hat{q}^{-1}\}$ and v is a common factor of p^x and q^y.

As $|v| \geq 2\sigma(|p|+|q|) \geq 2\sigma|p| > |p|$, v cannot be a factor of p. Thus, by Lemma 2.3 v can be written as $v = u_1 \cdots u_t p^z v_s \cdots v_1$. The length of v is given by $|v| = |u_1| + \cdots + |u_t| + z \cdot |p| + |w_s| + \cdots + |w_1|$. Solving for z we obtain $z = \frac{|v|-|u_1|-\cdots-|u_t|-|w_s|-\cdots-|w_1|}{|p|}$. Since w_i and u_j are factors of p, we have $|u_i|, |w_j| \leq |p|$. It follows that $z \geq \frac{|v|-(t+s)|p|}{|p|}$. We know that $t, s < \sigma$, thus $t, s \leq \sigma - 1$ and $(t + s) \leq 2\sigma - 2$. Therefore $z \geq \frac{|v|-2\sigma|p|+2|p|}{|p|}$. The inequality on the length of v gives us $|v| \geq 2\sigma(|p| + |q|) \geq 2\sigma|p|$. Plugging that in we obtain $z \geq \frac{2\sigma|p|-2\sigma|p|+2|p|}{|p|} = \frac{2|p|}{|p|} = 2$. Hence, p^2 is a factor of v. By symmetry q^2 also is a factor of v.

We write $\pi_i(p^x) = \tilde{p}_i^{s_i|x|}$ and $\pi_i(q^x) = \tilde{q}_i^{s_i|x|}$. By Corollary 5.2 the projections $\pi_i(p)$ and $\pi_i(q)$ are conjugate. As q is a factor of v it is a factor of p^x. We write $p^x = \alpha q \beta$. Clearly $q\beta\alpha$ is conjugate to p^x. Therefore $\pi_i(q\beta\alpha)$ is conjugate to $\pi_i(p^x)$. Since $\pi_i(q\beta\alpha)$ starts with $\tilde{q}_i$, has period $|\tilde{q}_i|$ and length $s_i|x| \cdot |\tilde{q}_i|$, we have $\pi_i(q\beta\alpha) = \tilde{q}_i^{s_i|x|}$. It follows that $q\beta\alpha = q^{|x|}$. Thus p^x is conjugate to $q^{|x|}$ and therefore p is conjugate to q. Since p and q uniquely represent their conjugacy class it follows that $p = q$. As $\hat{p}$ and $\hat{q}$ uniquely represent their conjugacy class and the conjugacy class of their respective inverses, which in the first case includes p, and in the second case includes q, it must hold that $\hat{p} = \hat{q}$. ∎

5.2 Graph Products

We adapt the theorems from the previous section to graph products. The main difference is that we prohibit elements which can be expressed as an element of a single base group. The proofs in this section are a verbatim copy of the proofs we have presented for graph groups.

Theorem 5.6 *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. Let $p, q \in G$ such that p and q are primitive, connected, cyclically reduced, and there is no $\alpha \in \mathcal{L}$ such that $p \in G_\alpha$ or $q \in G_\alpha$. Given positive integers x and y, a suffix β of p^x and a prefix α of q^y, where β and α are freely reduced and $\beta\alpha =_G 1$, if $|\beta| = |\alpha| > (\sigma + 1) \cdot \max\{|p|, |q|\}$ then $p =_G q^{-1}$.* □

Proof Using Lemma 2.3 and the length bound on α and β we obtain that p^2 is a suffix of β and q^2 is a prefix of α. Observe that $\beta =_G \alpha^{-1}$. Thus β is a common suffix of p^x and q^{-y}. Both p^2 and q^{-2} are suffixes of β. By Corollary 5.3 the projections $\pi_i(p)$ and $\pi_i(p^{-1})$ are conjugate. As both are suffixes of $\pi_i(\beta)$ they are equal, i. e., $\pi_i(p) = \pi_i(q^{-1})$. Using Lemma 4.6 we obtain $p =_G q^{-1}$. ∎

Theorem 5.7 *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. Let Ω be a set of words $w \in G$ with the following properties.*

- *$w \neq_G 1$,*
- *w is primitive,*
- *w is connected,*
- *w is cyclically reduced,*

- *w uniquely represents its conjugacy class and the conjugacy class of its inverse.*
- *There is no $\alpha \in \mathcal{L}$, such that $w \in G_\alpha$.*

Given $\hat{p}, \hat{q} \in \Omega$, $\hat{x}, \hat{y} \in \mathbb{Z}$, a factor u of $\hat{p}^x$ and a factor v of $\hat{q}^y$, where u and v are freely reduced and $uv =_G 1$, if $|u| = |v| > 2(\sigma + 1)(|\hat{p}| + |\hat{q}|)$ then $\hat{p} =_G \hat{q}$. □

Proof We have $u =_G v^{-1}$, thus v^{-1} is a factor of $\hat{p}^x$ and therefore v is a factor of $\hat{p}^{-x}$.

Let $\Omega^{\pm} = \Omega \cup \Omega^{-1}$ be the extension of Ω that includes the inverse of each element. Let $x = |\hat{x}|$, $y = |\hat{y}|$. There are $p, q \in \Omega^{\pm}$ such that $p \in \{\hat{p}, \hat{p}^{-1}\}$, $q \in \{\hat{q}, \hat{q}^{-1}\}$ and v is a common factor of p^x and q^y.

As $|v| \geq 2\sigma(|p|+|q|) \geq 2\sigma|p| > |p|$, v cannot be a factor of p. Thus, by Lemma 2.3 v can be written as $v = u_1 \cdots u_t p^z v_s \cdots v_1$. The length of v is given by $|v| = |u_1| + \cdots + |u_t| + z \cdot |p| + |w_s| + \cdots + |w_1|$. Solving for z we obtain $z = \frac{|v|-|u_1|-\cdots-|u_t|-|w_s|-\cdots-|w_1|}{|p|}$. Since w_i and u_j are factors of p, we have $|u_i|, |w_j| \leq |p|$. It follows that $z \geq \frac{|v|-(t+s)|p|}{|p|}$. We know that $t, s < \sigma$, thus $t, s \leq \sigma - 1$ and $(t + s) \leq 2\sigma - 2$. Therefore $z \geq \frac{|v|-2\sigma|p|+2|p|}{|p|}$. The inequality on the length of v gives us $|v| \geq 2\sigma(|p| + |q|) \geq 2\sigma|p|$. Plugging that in we obtain $z \geq \frac{2\sigma|p|-2\sigma|p|+2|p|}{|p|} = \frac{2|p|}{|p|} = 2$. Hence, p^2 is a factor of v. By symmetry q^2 also is a factor of v.

We write $\pi_i(p^x) = \tilde{p}_i^{s_i|x|}$ and $\pi_i(q^x) = \tilde{q}_i^{s_i|x|}$. By Corollary 5.3 the projections $\pi_i(p)$ and $\pi_i(q)$ are conjugate. As q is a factor of v, it is a factor of p^x. We write $p^x = \alpha q \beta$. Clearly $q\beta\alpha$ is conjugate to p^x. Therefore, $\pi_i(q\beta\alpha)$ is conjugate to $\pi_i(p^x)$. Since $\pi_i(q\beta\alpha)$ starts with $\tilde{q}_i$, has period $|\tilde{q}_i|$ and length $s_i|x| \cdot |\tilde{q}_i|$, we have $\pi_i(q\beta\alpha) = \tilde{q}_i^{s_i|x|}$. It follows that $q\beta\alpha = q^{|x|}$. Thus p^x is conjugate to $q^{|x|}$ and therefore p is conjugate to q. Since p and q uniquely represent their conjugacy class, it follows that $p = q$. As $\hat{p}$ and $\hat{q}$ uniquely represent their conjugacy class and the conjugacy class of their respective inverses, which in the first case includes p, and in the second case includes q, it must hold that $\hat{p} = \hat{q}$. ■

6 The Power Word Problem in Graph Groups

In this section we show that the power word problem in graph groups can be decided in polynomial time and is uAC^0-Turing-reducible to the word problem in the free group F_2. To achieve this we show that the exponents can be replaced with smaller ones, bounded by a polynomial in the length of the input.

Our proof uses ideas and techniques presented by Lohrey and Weiß [LW19], who studied the power word problem in free groups. Following their approach we divide our proof into the following major steps.

- In a preprocessing step we replace all powers with powers of a word in Ω.
- We define a symbolic rewriting system, which we will use to prove correctness of the reduction.
- We define the shortened word, calculating how much can be removed from each exponent.

Theorem 6.1 *Let G be a graph group. The power word problem in G is uAC^0-Turing reducible to the word problem in the free group F_2.* □

6.1 Preprocessing

The preprocessing consists of three steps.

1. Cyclically reducing powers.
2. Replacing powers with powers of connected words.
3. Replace each power with a power of a word in Ω.

The input of the power word problem is a word over $\Sigma^* \times \mathbb{Z}$. However, we can also write it as a word over $\Sigma^* \cup (\Sigma^* \times \mathbb{Z})$, adding the identity element in between the powers. Thus, we write the input as $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$, where $u_i, p_i \in \Gamma^*$ and $x_i \in \mathbb{Z}$. Note that the u_i are 1 at the beginning, however this will change during the preprocessing.

Step 1: Cyclically reducing powers. For each p_i a cyclically reduced conjugate $\tilde{p}_i$ is computed. The power $p_i^{x_i}$ is replaced with $\alpha_i \tilde{p}_i^{x_i} \beta_i$, where $p_i^{x_i} =_G \alpha_i \tilde{p}_i^{x_i} \beta_i$ and $\alpha_i =_G \beta_i^{-1}$. By Lemma 2.3 we know there are α_i and β_i such that $|\alpha_i| = |\beta_i| < \sigma|\tilde{p}|$. We can view α_i as part of u_{i-1} and β_i as part of u_i, thus we can assume that for the next step the input again has the shape $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$.

Step 2: Replacing powers with powers of connected words. We compute connected components of p_i, i. e., $p_{i,1}, \dots p_{i,k}$ such that $p_i =_G p_{i,1} \dots p_{i,k}$, each $p_{i,j}$ is connected and for each $j \neq \ell$ it holds that $p_{i,\ell}$ commutes with $p_{i,j}$. Observe that $k \leq \sigma$. We replace the power $p_i^{x_i}$ with $p_{i,1}^{x_i} \dots p_{i,k}^{x_i}$.

Step 3: Replace each power with a power of a word in Ω. Let $\Omega \subseteq \tilde{\Gamma}^*$ be the set containing only words w with the following properties.

- $w \neq_G 1$,
- w is primitive,
- w is connected,
- w is cyclically reduced,
- w is a cyclic normal form,
- w is minimal among its cyclic permutations and the cyclic permutations of a cyclic normal form of its inverse.

This definition fulfills the requirements of Theorem 5.5.

All requirements except for w being primitive and the last two requirements are already fulfilled after the previous two preprocessing steps. To take care of the last two requirements, for each p_i we compute a cyclic normal form and a cyclic normal form of its inverse. Then we compute the minimal element $\tilde{p}_i$ w. r. t. some linear order on Σ among all cyclic permutations of the two cyclic normal forms. If $\tilde{p}_i$ is conjugate to the inverse of p_i we set $\tilde{x}_i = -x_i$, otherwise $\tilde{x}_i = x_i$. There are α_i and β_i with $p_i^{x_i} =_G \alpha_i \tilde{p}_i^{\tilde{x}_i} \beta_i$ such that $\alpha =_G \beta^{-1}$ and $|\alpha_i| = |\beta_i| < \sigma|\tilde{p}|$. Again, we view α_i as part of u_{i-1} and β_i as part of u_i, thus we can assume that the input for the next reduction step, that is computing the shortened word, has the shape $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$.

Now that each p_i is a cyclic normal form, it is easy to check whether a given p_i is primitive, and replace it with a primitive factor if necessary. As p_i is a cyclic normal form we know that if $p_i =_G q^r$, then $p_i = \mathrm{nf}(q)^r$ (compare proof of Theorem 4.12). Thus, we only need to check for periods in p_i.

Lemma 6.2 *Let G be a graph group with independence relation I. Then the preprocessing as described above can be done in* uAC^0 *with an oracle gate for the word problem in G.* □

Proof The cyclically reduced conjugate in step one can be computed in uAC^0 with oracle gate for the word problem in G by Lemma 4.4.

To compute connected components of a power $p_i^{x_i}$ we define the predicate $\mathrm{con}_{p_i}(j, \ell)$ for $j, \ell \in \Sigma$ which is true if j and ℓ are connected in $D(p_i)$, the dependence graph restricted to the letters in p_i. The two vertices are connected if there is an undirected path from j to ℓ of length at most $\sigma - 1$. The following equation is equivalent to $\mathrm{con}_{p_i}(j, \ell)$.

$$\exists i_1, \dots i_{\sigma-1} : i_1 = j \wedge i_{\sigma-1} = \ell \wedge (i_1, i_2) \notin I \wedge \dots \wedge (i_{\sigma-2}, i_{\sigma-1}) \notin I$$

Furthermore we define the predicate $\text{smallest}_{p_i}(j)$ which is true if j is the smallest letter in the connected component of j. The following equation is equivalent to $\text{smallest}_{p_i}(j)$.

$$\forall \ell \in \Sigma : \ell \geq j \vee \neg\, \text{con}_{p_i}(j, \ell)$$

We define the projections $\pi_{p_i,j}$ for $p_i = a_{i,1} \dots a_{i,k}$ by

$$a_{i,\ell} \mapsto \begin{cases} a_{i,\ell} & \text{if } \text{con}_{p_i}(j, a_{i,\ell}) \wedge \text{smallest}_{p_i}(j), \\ 1 & \text{otherwise.} \end{cases}$$

Observe that $p_i =_G \pi_{p_i,1} \dots \pi_{p_i,\sigma}$ and each $\pi_{p_i,j}$ is connected.

To compute a representative for p_i from Ω, we first compute a cyclic normal form conjugate to p_i and a cyclic normal form conjugate to p_i^{-1}. By Theorem 4.12, we can compute a cyclic normal form in uAC^0 with an oracle gate for the word problem in G. Computing cyclic permutations, selecting the smallest, and checking for periods in the word and replacing the word with a primitive factor can be done in uAC^0. Hence, we have shown, that the preprocessing can be done in uAC^0 using an oracle gate for the word problem in G. ∎

6.2 Symbolic Rewriting System

Let G be a graph group with independence relation I. Let S be a rewriting system for G defined by the following relations, where a is an element of Σ, a^{-1} is the inverse of a in Σ and $\mathbf{u}$ is an element of Σ^*.

$$a\mathbf{u}a^{-1} \to \mathbf{u} \qquad \text{if } (a, \mathbf{u}) \in I$$

For $p \in \Omega$ we define

$$\Delta_p = \left\{ \beta p^x \alpha \;\middle|\; \begin{array}{l} x \in \mathbb{Z}, \\ \alpha \in \text{IRR}(S) \text{ is a prefix of } p^{\sigma \operatorname{sgn} x},\ p \text{ is no prefix of } \alpha, \\ \beta \in \text{IRR}(S) \text{ is a suffix of } p^{\sigma \operatorname{sgn} x} \text{ and } p \text{ is no suffix of } \beta. \end{array} \right\}$$

We define the alphabet Δ by $\Delta' = \bigcup_{p \in \Omega} \Delta_p$, $\Delta'' = \Sigma$ and $\Delta = \Delta' \cup \Delta''$.

Lemma 6.3 *Given $\beta p^x \alpha \in \Delta'$ it holds that $|\alpha| < (\sigma - 1)|p|$ and $|\beta| < (\sigma - 1)|p|$.* □

Proof By Lemma 2.3 we can write $\alpha = p^k w_s \cdots w_1$ with $s < \sigma$ where each w_i is a prefix of p. As p is not a prefix of α we have $k = 0$. Regarding the length of α we obtain $|\alpha| = \sum_{i=1}^{s} |w_i| < \sum_{i=1}^{s} |p| = s|p| \leq (\sigma - 1)|p|$. The bound on the length of β follows by symmetry. ∎

The rewriting system T over Δ^* is defined by the following rules. In the following $\beta p^x \alpha, \delta p^y \gamma, \delta q^y \gamma \in \Delta'$; $a, b \in \Delta''$; $r \in \Delta''^*$; $d, e \in \mathbb{Z}$ and $\mathbf{u} \in \Delta^*$.

(6.1) $\beta p^x \alpha \mathbf{u} \delta p^y \gamma \to \beta p^{x+y+d} \gamma \mathbf{u}$ if $\alpha\delta \overset{*}{\underset{S}{\Longrightarrow}} p^d$ and $(p, \pi(\mathbf{u})) \in I$

(6.2) $\beta p^x \alpha \mathbf{u} \delta p^y \gamma \to \beta p^{x-d} \alpha' \mathbf{u} \delta' p^{y-e} \gamma$ if $\nexists c \in \mathbb{Z} : \alpha\delta \overset{*}{\underset{S}{\Longrightarrow}} p^c$ or $(p, \pi(\mathbf{u})) \notin I$, $\beta p^x \alpha \mathbf{u} \in \mathrm{IRR}(S), \mathbf{u}\delta p^y \gamma \in \mathrm{IRR}(S)$ and $p^x \alpha \mathbf{u} \delta p^y \overset{*}{\underset{S}{\Longrightarrow}} p^{x-d} \alpha' \mathbf{u} \delta' p^{y-e} \in \mathrm{IRR}(S)$

(6.3) $\beta p^x \alpha \mathbf{u} \delta q^y \gamma \to \beta p^{x-d} \alpha' \mathbf{u} \delta' q^{y-e} \gamma$ if $p^x \alpha \mathbf{u} \delta q^y \overset{*}{\underset{S}{\Longrightarrow}} p^{x-d} \alpha' \mathbf{u} \delta' q^{y-e} \in \mathrm{IRR}(S)$ and $p \neq q$

(6.4) $\beta p^x \alpha \to r$ if $\beta\alpha \overset{*}{\underset{S}{\Longrightarrow}} r \in \mathrm{IRR}(S)$ and $x = 0$

(6.5) $a \mathbf{u} \beta p^x \alpha \to \mathbf{u} \beta' p^{x-d} \alpha$ if $(a, \mathbf{u}) \in I$ and $a\mathbf{u}\beta p^x \overset{*}{\underset{S}{\Longrightarrow}} \mathbf{u}\beta' p^{x-d} \in \mathrm{IRR}(S)$

(6.6) $\beta p^x \alpha \mathbf{u} a \to \beta p^{x-d} \alpha' \mathbf{u}$ if $(a, \mathbf{u}) \in I$ and $p^x \alpha \mathbf{u} a \overset{*}{\underset{S}{\Longrightarrow}} p^{x-d} \alpha' \mathbf{u} \in \mathrm{IRR}(S)$

(6.7) $a \mathbf{u} b \to \mathbf{u}$ if $(a, \mathbf{u}) \in I$ and $a = b^{-1}$

Lemma 6.4 *For $u, v \in \Delta^*$ it holds that*

(i) $\pi(\mathrm{IRR}(T)) \subseteq \mathrm{IRR}(S)$,

(ii) $u \overset{*}{\underset{T}{\Longrightarrow}} v$ *implies* $\pi(u) \overset{*}{\underset{S}{\Longrightarrow}} \pi(v)$,

(iii) $\pi(u) =_G 1$ *if and only if* $u \overset{*}{\underset{T}{\Longrightarrow}} 1$. □

Proof Assume we have an element $t \in \mathrm{IRR}(T)$ with $\pi(t) \notin \mathrm{IRR}(S)$. Then there is a factor $a\mathbf{u}a^{-1}$ of t, where $a \in \Sigma$, $\mathbf{u} \in \Sigma^*$ and $(a, \mathbf{u}) \in I$. As the letters of t are irreducible over S the preimages of a and a^{-1} must be located in different letters of t. Let t_i and t_j be the letters of t that contain the preimages of a and a^{-1}. Note that a is a suffix of t_i and a^{-1} is a prefix of t_j. All the letters in between hold parts of the preimage of $\mathbf{u}$, thus commute with a. It follows that one of the rules of T can be applied contradicting $t \in \mathrm{IRR}(T)$. Thus, $\pi(\mathrm{IRR}(T)) \subseteq \mathrm{IRR}(S)$.

For (ii) observe that the rules of T only allows such reductions that are also allowed in S.

(iii) follows from (i) and (ii). If $u \overset{*}{\underset{T}{\Longrightarrow}} 1$ then $\pi(u) \overset{*}{\underset{S}{\Longrightarrow}} 1$ by (ii). If $u \overset{*}{\underset{T}{\Longrightarrow}} v \in \mathrm{IRR}(T)$ with $v \neq 1$ then $\pi(u) \overset{*}{\underset{S}{\Longrightarrow}} \pi(v)$ by (ii) and $\pi(v) \in \mathrm{IRR}(S)$ by (i). ■

Lemma 6.5 *The following length bounds hold:*

- *Rule* (6.2)*:* $|d| \leq 2\sigma$ *and* $|e| \leq 2\sigma$
- *Rule* (6.3)*:* $|d| \leq 3(\sigma + 1)|q|_\Sigma$ *and* $|e| \leq 3(\sigma + 1)|p|_\Sigma$
- *Rule* (6.4)*:* $|r|_\Sigma < 2(\sigma - 1)|p|_\Sigma$
- *Rules* (6.5) *and* (6.6)*:* $|d| \leq 1$ □

Proof When applying rule (6.2) we distinguish two cases. If $(p, \pi(\mathbf{u})) \in I$ then there is a prefix $\hat{\alpha} \in \mathrm{IRR}(S)$ of α and a suffix $\hat{\delta} \in \mathrm{IRR}(S)$ if δ such that $\alpha\delta \overset{*}{\underset{S}{\Longrightarrow}} \hat{\alpha}\hat{\delta} \in \mathrm{IRR}(S)$ and $\hat{\alpha}\hat{\delta}$ is no power of p. Let $p_1, p_2 \in \mathrm{IRR}(S)$ be cyclically reduced conjugates of $p^{\mathrm{sgn}(x)}$ and $p^{\mathrm{sgn}(y)}$ such that $p_1 =_G \hat{\alpha}^{-1} p^{\mathrm{sgn}(x)} \hat{\alpha}$ and $p_2 =_G \hat{\delta} p^{\mathrm{sgn}(y)} \hat{\delta}^{-1}$. As p is primitive and $\hat{\alpha} \neq_G \hat{\delta}^{-1}$ we have $p_1 \neq_G p_2$.

Let α'' be a suffix of $p^x\hat{\alpha}$ that cancels with a prefix δ'' of $\hat{\delta}p^y$. Then α'' is a suffix of p_1^∞ and δ'' is a prefix of p_2^∞. From Theorem 5.4 it follows $|\alpha''| \leq (\sigma+1)|p|_\Sigma$. By Lemma 6.3, $|\alpha'|_\Sigma < (\sigma-1)|p|_\Sigma$. We write $p^d\hat{\alpha} = \alpha'\alpha''$. It follows that $|p^d|_\Sigma + |\hat{\alpha}|_\Sigma = |\alpha'|_\Sigma + |\alpha''|_\Sigma$, and thus $d|p|_\Sigma \leq |\alpha'| + |\alpha''| \leq (\sigma-1)|p|_\Sigma + (\sigma+1)|p|_\Sigma$. Solving for d we obtain $d \leq 2\sigma$. The bound on $|e|$ follows by symmetry.

If $(p, \pi(\mathbf{u})) \notin I$ observe that any prefix v of δp^y that cancels with a suffix of $p^x\alpha$ must commute with $\mathbf{u}$. Thus, $p^{\mathrm{sgn}(y)}$ cannot be a factor of v and by Lemma 2.3 v must be a prefix of $\delta p^{(\sigma-1)\,\mathrm{sgn}(y)}$. Thus $|e| \leq \sigma - 1 < 2\sigma$. The bound on $|e|$ follows by symmetry.

When applying rule (6.3) a suffix of $p^x\alpha$ cancels with a prefix of δq^y. Thus we can write $p^d\alpha = \alpha'\alpha''$ and $\delta q^e = \delta''\delta'$ where $\alpha''\delta'' \overset{*}{\underset{S}{\Longrightarrow}} 1$. α'' is a factor of $\left(p^{\mathrm{sgn}(x)}\right)^\infty$ and δ'' is a factor of $\left(q^{\mathrm{sgn}(y)}\right)^\infty$. It holds that $|\alpha''|_\Sigma = |\delta''|_\Sigma \leq 2(\sigma+1)(|p|_\Sigma + |q|_\Sigma)$. Otherwise, we would have $p = q$ by Theorem 5.5 contradicting $p \neq q$. By Lemma 6.3, $|\alpha'|_\Sigma < (\sigma-1)|p|_\Sigma$ and $|\delta'|_\Sigma < (\sigma-1)|q|_\Sigma$. It follows $|p^d\alpha|_\Sigma = |\alpha'\alpha''|_\Sigma < (\sigma+1)2(\sigma+1)(|p|_\Sigma + |q|_\Sigma) + (\sigma-1)|p|_\Sigma < 3(\sigma+1)(|p|_\Sigma + |q|_\Sigma) \leq 3(\sigma+1)|p|_\Sigma|q|_\Sigma$. Combining this with $|d| \cdot |p|_\Sigma = |p^d|_\Sigma \geq |p^d\alpha|_\Sigma$ and solving for $|d|$ we obtain $|d| < 3(\sigma+1)|q|_\Sigma$. By symmetry, it follows that $|e| < 3(\sigma+1)|p|_\Sigma$.

In Rule (6.4) we have $|r|_\Sigma \leq |\alpha\beta|_\Sigma$. By Lemma 6.3, $|\alpha|_\Sigma < (\sigma-1)|p|_\Sigma$ and $|\beta|_\Sigma < (\sigma-1)|p|_\Sigma$. Therefore, $|r|_\Sigma < 2(\sigma-1)|p|_\Sigma$.

When applying rule (6.5) a^{-1} is a prefix of βp^x. Thus either a^{-1} is a prefix of β in which case $d = 0$ or if it is not a^{-1} must be a prefix of $p^{\mathrm{sgn}\,x}$ in which case $|d| = 1$. The same bound on rule (6.6) follows by symmetry. ∎

Definition 6.6 Let $w = w_1 \dots w_n \in \Delta^*$. We define

- $\mu(w) = \max\{|p|_\Sigma \mid w_i = \beta p^x \alpha \in \Delta'\}$,
- $\lambda(w) = |w|_{\Delta''} + \sum_{w_i = \beta p^x \alpha \in \Delta'} |p|_\Sigma$ and
- $\pi(w) = \pi(w_1) \dots \pi(w_n)$, $\pi(a) = a$ for $a \in \Delta''$ and $\pi(\beta p^x \alpha) = \beta p^x \alpha$ for $\beta p^x \alpha \in \Delta'$. □

Lemma 6.7 *The rewriting system T, applied to a word $w \in \Delta^*$, has the following properties.*

1. *Rules* (6.4) *and* (6.1) *can be applied at most $|w|_{\Delta'}$ times in total.*
2. *Rules* (6.2) *and* (6.3) *can be applied at most $2\sigma|w|_{\Delta'}$ times.*
3. *The number of applications of rules* (6.5)*,* (6.6) *and* (6.7) *is at most $2(\sigma-1)\lambda(w)$.* □

Proof

1. For an application $w_1 \underset{T}{\Longrightarrow} w_2$ of rule (6.4) or (6.1) it holds that $|w_1|_{\Delta'} > |w_2|_{\Delta'}$. Thus there can be at most $|w|_{\Delta'}$ applications of that rule.

2. When looking at the number of times the rules (6.2) and (6.3) can be applied we only need to consider the letters from Δ' in w. The rules can be applied only once to each pair of letters from Δ'. Furthermore, each letter $\beta p^x \alpha \in \Delta'$ can cancel with at most σ other letters to its right (and at most σ other letters to its left). Hence, up to $\sigma|w|_{\Delta'}$ applications are possible initially (or may be unblocked by applications of rules (6.5), (6.6) and (6.7)). Each removal of a letter from Δ' by rule (6.4) enables up to σ additional applications of rules (6.2) and (6.3). In total the two rules can be applied at most $2\sigma|w|_{\Delta'}$ times.
3. For a transition $w_1 \underset{T}{\Longrightarrow} w_2$ the following holds.
   - $|w_2|_{\Delta''} = |w_1|_{\Delta''}$ if the applied rule is (6.1), (6.2) or (6.3).
   - $|w_2|_{\Delta''} \leq |w_1|_{\Delta''} + 2(\sigma-1)|p|_\Sigma$ if the applied rule is (6.4) as that rule adds several letters from Δ'' to w_2, namely a prefix α of $p^{(\sigma-1)\operatorname{sgn}(x)}$ and a suffix β of $p^{(\sigma-1)\operatorname{sgn}(x)}$. We obtain the bound $|\alpha|_\Sigma + |\beta|_\Sigma \leq 2(\sigma-1)|p|_\Sigma$.
   - $|w_2|_{\Delta''} \leq |w_1|_{\Delta''} - 1$ if the applied rule is (6.5), (6.6) or (6.7).

   Let $w = a_1 \dots a_n$. Rule (6.4) is the only one that increases $|\cdot|_{\Delta''}$. It can be applied at most $|w|_{\Delta'}$ times. Its applications increase $|\cdot|_{\Delta''}$ by at most $\sum_{a_i=\beta_i p_i^x \alpha_i \in \Delta'} 2(\sigma-1)|p_i|_\Sigma = 2(\sigma-1)\sum_{a_i=\beta_i p_i^x \alpha_i \in \Delta'} |p_i|_\Sigma$. Each application of rule (6.5), (6.6) or (6.7) decreases $|\cdot|_{\Delta''}$ by at least one. Therefore, those rules can be applied at most $|w|_{\Delta''} + 2(\sigma-1)\sum_{a_i=\beta_i p_i^x \alpha_i \in \Delta'} |p_i|_\Sigma \leq 2(\sigma-1)\lambda(w)$ times. ∎

Corollary 6.8 *If $w \overset{*}{\underset{T}{\Longrightarrow}} v$, then $w \overset{\leq k}{\underset{T}{\Longrightarrow}} v$ with $k = 4\sigma\lambda(w)$.* □

Proof Each rule can only be applied finitely many times. Adding up the bounds from Lemma 6.7 we obtain a bound of $(2\sigma+1)|w|_{\Delta'} + 2(\sigma-1)\lambda(w) \leq (2\sigma+1)\lambda(w) + 2(\sigma-1)\lambda(w) \leq 4\sigma\lambda(w)$.∎

Definition 6.9 Let $w \in \Delta^*$ and $p \in \Omega$. We write $w = u_0\beta_1 p^{y_1}\alpha_1 u_1 \dots \beta_m p^{y_m}\alpha_m u_m$ with $u_i \in (\Delta \setminus \Delta_p)^*$. We define

$$\eta_p = \sum_{j=1}^{m} y_j,$$

$$\eta_p^i = \sum_{j=1}^{i} y_j$$

□

Lemma 6.10 *Let $u, v \in \Delta^*$ and $u \underset{T}{\Longrightarrow} v$. Given a prefix v' of v there is a prefix u' of u such that for all $p \in \Omega$*

$$|\eta_p(u') - \eta_p(v')| \leq 4(\sigma+1)\mu(u).$$

If the applied rule is neither (6.1) *nor* (6.4)*, then for all $p \in \Omega$ and $0 \leq i \leq m$*

$$|\eta_p^i(u) - \eta_p^i(v)| \leq 4(\sigma+1)\mu(u).$$ □

Proof We can assume, that v' contains a letter from Δ_p, otherwise the Lemma is trivial. Moreover, we can assume that $v \in \Delta^*\Delta_p$, as appending or removing letters from $\Delta \setminus \Delta_p$ does not change η_p.

Let the applied rule be $l \to r \in T$. To prove the first part of the lemma we distinguish three cases.

(i) The right-hand side of the rule is not in v'. In that case we choose $u' = v'$.

(ii) The right-hand side of the rule is completely in v'. We write $v' = \alpha r \beta$. We choose $u' = \alpha l \beta$. The Lemma follows with the bounds from Lemma 6.5.

(iii) The right-hand side r of the rule overlaps with v'. It follows that $r = \beta p^y \alpha \mathbf{u}$ and consequently we can write $l = \beta' p^x \alpha' \mathbf{u}'$. We write $v' = v'' \beta p^y \alpha$. If rule (6.1) has been applied we choose $u' = v'' l$. Otherwise, one of rules (6.2), (6.3) or (6.6) has been applied. We choose $u' = v'' \beta' p^x \alpha'$. The Lemma follows with the bounds from Lemma 6.5.

To see that the second statement of the lemma is true, observe that in case the applied rule is neither (6.1) nor (6.4), there is a one-to-one correspondence between letters from Δ_p in u and v. ■

6.3 The Shortened Word

In this section we describe the shortening process. Given $u \in \Delta^*$ and some $p \in \Omega$ we consider all letters from Δ_p in u. We write $u = u_0 \beta_1 p^{y_1} \alpha_1 u_1 \dots \beta_m p^{y_m} \alpha_m u_m$ with $u_i \in (\Delta \setminus \Delta_p)^*$. In the following we define a set C of intervals to be carved out of the exponents during the shortening process.

Definition 6.11 Let $C = \{[l_j, r_j] \mid 1 \le j \le k\}$ be a set of finite, non-empty, non-overlapping intervals, where $k = |C|$. We assume the intervals to be ordered, i. e., $r_j \le l_{j+1}$. We define the size of an interval $d_j = r_j - l_j + 1$.

An element $u \in \Delta^*$ is said to be **compatible** with C, if for every prefix u' of u it holds that $\eta_p(u') \notin [l_j, r_j]$ for all $1 \le j \le k$. □

Definition 6.12 Let C compatible with u. The shortened version of u is

$$\mathcal{S}_C(u) = u_0 \beta_1 p^{z_1} \alpha_1 u_1 \dots \beta_m p^{z_m} \alpha_m u_m.$$

The new exponents are defined as

$$z_i = y_i - \operatorname{sgn}(y_i) \cdot \sum_{j \in C_i} d_j,$$

where C_i is the set of intervals to be removed from y_i, defined by

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

Lemma 6.13 *If $u \in \mathrm{IRR}(T)$ then $\mathcal{S}_C(u) \in \mathrm{IRR}(T)$.* □

Proof We prove the lemma by showing that $\operatorname{sgn}(y_i) = \operatorname{sgn}(z_i)$ and $z_i \neq 0$. As the intervals in C are ordered, there are α and β such that C_i contains all indices from α to β. In case $y_i > 0$ we have

$$\begin{aligned} z_i &= y_i - \sum_{j \in C_i} d_j \\ &= y_i - \sum_{j=\alpha}^{\beta} d_j \\ &= y_i - \sum_{j=\alpha}^{\beta} (r_j - l_j + 1) \\ &\leq y_i - r_\beta - l_\alpha + 1 \\ &\leq y_i - (\eta_p^i(u) - 1) - (\eta_p^i(u) + 1) + 1 \\ &= y_i - \eta_p^i(u) - \eta_p^i(u) + 1 \\ &= y_i - y_i + 1 \\ &= 1 \end{aligned}$$

The case $y_i < 0$ follows by symmetry. ∎

Definition 6.14 We define the distance between some $\eta_p^i(\cdot)$ and the closest interval from C as

$$\operatorname{dist}_p(u, C) = \min \left\{ |\eta_p^i(u) - x| \mid 1 \leq i \leq m, x \in [l, r] \in C \right\}. \qquad \square$$

From that definition the following statement follows immediately.

Corollary 6.15 $\operatorname{dist}_p(u, C) > 0$ *if and only if* u *is compatible with* C. $\square$

We want to show, that given some requirements are fulfilled, any rewriting step that is possible on u is also possible on $\mathcal{S}_C(u)$.

Lemma 6.16 *If* $\operatorname{dist}_p(u, C) \geq 4(\sigma + 1)\mu(u)$ *and* $u \underset{T}{\Longrightarrow} v$*, then* $\mathcal{S}_C(u) \underset{T}{\Longrightarrow} \mathcal{S}_C(v)$. $\square$

Proof Observe that u is compatible with C. By Lemma 6.10 we have $\operatorname{dist}(v, C) > 0$ and thus v is compatible with C. It follows that $\mathcal{S}_C(u)$ and $\mathcal{S}_C(v)$ are well-defined.

To prove the Lemma we compare the shortened version of u and v and show that a rule from T can be applied.

We distinguish which rule from T has been applied to u.

- If rule (6.2), (6.3), (6.5), (6.6) or (6.7) has been applied, the shortening process has the same effect on u and v, i. e., $C_i(u) = C_i(v)$. The same rule that has been applied to u to obtain v can also be applied to $\mathcal{S}_C(u)$ to obtain $\mathcal{S}_C(v)$.
- If rule (6.4) is applied, then $C_\ell(v) = C_\ell(u)$ for $\ell < i$ and $C_\ell(v) = C_{\ell+1}(u)$ for $\ell \geq i$. We also know $y_i = 0$, which is not altered by the shortening process, i. e., $C_i(u) = \emptyset$. Thus, the same rule can be applied to $\mathcal{S}_C(u)$ to obtain $\mathcal{S}_C(v)$.

- Assume rule (6.1) has been applied.

  Let

  $$u = u_0\beta_1 p^{y_1}\alpha_1 u_1 \dots \beta_i p^{y_i}\alpha_i u_i \beta_{i+1} p^{y_{i+1}}\alpha_{i+1}u_{i+1}\dots\beta_m p^{y_m}\alpha_m u_m.$$

  The result of applying the rule is

  $$v = u_0\beta_1 p^{y_1}\alpha_1 u_1 \dots \beta_i p^{y_i+y_{i+1}+d}\alpha_{i+1} u_i u_{i+1}\dots\beta_m p^{y_m}\alpha_m u_m$$

  On powers not modified by the rule the shortening process behaves the same on u and v, i. e., $C_\ell(v) = C_\ell(u)$ for $\ell < i$ and $C_\ell(v) = C_{\ell+1}(u)$ for $\ell > i$. The result of the shortening process on v is

  $$\mathcal{S}_C(v) = u_0\beta_1 p^{z_1}\alpha_1 u_1 \dots \beta_i p^{\tilde{z}_i}\alpha_{i+1} u_i u_{i+1}\dots\beta_m p^{z_m}\alpha_m u_m,$$

  where $\tilde{z}_i = y_i + y_{i+1} + d - \operatorname{sgn}(y_i + y_{i+1} + d)\cdot\sum_{\ell\in C_i(v)} d_\ell$.

  When applying rule (6.1) to the corresponding letters of $\mathcal{S}_C(u)$ we obtain

  $$\hat{v} = u_0\beta_1 p^{z_1}\alpha_1 u_1 \dots \beta_i p^{z_i+z_{i+1}+d}\alpha_{i+1} u_i u_{i+1}\dots\beta_m p^{z_m}\alpha_m u_m,$$

  We need to show that $\tilde{z}_i = z_i + z_{i+1} + d$, i. e., $z_i + z_{i+1} = y_i + y_{i+1} - \operatorname{sgn}(y_i + y_{i+1} + d)\cdot\sum_{\ell\in C_i(v)} d_\ell$.

  Observe that $\operatorname{dist}(u, C) > d$, thus if $\operatorname{sgn}(y_i + y_{i+1} + d) \neq \operatorname{sgn}(y_i + y_{i+1})$ we have $C_i(u) = C_{i+1}(u)$, $C_i(v) = \emptyset$ and the lemma follows. From now on we can assume $\operatorname{sgn}(y_i + y_{i+1} + d) = \operatorname{sgn}(y_i + y_{i+1})$.

  First, consider the case that y_i and y_{i+1} have the same sign. In that case $C_i(v) = C_i(u)\cup C_{i+1}(u)$ we have

  $$\begin{aligned} z_i + z_{i+1} &= y_i - \operatorname{sgn}(y_i)\cdot\sum_{\ell\in C_i(u)} d_\ell + y_{i+1} - \operatorname{sgn}(y_{i+1})\cdot\sum_{\ell\in C_{i+1}(u)} d_\ell \\ &= y_i + y_{i+1} - \operatorname{sgn}(y_i + y_{i+1})\cdot\sum_{\ell\in C_i(v)} d_\ell \end{aligned}$$

  Second, we look at the case where y_i and y_{i+1} have opposite sign. We assume $|y_i| > |y_{i+1}|$. The other case is symmetric. Now $C_i(v) = C_i(u) \setminus C_{i+1}(u)$.

  $$\begin{aligned} z_i + z_{i+1} &= y_i - \operatorname{sgn}(y_i)\cdot\sum_{\ell\in C_i(u)} d_\ell + y_{i+1} - \operatorname{sgn}(y_{i+1})\cdot\sum_{\ell\in C_{i+1}(u)} d_\ell \\ &= y_i + y_{i+1} - \operatorname{sgn}(y_i)\left(\sum_{\ell\in C_i(u)} d_\ell - \sum_{\ell\in C_{i+1}(u)} d_\ell\right) \\ &= y_i + y_{i+1} - \operatorname{sgn}(y_i + y_{i+1})\cdot\sum_{\ell\in C_i(v)} d_\ell \end{aligned}$$

  ∎

Lemma 6.17 *If* $\operatorname{dist}_p(u, C) \geq 4k(\sigma+1)\mu(u)$ *and* $u \overset{\leq k}{\underset{T}{\Longrightarrow}} v$, *then* $\mathcal{S}_C(u) \overset{\leq k}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(v)$. □

Proof We prove the lemma by induction. If $k = 1$ then the statement follows from Lemma 6.16. If $k > 1$, then there is a $u' \in \Delta^*$ such that

$$u \underset{T}{\Longrightarrow} u' \overset{\leq k-1}{\underset{T}{\Longrightarrow}} v.$$

By Lemma 6.10 we have $\mathrm{dist}_p(u', C) \geq 4(k-1)(\sigma+1)\mu(u)$. As none of the rules of T increases $\mu(\cdot)$, it follows that $\mathrm{dist}_p(u', C) \geq 4(k-1)(\sigma+1)\mu(u')$. Therefore, $\mathcal{S}_C(u') \overset{\leq k-1}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(v)$ by induction. By Lemma 6.16 we have $\mathcal{S}_C(u) \overset{\leq k-1}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(u')$. Combining those statements we conclude $\mathcal{S}_C(u) \overset{\leq k}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(v)$. ∎

Lemma 6.18 *If* $\mathrm{dist}_p(u, C) \geq 16(\sigma+1)^2\lambda(u)^2$*, then* $\pi(u) =_G 1$ *if and only if* $\pi(\mathcal{S}_C(u)) =_G 1$. □

Proof Let $k = 4\sigma\lambda(u)$. From the precondition of the lemma we conclude $\mathrm{dist}_p(u, C) \geq 4k(\sigma+1)\lambda(u) \geq 4k(\sigma+1)\mu(u)$.

First, let $\pi(u) =_G 1$. By Lemma 6.4 that is equivalent to $u \overset{*}{\underset{T}{\Longrightarrow}} 1$. Which by Corollary 6.8 is equivalent to $u \overset{\leq k}{\underset{T}{\Longrightarrow}} 1$. By Lemma 6.17 we have $\mathcal{S}_C(u) \overset{\leq k}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(1) = 1$. Applying Corollary 6.8 and Lemma 6.4 again we obtain $\pi(\mathcal{S}_C(u)) =_G 1$.

Second, assume $\pi(u) =_G v \in \mathrm{IRR}(S)$, with $v \neq_G 1$. Again, applying Corollary 6.8 and Lemma 6.4 that is equivalent to $u \overset{\leq k}{\underset{T}{\Longrightarrow}} \tilde{v}$, with $\tilde{v} \in \mathrm{IRR}(T)$, $\tilde{v} \neq_G 1$. By Lemma 6.13 $\mathcal{S}_C(v)$ is irreducible. As the shortening process does not remove any letters, but only replaces them we have $\mathcal{S}_C(\tilde{v}) \neq_G 1$. Again, applying Corollary 6.8 and Lemma 6.4, we obtain $\pi(\mathcal{S}_C(u)) =_G \mathcal{S}_C(\tilde{v}) \neq_G 1$. ∎

We continue by defining a concrete set of intervals $C^K_{u,p}$ that will be used to show that the exponents of the shortened word are bounded by a polynomial.

Let $\{c_1, \ldots c_\ell\} = \left\{\eta^i_p(u) \mid 0 \leq i \leq m\right\}$ be the ordered set of $\eta^i_p(u)$, i. e., $c_1 < \cdots < c_\ell$. We define the set of intervals

$$C^K_{u,p} = \{[c_i + K, c_{i+1} - K] \mid 1 \leq i < \ell, c_{i+1} - c_i \geq 2K\}. \tag{6.8}$$

Lemma 6.19 *Let* $K = 16(\sigma+1)^2\lambda(u)^2$ *and* $\mathcal{S}_C(u) = u_0\beta_1 p^{z_1}\alpha_1 u_1 \ldots \beta_m p^{z_m}\alpha_m u_m$ *for some* $u \in \Delta^*$*. Then* $|z_i| \leq 32m(\sigma+1)^2\lambda(u)^2$ *for* $1 \leq i \leq m$. □

Proof

$$
\begin{aligned}
|z_i| &= \left| y_i - \operatorname{sgn}(y_i) \cdot \sum_{j \in C_i} d_j \right| \\
&= |y_i| - \sum_{j \in C_i} d_j \\
&\overset{(i)}{=} |y_i| - \sum_{j \in C_i} \max\{0, c_{j+1} - c_j - 2K + 1\} \\
&\leq |y_i| - \sum_{j \in C_i} c_{j+1} - c_j - 2K + 1 \\
&= |y_i| - \sum_{j \in C_i} c_{j+1} - c_j + \sum_{j \in C_i} 2K - 1 \\
&= \sum_{j \in C_i} 2K - 1 \overset{(ii)}{\leq} m(2K-1) \leq m2K
\end{aligned}
$$

We used the following facts.

(i) Definition of $C^K_{u,p}$ in (6.8).

(ii) $|C_i| \leq |C^K_{u,p}| \leq m$.

The lemma follows by plugging in the formula for K. ∎

Proof (of Theorem 6.1) To prove the lemma we apply the shortening process presented above and then solve the word problem.

We begin with the preprocessing described in Section 6.1. By Lemma 6.2 the preprocessing can be done in uAC^0 with an oracle gate for the word problem in G. Following the preprocessing we have a word $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$. We proceed with the shortening procedure for each $p \in \{p_i \mid 1 \leq i \leq n\}$. The shortening procedure can be computed in parallel for each p. The computation of the shortened word requires iterated additions which is in uTC^0, and therefore can be done in uAC^0 with oracle gates for $\mathrm{WP}(F_2)$. By Lemma 6.19 the exponents of the shortened word are bounded by a polynomial. Thus, the shortened word can be represented as a word $\hat{w}$ of polynomial length over the alphabet Σ. We can decide whether the power word w is equal to the identity by deciding whether the word $\hat{w}$ is equal to the identity, i. e., we have reduced the power word problem in G to the word problem in G.

The word problem in a graph group is uAC^0-Turing reducible to the word problem of the free group F_2 [Kau17, Corollary 5.6.7]. ∎

6.4 Example

In this section we will illustrate the algorithm for solving the power word problem with a detailed example. Let G be the graph group with the following dependence graph D.

$$a \text{ —— } b \text{ —— } c \qquad d \text{ —— } e$$

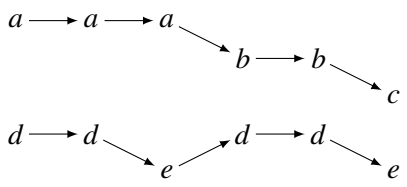
We will go through the individual steps of the algorithm for the following input word w.

$$w = (aaabbaa^{-1}c)^{2048000}\ (adaadebddbec)^{-4096000}\ (aaabbc)^{2048000}\ (ddd^{-1}de)^{8192000}$$

The first preprocessing step is to replace each power with a power of a cyclically reduced conjugate. We replace $aaabbaa^{-1}c$ with the cyclically reduced conjugate $aabbac$ and we replace $ddd^{-1}de$ by dde. In the second case only free reduction has been applied. The result of the preprocessing step is the following word.

$$a\ (aabbac)^{2048000}\ a^{-1}\ (adaadebddbec)^{-4096000}\ (aaabbc)^{2048000}\ (dde)^{8192000}$$

The second step is to identify connected components in the induced dependence graph of each power. Then we replace each power with powers of connected words. In our example this only affects $(adaadebddbec)^{-4096000}$. We can identify the connected components in the directed acyclic graph associated with the word $adaadebddbec$.



Splitting $adaadebddbec$ into the connected words $aaabbc$ and $ddedde$ we obtain

$$a\ (aabbac)^{2048000}\ a^{-1}\ (aaabbc)^{-4096000}\ (ddedde)^{-4096000}\ (aaabbc)^{2048000}\ (dde)^{8192000}.$$

The third and last preprocessing step is to replace each power with a power of some word in Ω. This includes making sure each word is primitive and finding conjugates in Ω. In our example $ddedde$ is not primitive. We replace $(ddedde)^{-4096000}$ with $(dde)^{-8192000}$. Then $aabbac$ is not in Ω, we have to use the conjugate $aaabbc$. This leads to the following word.

$$u = aa^{-1}\ (aaabbc)^{2048000}\ aa^{-1}\ (aaabbc)^{-4096000}\ (dde)^{-8192000}\ (aaabbc)^{2048000}\ (dde)^{8192000}$$

At this point it is easy to observe that the word we chose for our example is equal to the identity of the group. By Lemma 6.19, we have

$$K = 16(\sigma + 1)^2\lambda(u)^2 = 788544,$$

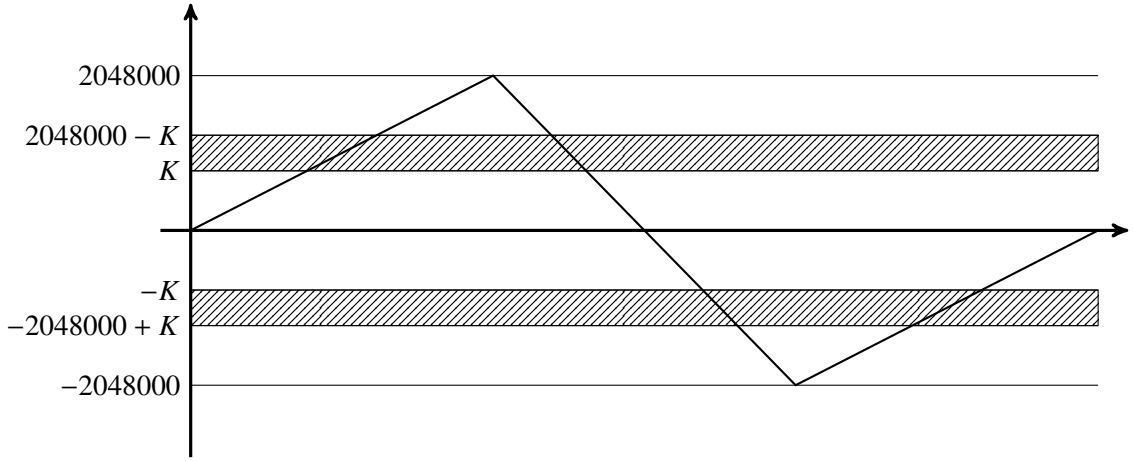


Figure 6.1: Computation of the shortened word.

with $\sigma = 5$ and $\lambda(u) = 37$. Note that, for the sake of the example, we have calculated $\lambda(u)$ of the word after the preprocessing. In an actual implementation we would need to use an upper bound based on the input length and the maximum length increase due to preprocessing.

The value K determines the size of the intervals to be cut out from the exponents, as defined in Equation (6.8). See Figure 6.1 for an illustration of the intervals when applying the shortening process to all powers of $aaabbc$. The hatched areas represent the intervals to be cut out. Our example is symmetric, both intervals have the size 470913. Applying the shortening process to all powers, we obtain the shortened word

$$aa^{-1}\ (aaabbc)^{1577087}\ aa^{-1}\ (aaabbc)^{-3154174}\ (dde)^{-1577087}\ (aaabbc)^{1577087}\ (dde)^{1577087}.$$

7 The Power Word Problem in Graph Products

In this section we show that the power word problem in graph products can be solved in uAC^0 with oracles for the word problem and for the power word problem of the base groups. We use the same techniques we have used in the previous chapter to solve the power word problem for graph groups. We show that the exponents can be replaced with smaller ones, bounded by a polynomial in the input. The polynomial is larger than the on we obtained for graph groups.

An important difference to the previous chapter on the power word problem in graph groups is, that we will be required to solve the power word problem in the base groups of the graph product. The reason is that we cannot replace exponents of powers that are of a single base group, with smaller ones. Those will remain after the reduction step. We define solving the word problem with powers of individual letters as the simple power word problem. The last two sections of this chapter are dedicated to solving the simple power word problem.

Definition 7.1 (simple power word problem) Let G be a graph product. The input consists of a list of letters $a_1, \dots, a_n \in \Gamma$ and a list of binary encoded integers $x_1, \dots, x_n \in \mathbb{Z}$. We interpret the input as $w = a_1^{x_1} \dots a_n^{x_n}$. The simple power word problem SPowWP(G) is to decide whether $w =_G 1$. □

Definition 7.2 (generalized simple power word problem) Let G be a graph product. Let $H \leq G$. The input consists of a list of letters $a_1, \dots, a_n \in \Gamma$ and a list of binary encoded integers $x_1, \dots, x_n \in \mathbb{Z}$. We interpret the input as $w = a_1^{x_1} \dots a_n^{x_n}$. The generalized simple power word problem GSPowWP(G, H) is to decide whether $w \in H$. □

We proceed similar to the previous chapter. Especially the sections on the symbolic rewriting system and the shortened word are in large parts identical with only minor adjustments.

- In a preprocessing step we replace all powers with powers of a word in Ω.
- We define a symbolic rewriting system, which we will use to prove correctness of the reduction.
- We define the shortened word, calculating how much can be removed from each exponent.
- We solve the simple power word problem.

Theorem 7.3 *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ be a graph product of f.g. groups. The power word problem in G can be decided in uAC^0 with oracle gates for the word problem in the free group F_2 and for the power word problem in each base group G_α.* □

Theorem 7.4 *Let $\mathcal{C}$ be a non-trivial class of f.g. groups, $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$. The uniform power word problem in G, where $G_1, \dots G_n$ and I are part of the input, can be solved in* $\mathrm{C}_=\mathrm{L}$ *with oracle gates for the uniform power word problem* $\mathrm{PowWP}\,\mathcal{C}$. □

7.1 Preprocessing

Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups.

The preprocessing consists of five steps.

1. Cyclically reducing powers.
2. Replacing powers with powers of connected words.
3. Removing powers of a single letter.
4. Replacing each letter with a normal form specific to the input.
5. Replacing each power with a power of a word in Ω.

Let $\Gamma = \bigcup_{\alpha\in\mathcal{L}} G_\alpha$ be the alphabet of G. The input of the power word problem is a word over $\Gamma^* \cup (\Gamma^* \times \mathbb{Z})$. We can write the input as $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$, where $u_i, p_i \in \Gamma^*$ and $x_i \in \mathbb{Z}$. Note that some u_i may be 1.

Step 1: Cyclically reducing powers. For each p_i a cyclically reduced conjugate $\tilde{p}_i$ is computed. The power $p_i^{x_i}$ is replaced with $\alpha_i \tilde{p}_i^{x_i} \beta_i$, where $p_i^{x_i} =_G \alpha_i \tilde{p}_i^{x_i} \beta_i$ and $\alpha_i =_G \beta_i^{-1}$. By Lemma 2.3 we know there are α_i and β_i such that $|\alpha_i| = |\beta_i| < |\mathcal{L}| \cdot |\tilde{p}|$. We can view α_i as part of u_{i-1} and β_i as part of u_i, thus we can assume that for the next step the input again has the shape $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$.

Step 2: Replacing powers with powers of connected words. We compute connected components of p_i, i. e., $p_{i,1}, \dots p_{i,k}$ such that $p_i =_G p_{i,1} \dots p_{i,k}$, each $p_{i,j}$ is connected and for each $j \neq \ell$ it holds that $p_{i,\ell}$ commutes with $p_{i,j}$. Observe that $k \leq |\mathcal{L}|$. We replace the power $p_i^{x_i}$ with $p_{i,1}^{x_i} \cdots p_{i,k}^{x_i}$.

Step 3: Removing powers of a single letter. We define the alphabet $\tilde{\Gamma} = \Gamma \times \mathbb{Z}$, where (v, z) represents the letter v^z. Note that $\tilde{\Gamma}$ is the alphabet of the simple power word problem in G. We replace any power $p_i^{x_i}$ where $p_i \in G_\alpha$ for some $\alpha \in \mathcal{L}$ with the letter $[p_i^{x_i}] \in \tilde{\Gamma}$. For the next step we still assume that the input has the shape $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$, however from here on $u_i \in \tilde{\Gamma}^*$.

Step 4: Replace each letter with a normal form specific to the input. For each i we write $p_i = a_{i,1} \dots a_{i,k_i}$, where $a_{i,j} \in \Gamma$. Let $N = [a_{1,1}, a_{1,1}^{-1}, a_{1,2}, \dots a_{1,k_1}, a_{1,k_1}^{-1}, \dots, a_{n,1}, \dots a_{n,k_n}^{-1}]$ be the list of letters (and their inverses) occurring in some power. For convenience, we write $N = [n_1, \dots n_{|N|}]$. We replace each p_i with $\tilde{p}_i = \tilde{a}_{i,1} \dots \tilde{a}_{i,k_i}$, where $\tilde{a}_{i,j}$ is the first element in N equivalent to $a_{i,j}$. Note that we need to solve the word problem in the base groups G_α to compute this. After that transformation, any two letters $\tilde{a}_{i,j}$ and $\tilde{a}_{\ell,m}$ representing the same element are equal as words. Thus, the letters are in a normal form. That normal form is dependent on the input of the power word problem in G, but that is not an issue for our application. Again, we assume the input for the next step to be $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$.

Step 5: Replace each power with a power of a word in Ω. Let $\Omega \subseteq \tilde{\Gamma}^*$ be the set containing only words w with the following properties.

- $w \neq_G 1$,
- w is primitive,
- w is connected,
- w is cyclically reduced,
- there is no $\alpha \in \mathcal{L}$, such that $w \in G_\alpha$,
- all letters of w are in normal form (w. r. t. the normal form defined in step 4),
- w is a cyclic normal form,
- w is minimal w. r. t. $\leq_{\mathcal{L}}$ among its cyclic permutations and the cyclic permutations of a cyclic normal form of its inverse.

This definition fulfills the requirements of Theorem 5.7.

All requirements except for w being primitive and the last two requirements are already fulfilled after the previous four preprocessing steps. To take care of the last two requirements, for each p_i we compute a cyclic normal form and a cyclic normal form of its inverse. Then we compute the minimal element $\tilde{p}_i$ w. r. t. $\leq_{\mathcal{L}}$ among all cyclic permutations of the two cyclic normal forms. If $\tilde{p}_i$ is conjugate to the inverse of p_i we set $\tilde{x}_i = -x_i$, otherwise $\tilde{x}_i = x_i$. There are α_i and β_i with $p_i^{x_i} =_G \alpha_i \tilde{p}_i^{\tilde{x}_i} \beta_i$ such that $\alpha_i =_G \beta_i^{-1}$ and $|\alpha_i| = |\beta_i| < |\mathcal{L}| \cdot |\tilde{p}|$. Again, we view α_i as part of u_{i-1} and β_i as part of u_i, thus we can assume that the input for the next reduction step, that is computing the shortened word, has the shape $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$.

Now that each p_i is a cyclic normal form, it is easy to check whether a given p_i is primitive, and replace it with a primitive factor if necessary. As p_i is a cyclic normal form we know that if $p_i =_G q^r$, then $p_i = \mathrm{nf}(q)^r$ (compare proof of Theorem 4.12). Thus, we only need to check for periods in p_i.

7.1.1 The non-uniform case

Lemma 7.5 *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. Then the preprocessing as described above can be done in* uAC^0 *with oracle gates for the word problem in G and for the word problem in the base groups G_α.* □

Proof The cyclically reduced conjugate in step one can be computed in uAC^0 with oracle gate for the word problem in G by Lemma 4.4.

To compute connected components of a power $p_i^{x_i}$ we define the predicate $\mathrm{con}_{p_i}(j, \ell)$ for $j, \ell \in \mathcal{L}$ which is true if j and ℓ are connected in $D(p_i)$, the dependence graph restricted to the letters in p_i. The two vertices are connected if there is an undirected path from j to ℓ of length at most $|\mathcal{L}| - 1$. The following equation is equivalent to $\mathrm{con}_{p_i}(j, \ell)$.

$$\exists i_1, \dots i_{|\mathcal{L}|-1} : i_1 = j \wedge i_{|\mathcal{L}|-1} = \ell \wedge (i_1, i_2) \notin I \wedge \dots \wedge (i_{|\mathcal{L}|-2}, i_{|\mathcal{L}|-1}) \notin I$$

Furthermore we define the predicate $\mathrm{smallest}_{p_i}(j)$ which is true if $j \in \mathcal{L}$ is the smallest member of $\mathcal{L}$ in the connected component of j. The following equation is equivalent to $\mathrm{smallest}_{p_i}(j)$.

$$\forall \ell \in \mathcal{L} : \ell \geq j \vee \neg\, \mathrm{con}_{p_i}(j, \ell)$$

We define the projections $\pi_{p_i,j}$ for $p_i = a_{i,1} \dots a_{i,k}$ by

$$a_{i,\ell} \mapsto \begin{cases} a_{i,\ell} & \text{if } \mathrm{con}_{p_i}(j, \mathrm{alph}(a_{i,\ell})) \wedge \mathrm{smallest}_{p_i}(j), \\ 1 & \text{otherwise.} \end{cases}$$

Observe that $p_i =_G \pi_{p_i,1} \dots \pi_{p_i,|\mathcal{L}|}$ and each $\pi_{p_i,j}$ is connected.

Replacing each power of a single letter with the corresponding letter from the input alphabet of the power word problem of the base group can obviously be done in uAC^0.

To compute our normal form we define the predicate $\mathrm{nfletter}_N(n_i, a)$ to decide whether $n_i \in N$ is the normal form of a. The predicate is equivalent to

$$\mathrm{alph}(n_i) = \mathrm{alph}(a) \wedge n_i =_{G_{\mathrm{alph}(a)}} a \wedge \forall n_j \in N : \mathrm{alph}(n_j) \neq \mathrm{alph}(a) \vee n_j \neq_{G_{\mathrm{alph}(a)}} n_i \vee j \geq i.$$

We define the following mapping of a letter to its normal form.

$$\pi_{\mathrm{nfletter}} : a \mapsto n_i \quad \text{where } \mathrm{nfletter}_N(n_i, a).$$

To compute a representative for p_i from Ω, we first compute a cyclic normal form conjugate to p_i and a cyclic normal form conjugate to p_i^{-1}. By Theorem 4.12, we can compute a cyclic normal form in uAC^0 with oracle gates for the word problem in G and for the normal form problem in each G_α. Using the mapping π_{nfletter} presented above we can compute our normal form in uAC^0. Computing cyclic permutations, selecting the smallest, checking for periods and replacing each power with a primitive factor is obviously in uAC^0. Hence, we have shown, that the preprocessing can be done in uAC^0 using oracle gates for the word problem in G and for the word problem in each G_α. ■

7.1.2 The uniform case

Lemma 7.6 *Let C be a non-trivial class of f.g. groups. Given $w \in G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$, where $G_1, \dots G_n$ and I are part of the input, the preprocessing as described above can be done in* uAC^0 *with oracle gates for* $\mathrm{C}_=\mathrm{L}$ *and for the uniform word problem of* $\mathrm{GP}\,C$. □

Proof The cyclically reduced conjugate in step one can be computed in uAC^0 with oracle gate for the uniform word problem of $\mathrm{GP}\,C$ by Lemma 4.4.

To compute connected components of a power $p_i^{x_i}$ we define the predicate $\mathrm{con}_{p_i}(j, \ell)$ for $j, \ell \in \mathcal{L}$ which is true if j and ℓ are connected in $D(p_i)$ (the dependence graph restricted to the letters in p_i). The two vertices are connected if there is an undirected path from j to ℓ of length at most $|\mathcal{L}| - 1$. Computing the con_{p_i} predicate in the uniform case requires solving the undirected path connectivity problem. By [Rei08] this can be done in LOGSPACE and thus in $\mathrm{C}_=\mathrm{L}$.

Furthermore, we define the predicate $\mathrm{smallest}_{p_i}(j)$ which is true if $j \in \mathcal{L}$ is the smallest member of $\mathcal{L}$ in the connected component of j. The following equation is equivalent to $\mathrm{smallest}_{p_i}(j)$.

$$\forall \ell \in \mathcal{L} : \ell \geq j \vee \neg\, \mathrm{con}_{p_i}(j, \ell)$$

We define the projections $\pi_{p_i,j}$ for $p_i = a_{i,1} \dots a_{i,k}$ by

$$a_{i,\ell} \mapsto \begin{cases} a_{i,\ell} & \text{if } \mathrm{con}_{p_i}(j, \mathrm{alph}(a_{i,\ell})) \wedge \mathrm{smallest}_{p_i}(j), \\ 1 & \text{otherwise.} \end{cases}$$

Observe that $p_i =_G \pi_{p_i,1} \dots \pi_{p_i,|\mathcal{L}|}$ and each $\pi_{p_i,j}$ is connected.

Replacing each power of a single letter with the corresponding letter from the input alphabet of the power word problem of the base group can obviously be done in uAC^0.

To compute our normal form we define the predicate $\mathrm{nfletter}_N(n_i, a)$ to decide whether $n_i \in N$ is the normal form of a. The predicate is equivalent to

$$\mathrm{alph}(n_i) = \mathrm{alph}(a) \wedge n_i =_{G_{\mathrm{alph}(a)}} a \wedge \forall n_j \in N : \mathrm{alph}(n_j) \neq \mathrm{alph}(a) \vee n_j \neq_{G_{\mathrm{alph}(a)}} n_i \vee j \geq i.$$

We define the following mapping of a letter to its normal form.

$$\pi_{\mathrm{nfletter}} : a \mapsto n_i \quad \text{where } \mathrm{nfletter}_N(n_i, a).$$

To compute a representative for p_i from Ω, we first compute a cyclic normal form conjugate to p_i and a cyclic normal form conjugate to p_i^{-1}. By Theorem 4.12, we can compute a cyclic normal form in uAC^0 with oracle gates for the uniform word problem in $\mathrm{C}_=\mathrm{L}^{\mathrm{nf}_C}$ and $\mathrm{GP}\,C$. Using the mapping π_{nfletter} presented above as oracle for nf_C we can use the oracle for $\mathrm{C}_=\mathrm{L}$ instead of $\mathrm{C}_=\mathrm{L}^{\mathrm{nf}_C}$. Computing cyclic permutations, selecting the smallest, checking for periods and replacing each power with a power of a primitive factor can be done in uAC^0. Hence, we have shown, that the preprocessing can be done in uAC^0 using oracle gates for $\mathrm{C}_=\mathrm{L}$ and for the uniform word problem of $\mathrm{GP}\,C$. ■

7.2 Symbolic Rewriting System

Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. Let S be the rewriting system for G defined by the following relations, where $a, b \in \Gamma$ and $\mathbf{u} \in \Gamma^*$.

$$a\mathbf{u}b \to [ab]\mathbf{u} \qquad \text{if } \mathrm{alph}(a) = \mathrm{alph}(b) \text{ and } (a, \mathbf{u}) \in I$$

For $p \in \Omega$ we define

$$\Delta_p = \left\{ \beta_i p^x \alpha \;\middle|\; \begin{array}{l} x \in \mathbb{Z}, \\ \alpha \in \mathrm{IRR}(S) \text{ is a prefix of } p^{\sigma \operatorname{sgn} x},\ p \text{ is no prefix of } \alpha, \\ \beta \in \mathrm{IRR}(S) \text{ is a suffix of } p^{\sigma \operatorname{sgn} x} \text{ and } p \text{ is no suffix of } \beta. \end{array} \right\}$$

Recall, that we defined the input alphabet of the simple power word problem $\tilde{\Gamma} = \Gamma \times \mathbb{Z}$. A letter $(v, z) \in \tilde{\Gamma}$ is interpreted as v^z. In $\tilde{\Gamma}^*$ we can have powers of individual letters (which are words in the base groups), but not powers of words containing letters from multiple base groups. We define the alphabet Δ by $\Delta' = \bigcup_{p\in\Omega} \Delta_p$, $\Delta'' = \tilde{\Gamma}$ and $\Delta = \Delta' \cup \Delta''$. In the following, let $\sigma = |\mathcal{L}|$.

Lemma 7.7 *Given $\beta p^x \alpha \in \Delta'$ it holds that $|\alpha| < (\sigma - 1)|p|$ and $|\beta| < (\sigma - 1)|p|$.* □

Proof By Lemma 2.3 we can write $\alpha = p^k w_s \cdots w_1$ with $s < \sigma$ where each w_i is a prefix of p. As p is not a prefix of α we have $k = 0$. Regarding the length of α we obtain $|\alpha| = \sum_{i=1}^{s} |w_i| < \sum_{i=1}^{s} |p| = s|p| \le (\sigma - 1)|p|$. The bound on the length of β follows by symmetry. ■

The rewriting system T over Δ^* is defined by the following rules. In the following $\beta p^x \alpha, \delta p^y \gamma, \delta q^y \gamma \in \Delta'$; $a, b \in \Delta''$; $r \in \Delta''^*$; $d, e \in \mathbb{Z}$; $0 \le k \le \sigma$; $a_i \in \Delta''$ and $\mathbf{u} \in \Delta^*$.

(7.1) $\beta p^x \alpha \mathbf{u} \delta p^y \gamma \to \beta p^{x+y+d} \gamma \mathbf{u}$ if $\alpha\delta \overset{*}{\underset{S}{\Longrightarrow}} p^d$ and $(p, \pi(\mathbf{u})) \in I$

(7.2) $\beta p^x \alpha \mathbf{u} \delta p^y \gamma \to \beta p^{x-d} \alpha' a_1 \cdots a_k \mathbf{u} \delta' p^{y-e} \gamma$ if $\nexists c \in \mathbb{Z} : \alpha\delta \overset{*}{\underset{S}{\Longrightarrow}} p^c$ or $(p, \pi(\mathbf{u})) \notin I$, $\beta p^x \alpha \mathbf{u} \in \mathrm{IRR}(S), \mathbf{u}\delta p^y \gamma \in \mathrm{IRR}(S)$ and $p^x \alpha \mathbf{u} \delta p^y \overset{*}{\underset{S}{\Longrightarrow}} p^{x-d} \alpha' a_1 \cdots a_k \mathbf{u} \delta' p^{y-e} \in \mathrm{IRR}(S)$

(7.3) $\beta p^x \alpha \mathbf{u} \delta q^y \gamma \to \beta p^{x-d} \alpha' a_1 \cdots a_k \mathbf{u} \delta' q^{y-e} \gamma$ if $p \neq_G q$ and $p^x \alpha \mathbf{u} \delta q^y \overset{*}{\underset{S}{\Longrightarrow}} p^{x-d} \alpha' a_1 \cdots a_k \mathbf{u} \delta' q^{y-e} \in \mathrm{IRR}(S)$

(7.4) $\beta p^x \alpha \to r$ if $\beta\alpha \overset{*}{\underset{S}{\Longrightarrow}} r \in \mathrm{IRR}(S)$ and $x = 0$

(7.5) $a\mathbf{u}\beta p^x \alpha \to a'\mathbf{u}\beta' p^{x-d} \alpha$ if $(a, \mathbf{u}) \in I$ and $a\mathbf{u}\beta p^x \overset{*}{\underset{S}{\Longrightarrow}} a'\mathbf{u}\beta' p^{x-d} \in \mathrm{IRR}(S)$

(7.6) $\beta p^x \alpha \mathbf{u} a \to \beta p^{x-d} \alpha' \mathbf{u} a'$ if $(a, \mathbf{u}) \in I$ and $p^x \alpha \mathbf{u} a \overset{*}{\underset{S}{\Longrightarrow}} p^{x-d} \alpha' \mathbf{u} a' \in \mathrm{IRR}(S)$

(7.7) $a\mathbf{u}b \to r\mathbf{u}$ if $(a, \mathbf{u}) \in I, \mathrm{alph}(a) = \mathrm{alph}(b)$ and $r = [ab]$

Lemma 7.8 *For $u, v \in \Delta^*$ it holds that*

(i) $\pi(\mathrm{IRR}(T)) \subseteq \mathrm{IRR}(S)$,

(ii) $u \overset{*}{\underset{T}{\Longrightarrow}} v$ *implies* $\pi(u) \overset{*}{\underset{S}{\Longrightarrow}} \pi(v)$,

(iii) $\pi(u) =_G 1$ *if and only if* $u \overset{*}{\underset{T}{\Longrightarrow}} 1$. □

Proof Assume we have an element $t \in \mathrm{IRR}(T)$ with $\pi(t) \notin \mathrm{IRR}(S)$. Then there is a factor $a\mathbf{u}b$ of t, where $a, b \in \tilde{\Gamma}$, $ab \overset{*}{\underset{S}{\Longrightarrow}} [ab]$, $\mathbf{u} \in \Gamma^*$ and $(a, \mathbf{u}) \in I$. As the letters of t are irreducible over S the preimages of a and b must be located in different letters of t. Let t_i and t_j be the letters of t that contain the preimages of a and b. Note that a is a suffix of t_i and b is a prefix of t_j. All the letters in between hold parts of the preimage of $\mathbf{u}$, thus commute with a. It follows that one of the rules of T can be applied contradicting $t \in \mathrm{IRR}(T)$. Thus $\pi(\mathrm{IRR}(T)) \subseteq \mathrm{IRR}(S)$.

For (ii) observe that the rules of T only allows such reductions that are also allowed in S.

(iii) follows from (i) and (ii). If $u \overset{*}{\underset{T}{\Longrightarrow}} 1$ then $\pi(u) \overset{*}{\underset{S}{\Longrightarrow}} 1$ by (ii). If $u \overset{*}{\underset{T}{\Longrightarrow}} v \in \mathrm{IRR}(T)$ with $v \neq 1$ then $\pi(u) \overset{*}{\underset{S}{\Longrightarrow}} \pi(v)$ by (ii) and $\pi(v) \in \mathrm{IRR}(S)$ by (i). ■

Lemma 7.9 *The following length bounds hold:*

- *Rule* (7.2)*:* $|d| \leq 2\sigma$ *and* $|e| \leq 2\sigma$
- *Rule* (7.3)*:* $|d| \leq 4(\sigma+1)|q|_\Gamma$ *and* $|e| \leq 4(\sigma+1)|p|_\Gamma$
- *Rule* (7.4)*:* $|r|_\Gamma < 2(\sigma-1)|p|_\Gamma$
- *Rules* (7.5) *and* (7.6)*:* $|d| \leq 1$ □

Proof When applying rule (7.2) we distinguish two cases. If $(p, \pi(\mathbf{u})) \in I$ then $\alpha\beta$ is no power of p. Most importantly we have $\alpha\beta \neq_G 1$. Let $p_1, p_2 \in \mathrm{IRR}(S)$ be cyclically reduced conjugates of p such that $p_1 =_G \alpha^{-1}p\alpha$ and $p_2 =_G \delta p \delta^{-1}$. As p is primitive and $\alpha \neq_G \delta^{-1}$, we have $p_1 \neq_G p_2$.

Let α'' be the suffix of $p^x\alpha$ and δ'' the prefix of δp^y such that $\alpha''\delta'' \overset{*}{\underset{S}{\Longrightarrow}} a_1 \dots a_k$. Then α'' is a suffix of p_1^∞ and δ'' is a prefix of p_2^∞. From Theorem 5.6 it follows $|\alpha''| \leq (\sigma+1)|p|_\Gamma$. By Lemma 7.7, $|\alpha'|_\Gamma < (\sigma-1)|p|_\Gamma$. We write $p^d\alpha = \alpha'\alpha''$. It follows that $|p^d|_\Gamma + |\alpha|_\Gamma = |\alpha'|_\Gamma + |\alpha''|_\Gamma$, and thus $d|p|_\Gamma \leq |\alpha'| + |\alpha''| \leq (\sigma-1)|p|_\Gamma + (\sigma+1)|p|_\Gamma$. Solving for d we obtain $d \leq 2\sigma$. The bound on $|e|$ follows by symmetry.

If $(p, \pi(\mathbf{u})) \notin I$ observe that if there is some prefix δ'' of δp^y and some suffix α'' of $p^x\alpha$ such that $\alpha''\delta'' \overset{*}{\underset{S}{\Longrightarrow}} a_1 \dots a_k$ then δ'' must commute with $\mathbf{u}$. Thus, $p^{\mathrm{sgn}(y)}$ cannot be a factor of δ'' and by Lemma 2.3 δ'' must be a prefix of $\delta p^{(\sigma-1)\,\mathrm{sgn}(y)}$. Thus $|e| \leq \sigma - 1 < 2\sigma$. The bound on $|e|$ follows by symmetry.

When applying rule (7.3) we have a suffix α'' of $p^x\alpha$ and a prefix δ'' of δq^y such that $\alpha''\delta'' \overset{*}{\underset{S}{\Longrightarrow}} a_1 \dots a_k$. Thus, we can write $p^d\alpha = \alpha'\alpha''$ and $\delta q^e = \delta''\delta'$. Let α''' be a suffix of α'' and δ''' be a prefix of δ'' such that $\alpha'''\delta''' \overset{*}{\underset{S}{\Longrightarrow}} 1$ and the length of α''' and δ''' is maximal. α''' is a factor of $\left(p^{\mathrm{sgn}(x)}\right)^\infty$ and δ''' is a factor of $\left(q^{\mathrm{sgn}(y)}\right)^\infty$. It holds that $|\alpha'''|_\Gamma = |\delta'''|_\Gamma \leq 2(\sigma+1)(|p|_\Gamma + |q|_\Gamma)$. Otherwise, we would have $p = q$ by Theorem 5.7 contradicting $p \neq q$. By Lemma 7.7, $|\alpha'|_\Gamma <$

$(\sigma-1)|p|_\Gamma$ and $|\delta'|_\Gamma < (\sigma-1)|q|_\Gamma$. We also have $|\alpha''| \leq |\alpha'''| + \sigma$ and $|\delta''| \leq |\delta'''| + \sigma$. It follows $|p^d\alpha|_\Gamma = |\alpha'\alpha''|_\Gamma < 2(\sigma+1)(|p|_\Gamma+|q|_\Gamma)+\sigma+(\sigma-1)|p|_\Gamma < 4(\sigma+1)(|p|_\Gamma+|q|_\Gamma) \leq 4(\sigma+1)|p|_\Gamma|q|_\Gamma$. Combining this with $|d| \cdot |p|_\Gamma = |p^d|_\Gamma \geq |p^d\alpha|_\Gamma$ and solving for $|d|$ we obtain $|d| < 4(\sigma+1)|q|_\Gamma$. By symmetry, it follows that $|e| < 4(\sigma+1)|p|_\Gamma$.

In Rule (7.4) we have $|r|_\Gamma \leq |\alpha\beta|_\Gamma$. By Lemma 7.7, $|\alpha|_\Gamma < (\sigma-1)|p|_\Gamma$ and $|\beta|_\Gamma < (\sigma-1)|p|_\Gamma$. Therefore, $|r|_\Gamma < 2(\sigma-1)|p|_\Gamma$.

When applying rule (7.5) there is a single letter prefix b of βp^x with $\mathrm{alph}(a) = \mathrm{alph}(b)$. Either b is a prefix of β in which case $d = 0$ or if it is not, then b must be a prefix of $p^{\operatorname{sgn} x}$ in which case $|d| = 1$. The same bound on rule (7.6) follows by symmetry. ∎

Definition 7.10 Let $w = w_1 \dots w_n \in \Delta^*$. We define

- $\mu(w) = \max\{|p|_\Gamma \mid w_i = \beta p^x \alpha \in \Delta'\}$,
- $\lambda(w) = |w|_{\Delta''} + \sum_{w_i=\beta p^x\alpha\in\Delta'} |p|_\Gamma$ and
- $\pi(w) = \pi(w_1)\dots\pi(w_n)$, $\pi(a) = a$ for $a \in \Delta''$ and $\pi(\beta p^x\alpha) = \beta p^x\alpha$ for $\beta p^x\alpha \in \Delta'$. □

Lemma 7.11 *The rewriting system T, applied to a word $w \in \Delta^*$, has the following properties.*

1. *Rules* (7.4) *and* (7.1) *can be applied at most $|w|_{\Delta'}$ times in total.*
2. *Rules* (7.2) *and* (7.3) *can be applied at most $2\sigma|w|_{\Delta'}$ times.*
3. *The number of applications of rules* (7.5), (7.6) *and* (7.7) *is at most $2\sigma^3\lambda(w) + 2\sigma^2\lambda(w)^2$.* □

Proof

1. For an application $w_1 \underset{T}{\Longrightarrow} w_2$ of rule (7.4) or (7.1) it holds that $|w_1|_{\Delta'} > |w_2|_{\Delta'}$. Thus there can be at most $|w|_{\Delta'}$ applications of that rule.
2. When looking at the number of times the rules (7.2) and (7.3) can be applied we only need to consider the letters from Δ' in w. The rules can be applied only once to each pair of letters from Δ'. Furthermore, each letter $\beta p^x\alpha \in \Delta'$ can cancel with at most σ other letters to its right (and at most σ other letters to its left). Hence, up to $\sigma|w|_{\Delta'}$ applications are possible initially (or may be unblocked by applications of rules (7.5), (7.6) and (7.7)). Each removal of a letter from Δ' by rule (7.4) enables up to σ additional applications of rules (7.2) and (7.3). In total the two rules can be applied at most $2\sigma|w|_{\Delta'}$ times.
3. For a transition $w_1 \underset{T}{\Longrightarrow} w_2$ the following holds.
   - $|w_2|_{\Delta''} = |w_1|_{\Delta''}$ if the applied rule is (7.1).
   - Each application of rules (7.2) or (7.3) increases $|\cdot|_{\Delta''}$ by up to σ.
   - Each application of rule (7.4) increases $|\cdot|_{\Delta''}$ by up to $2(\sigma-1)\lambda(w)$, as that rule adds several letters from Δ'' to w_2, more precisely it adds a prefix α of $p^{(\sigma-1)\operatorname{sgn}(x)}$ and a suffix β of $p^{(\sigma-1)\operatorname{sgn}(x)}$. We compute the bound $|\alpha|_{\Delta''} + |\beta|_{\Delta''} \leq 2(\sigma-1)|p|_{\Delta''}$.

We first look at length-reducing applications of rules (7.5), (7.6) and (7.7), where the length $|\cdot|_{\Delta''}$ is reduced. Initially, there are $|w|_{\Delta''}$ letters from Δ''. Rules (7.2), (7.3) and (7.4) in total create up to $2\sigma^2|w|_{\Delta'} + 2(\sigma-1)\lambda(w)|w|_{\Delta'}$ additional letters. Thus we can bound the number of length-decreasing applications by $(2\sigma^2 + 2(\sigma-1)\lambda(w) + 1)\lambda(w)$.

To bound the total number of applications of rules (7.5), (7.6) and (7.7), we look the number of possible applications. Up to $2|w|_{\Delta''}$ such applications are possible initially. Rules (7.2), (7.3) and (7.4) in total create up to $2\sigma^2|w|_{\Delta'} + 2(\sigma-1)\lambda(w)|w|_{\Delta'}$ additional letters from Δ'', thus allowing up to twice that many additional applications of rules (7.5), (7.6) and (7.7). Each length-decreasing application of those rules allows up $\sigma - 1$ additional applications. Therefore, those rules can be applied at most $2|w|_{\Delta''} + 2\sigma^2|w|_{\Delta'} + 2(\sigma-1)\lambda(w)|w|_{\Delta'} + (2\sigma^2 + 2(\sigma-1)\lambda(w) + 1)\lambda(w)(\sigma-1) \leq 2\sigma^3\lambda(w) + 2\sigma^2\lambda(w)^2$ times. ∎

Corollary 7.12 *If* $w \overset{*}{\underset{T}{\Longrightarrow}} v$*, then* $w \overset{\leq k}{\underset{T}{\Longrightarrow}} v$ *with* $k = 3\sigma^3\lambda(w) + 3\sigma^2\lambda(w)^2$. □

Proof Each rule can only be applied finitely many times. Adding up the bounds from Lemma 7.11 we obtain a bound of $(2\sigma+1)|w|_{\Delta'} + 2\sigma^3\lambda(w) + 2\sigma^2\lambda(w)^2 \leq 3\sigma^3\lambda(w) + 3\sigma^2\lambda(w)^2$. ∎

Definition 7.13 Let $w \in \Delta^*$ and $p \in \Omega$. We write $w = u_0\beta_1 p^{y_1}\alpha_1 u_1 \dots \beta_m p^{y_m}\alpha_m u_m$ with $u_i \in (\Delta \setminus \Delta_p)^*$. We define

$$\eta_p = \sum_{j=1}^{m} y_j,$$

$$\eta_p^i = \sum_{j=1}^{i} y_j$$

□

Lemma 7.14 *Let* $u, v \in \Delta^*$ *and* $u \overset{}{\underset{T}{\Longrightarrow}} v$*. Given a prefix* v' *of* v *there is a prefix* u' *of* u *such that for all* $p \in \Omega$

$$|\eta_p(u') - \eta_p(v')| \leq 4(\sigma+1)\mu(u).$$

If the applied rule is neither (7.1) *nor* (7.4)*, then for all* $p \in \Omega$ *and* $0 \leq i \leq m$

$$|\eta_p^i(u) - \eta_p^i(v)| \leq 4(\sigma+1)\mu(u).$$ □

Proof We can assume, that v' contains a letter from Δ_p, otherwise the Lemma is trivial. Moreover, we can assume that $v \in \Delta^*\Delta_p$, as appending or removing letters from $\Delta \setminus \Delta_p$ does not change η_p.

Let the applied rule be $l \to r \in T$. To proof the first part of the lemma we distinguish three cases.

(i) The right-hand side of the rule is not in v'. In that case we choose $u' = v'$.

(ii) The right-hand side of the rule is completely in v'. We write $v' = \alpha r \beta$. We choose $u' = \alpha l \beta$. The Lemma follows with the bounds from Lemma 7.9.

(iii) The right-hand side r of the rule overlaps with v'. It follows that $r = \beta p^y \alpha \mathbf{u}$ and consequently we can write $l = \beta' p^x \alpha' \mathbf{u}'$. We write $v' = v'' \beta p^y \alpha$. If rule (7.1) has been applied we choose $u' = v'' l$. Otherwise, one of rules (7.2), (7.3) or (7.6) has been applied. We choose $u' = v'' \beta' p^x \alpha'$. The Lemma follows with the bounds from Lemma 7.9.

To see that the second statement of the lemma is true, observe that in case the applied rule is neither (7.1) nor (7.4), there is a one-to-one correspondence between letters from Δ_p in u and v.■

7.3 The Shortened Word

In this section we describe the shortening process. Given $u \in \Delta^*$ and some $p \in \Omega$ we consider all letters from Δ_p in u. We write $u = u_0 \beta_1 p^{y_1} \alpha_1 u_1 \dots \beta_m p^{y_m} \alpha_m u_m$ with $u_i \in (\Delta \setminus \Delta_p)^*$. In the following we define a set C of intervals to be carved out of the exponents during the shortening process.

Definition 7.15 Let $C = \{[l_j, r_j] \mid 1 \le j \le k\}$ be a set of finite, non-empty, non-overlapping intervals, where $k = |C|$. We assume the intervals to be ordered, i. e., $r_j \le l_{j+1}$. We define the size of an interval $d_j = r_j - l_j + 1$.

An element $u \in \Delta^*$ is said to be **compatible** with C, if for every prefix u' of u it holds that $\eta_p(u') \notin [l_j, r_j]$ for all $1 \le j \le k$. □

Definition 7.16 Let C compatible with u. The shortened version of u is

$$\mathcal{S}_C(u) = u_0 \beta_1 p^{z_1} \alpha_1 u_1 \dots \beta_m p^{z_m} \alpha_m u_m.$$

The new exponents are defined as

$$z_i = y_i - \operatorname{sgn}(y_i) \cdot \sum_{j \in C_i} d_j,$$

where C_i is the set of intervals to be removed from y_i, defined by

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

□

Lemma 7.17 *If $u \in \mathrm{IRR}(T)$ then $\mathcal{S}_C(u) \in \mathrm{IRR}(T)$.* □

Proof We prove the lemma by showing that $\operatorname{sgn}(y_i) = \operatorname{sgn}(z_i)$ and $z_i \neq 0$. As the intervals in C are ordered, there are α and β such that C_i contains all indices from α to β. In case $y_i > 0$ we have

$$\begin{aligned} z_i &= y_i - \sum_{j \in C_i} d_j \\ &= y_i - \sum_{j=\alpha}^{\beta} d_j \\ &= y_i - \sum_{j=\alpha}^{\beta} (r_j - l_j + 1) \\ &\leq y_i - r_\beta - l_\alpha + 1 \\ &\leq y_i - (\eta_p^i(u) - 1) - (\eta_p^i(u) + 1) + 1 \\ &= y_i - \eta_p^i(u) - \eta_p^i(u) + 1 \\ &= y_i - y_i + 1 \\ &= 1 \end{aligned}$$

The case $y_i < 0$ follows by symmetry. ∎

Definition 7.18 We define the distance between some $\eta_p^i(\cdot)$ and the closest interval from C as

$$\operatorname{dist}_p(u, C) = \min \left\{ |\eta_p^i(u) - x| \mid 1 \leq i \leq m, x \in [l, r] \in C \right\}. \qquad \square$$

From that definition the following statement follows immediately.

Corollary 7.19 $\operatorname{dist}_p(u, C) > 0$ *if and only if* u *is compatible with* C. $\square$

We want to show, that given some requirements are fulfilled, any rewriting step that is possible on u is also possible on $\mathcal{S}_C(u)$.

Lemma 7.20 *If* $\operatorname{dist}_p(u, C) \geq 4(\sigma + 1)\mu(u)$ *and* $u \underset{T}{\Longrightarrow} v$*, then* $\mathcal{S}_C(u) \underset{T}{\Longrightarrow} \mathcal{S}_C(v)$. $\square$

Proof Observe that u is compatible with C. By Lemma 7.14 we have $\operatorname{dist}(v, C) > 0$ and thus v is compatible with C. It follows that $\mathcal{S}_C(u)$ and $\mathcal{S}_C(v)$ are well-defined.

To prove the lemma we compare the shortened version of u and v and show that a rule from T can be applied.

We distinguish which rule from T has been applied to u.

- If rule (7.2), (7.3), (7.5), (7.6) or (7.7) has been applied, the shortening process has the same effect on u and v, i. e., $C_i(u) = C_i(v)$. The same rule that has been applied to u to obtain v can also be applied to $\mathcal{S}_C(u)$ to obtain $\mathcal{S}_C(v)$.
- If rule (7.4) is applied, then $C_\ell(v) = C_\ell(u)$ for $\ell < i$ and $C_\ell(v) = C_{\ell+1}(u)$ for $\ell \geq i$. We also know $y_i = 0$, which is not altered by the shortening process, i. e., $C_i(u) = \emptyset$. Thus, the same rule can be applied to $\mathcal{S}_C(u)$ to obtain $\mathcal{S}_C(v)$.

- Assume rule (7.1) has been applied.

  Let

  $$u = u_0\beta_1 p^{y_1}\alpha_1 u_1 \dots \beta_i p^{y_i}\alpha_i u_i \beta_{i+1} p^{y_{i+1}}\alpha_{i+1}u_{i+1}\dots\beta_m p^{y_m}\alpha_m u_m.$$

  The result of applying the rule is

  $$v = u_0\beta_1 p^{y_1}\alpha_1 u_1 \dots \beta_i p^{y_i+y_{i+1}+d}\alpha_{i+1} u_i u_{i+1}\dots\beta_m p^{y_m}\alpha_m u_m$$

  On powers not modified by the rule the shortening process behaves the same on u and v, i. e., $C_\ell(v) = C_\ell(u)$ for $\ell < i$ and $C_\ell(v) = C_{\ell+1}(u)$ for $\ell > i$. The result of the shortening process on v is

  $$\mathcal{S}_C(v) = u_0\beta_1 p^{z_1}\alpha_1 u_1 \dots \beta_i p^{\tilde{z}_i}\alpha_{i+1} u_i u_{i+1}\dots\beta_m p^{z_m}\alpha_m u_m,$$

  where $\tilde{z}_i = y_i + y_{i+1} + d - \operatorname{sgn}(y_i + y_{i+1} + d)\cdot\sum_{\ell\in C_i(v)} d_\ell$.

  When applying rule (7.1) to the corresponding letters of $\mathcal{S}_C(u)$ we obtain

  $$\hat{v} = u_0\beta_1 p^{z_1}\alpha_1 u_1 \dots \beta_i p^{z_i+z_{i+1}+d}\alpha_{i+1} u_i u_{i+1}\dots\beta_m p^{z_m}\alpha_m u_m,$$

  We need to show that $\tilde{z}_i = z_i + z_{i+1} + d$, i. e., $z_i + z_{i+1} = y_i + y_{i+1} - \operatorname{sgn}(y_i + y_{i+1} + d)\cdot\sum_{\ell\in C_i(v)} d_\ell$.

  Observe that $\operatorname{dist}(u, C) > d$, thus if $\operatorname{sgn}(y_i + y_{i+1} + d) \neq \operatorname{sgn}(y_i + y_{i+1})$ we have $C_i(u) = C_{i+1}(u)$, $C_i(v) = \emptyset$ and the lemma follows. From now on we can assume $\operatorname{sgn}(y_i + y_{i+1} + d) = \operatorname{sgn}(y_i + y_{i+1})$.

  First, consider the case that y_i and y_{i+1} have the same sign. In that case $C_i(v) = C_i(u)\cup C_{i+1}(u)$ we have

  $$\begin{aligned} z_i + z_{i+1} &= y_i - \operatorname{sgn}(y_i)\cdot\sum_{\ell\in C_i(u)} d_\ell + y_{i+1} - \operatorname{sgn}(y_{i+1})\cdot\sum_{\ell\in C_{i+1}(u)} d_\ell \\ &= y_i + y_{i+1} - \operatorname{sgn}(y_i + y_{i+1})\cdot\sum_{\ell\in C_i(v)} d_\ell \end{aligned}$$

  Second, we look at the case where y_i and y_{i+1} have opposite sign. We assume $|y_i| > |y_{i+1}|$. The other case is symmetric. Now $C_i(v) = C_i(u) \setminus C_{i+1}(u)$.

  $$\begin{aligned} z_i + z_{i+1} &= y_i - \operatorname{sgn}(y_i)\cdot\sum_{\ell\in C_i(u)} d_\ell + y_{i+1} - \operatorname{sgn}(y_{i+1})\cdot\sum_{\ell\in C_{i+1}(u)} d_\ell \\ &= y_i + y_{i+1} - \operatorname{sgn}(y_i)\left(\sum_{\ell\in C_i(u)} d_\ell - \sum_{\ell\in C_{i+1}(u)} d_\ell\right) \\ &= y_i + y_{i+1} - \operatorname{sgn}(y_i + y_{i+1})\cdot\sum_{\ell\in C_i(v)} d_\ell \end{aligned}$$

  ∎

Lemma 7.21 *If* $\operatorname{dist}_p(u, C) \geq 4k(\sigma+1)\mu(u)$ *and* $u \overset{\leq k}{\underset{T}{\Longrightarrow}} v$, *then* $\mathcal{S}_C(u) \overset{\leq k}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(v)$. □

Proof We prove the lemma by induction. If $k = 1$ then the statement follows from Lemma 7.20. If $k > 1$, then there is a $u' \in \Delta^*$ such that

$$u \underset{T}{\Longrightarrow} u' \overset{\leq k-1}{\underset{T}{\Longrightarrow}} v.$$

By Lemma 7.14 we have $\mathrm{dist}_p(u', C) \geq 4(k-1)(\sigma+1)\mu(u)$. As none of the rules of T increases $\mu(\cdot)$, it follows that $\mathrm{dist}_p(u', C) \geq 4(k-1)(\sigma+1)\mu(u')$. Therefore $\mathcal{S}_C(u') \overset{\leq k-1}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(v)$ by induction. By Lemma 7.20 we have $\mathcal{S}_C(u) \overset{\leq k-1}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(u')$. Combining those statements we conclude $\mathcal{S}_C(u) \overset{\leq k}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(v)$. ∎

Lemma 7.22 *If* $\mathrm{dist}_p(u, C) \geq 12(\sigma+1)^3\lambda(u)^2(\sigma+\lambda(u))$*, then* $\pi(u) =_G 1$ *if and only if* $\pi(\mathcal{S}_C(u)) =_G 1$. □

Proof Let $k = 3\sigma^3\lambda(w) + 3\sigma^2\lambda(w)^2$. From the precondition of the lemma we conclude $\mathrm{dist}_p(u, C) \geq 4k(\sigma+1)\lambda(u) \geq 4k(\sigma+1)\mu(u)$.

First, let $\pi(u) =_G 1$. By Lemma 6.4 that is equivalent to $u \overset{*}{\underset{T}{\Longrightarrow}} 1$. Which by Corollary 6.8 is equivalent to $u \overset{\leq k}{\underset{T}{\Longrightarrow}} 1$. By Lemma 6.17 we have $\mathcal{S}_C(u) \overset{\leq k}{\underset{T}{\Longrightarrow}} \mathcal{S}_C(1) = 1$. Applying Corollary 6.8 and Lemma 6.4 again we obtain $\pi(\mathcal{S}_C(u)) =_G 1$.

Second, assume $\pi(u) =_G v \in \mathrm{IRR}(S)$, with $v \neq_G 1$. Again, applying Corollary 6.8 and Lemma 6.4 that is equivalent to $u \overset{\leq k}{\underset{T}{\Longrightarrow}} \tilde{v}$, with $\tilde{v} \in \mathrm{IRR}(T)$, $\tilde{v} \neq_G 1$. By Lemma 6.13 $\mathcal{S}_C(v)$ is irreducible. As the shortening process does not remove any letters, but only replaces them we have $\mathcal{S}_C(\tilde{v}) \neq_G 1$. Again, applying Corollary 6.8 and Lemma 6.4, we obtain $\pi(\mathcal{S}_C(u)) =_G \mathcal{S}_C(\tilde{v}) \neq_G 1$. ∎

We continue by defining a concrete set of intervals $C^K_{u,p}$ that will be used to show that the exponents of the shortened word are bounded by a polynomial.

Let $\{c_1, \ldots c_\ell\} = \left\{\eta^i_p(u) \mid 0 \leq i \leq m\right\}$ be the ordered set of $\eta^i_p(u)$, i. e., $c_1 < \cdots < c_\ell$. We define the set of intervals

$$C^K_{u,p} = \{[c_i + K, c_{i+1} - K] \mid 1 \leq i < \ell, c_{i+1} - c_i \geq 2K\}. \tag{7.8}$$

Lemma 7.23 *Let* $K = 12(\sigma+1)^3\lambda(u)^2(\sigma+\lambda(u))$ *and* $\mathcal{S}_C(u) = u_0\beta_1 p^{z_1}\alpha_1 u_1 \ldots \beta_m p^{z_m}\alpha_m u_m$ *for some* $u \in \Delta^*$*. Then* $|z_i| \leq 24m(\sigma+1)^3\lambda(u)^2(\sigma+\lambda(u))$ *for* $1 \leq i \leq m$. □

Proof

$$
\begin{aligned}
|z_i| &= \left| y_i - \operatorname{sgn}(y_i) \cdot \sum_{j \in C_i} d_j \right| \\
&= |y_i| - \sum_{j \in C_i} d_j \\
&\overset{(i)}{=} |y_i| - \sum_{j \in C_i} \max\{0, c_{j+1} - c_j - 2K + 1\} \\
&\leq |y_i| - \sum_{j \in C_i} c_{j+1} - c_j - 2K + 1 \\
&= |y_i| - \sum_{j \in C_i} c_{j+1} - c_j + \sum_{j \in C_i} 2K - 1 \\
&= \sum_{j \in C_i} 2K - 1 \overset{(ii)}{\leq} m(2K-1) \leq m2K
\end{aligned}
$$

We used the following facts.

(i) Definition of $C^K_{u,p}$ in (7.8).

(ii) $|C_i| \leq |C^K_{u,p}| \leq m$.

The lemma follows by plugging in the formula for K. ∎

7.4 The non-uniform Case

In this section we show that the power word problem in graph groups can be solved in uAC^0 with oracles for the power word problem in the base groups G_α and for the word problem of the free group F_2. First, we reduce the power word problem to the simple power word problem, where each power is restricted to a single base group. We do this using the shortening process presented in the previous section. To solve the simple power word problem, we need to handle the remaining powers which are restricted to a single base group. We do that by using the oracles for the power word problem in the base groups.

Lemma 7.24 *Let G be a graph product. The power word problem* $\mathrm{PowWP}(G)$ *is* uAC^0*-Turing reducible to the simple power word problem* $\mathrm{SPowWP}(G)$ *and the word problem of the free group* $\mathrm{WP}(F_2)$. □

Proof We begin with the preprocessing described in Section 7.1. By Lemma 7.5 the preprocessing can be done in uAC^0 with oracles for the word problem in G and for the power word problem in each G_α. We can use the oracle for the simple power word problem $\mathrm{SPowWP}(G)$ in G to solve the word problem in G and the power word problem in each G_α. Following the preprocessing we have a word $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$. We proceed with the shortening procedure for each $p \in \{p_i \mid 1 \leq i \leq n\}$. The shortening procedure can be computed in parallel for each p. The computation of the shortened word requires iterated additions which is in uTC^0, and therefore can be done in uAC^0 with oracle

gates for $\mathrm{WP}(F_2)$. By Lemma 7.23 the exponents of the shortened word are bounded by a polynomial. Thus, the shortened word can be represented as a word $\hat{w}$ of polynomial length over the alphabet $(\Gamma \times \mathbb{Z})$. We can solve the power word problem of w by solving the simple power word problem of $\hat{w}$. ■

Before solving the simple power word problem in G we introduce lemmata to help us achieve this goal. The following lemma is due to Jonathan Kausch [Kau17].

Lemma 7.25 *[Kau17, Lemma 5.4.4] Let B be a f.g. group, $\mathcal{R}$ a (possibly infinite) set, let $G = *_{v\in\mathcal{R}} B^{(v)}$ with $B^{(v)} \simeq B$ and $1 \in \mathcal{R}$ a distinguished element. Then, it holds that $G \simeq F(X) \rtimes B$, where $F(X)$ is a free group with basis*

$$X = \{g^{(v)} \mid 1 \neq g \in B, 1 \neq v \in \mathcal{R}\}.$$ □

Let B be a f.g. group, $\mathcal{R}$ be a finite list. We begin by looking at a specific free product. Jonathan Kausch [Kau17, Lemma 5.4.5] has shown that for the free product $G = *_{v\in\mathcal{R}} B^{(v)}$, the word problem can be solved in uAC^0 with oracle gates for $\mathrm{WP}(B)$ and $\mathrm{WP}(F_2)$. We show a similar result for the simple word problem. Our proof is mostly identical to the one presented in [Kau17], with only a few changes to account for the different encoding of the input.

Lemma 7.26 *Let B be a f.g. group, $\mathcal{R}$ be a finite list. Let $G \simeq *_{v\in\mathcal{R}} B^{(v)}$. Let the input be $w = \left(w_1^{(v_1)}\right)^{x_1} \dots \left(w_n^{(v_n)}\right)^{x_n} \in (B \times \mathcal{R} \times \mathbb{Z})^*$, where the exponents x_i are encoded as binary numbers. Then the problem $w \in G$ can be decided in uAC^0 with oracles for the power word problem in B and for the word problem of the free group F_2.* □

Proof By Lemma 7.25 we have $G \simeq F(X) \rtimes B$. The set X is given by

$$X = \{g^{(v)} \mid 1 \neq g \in B, 1 \neq v \in \mathcal{R}\}.$$

Let $\varphi : G \to B$ be the homomorphism defined by $\varphi(b^{(v)}) = b$. We can assume $\varphi(w) = w_1^{x_1} \dots w_n^{x_n} =_B 1$ as otherwise $w \neq_G 1$. Now our aim is to write w as a member of the kernel of φ, which is $F(X)$.

Let $g_i = w_1^{x_1} \dots w_i^{x_i} \in (B \times \mathbb{Z})^*$. Observe that we can construct the g_i in uAC^0. We have

$$w =_G g_1^{(v_1)} \cdot \prod_{i=2}^{n} \left(g_{i-1}^{(v_{i-1})}\right)^{-1} \cdot g_i^{(v_i)}.$$

Using the fact that $g_n = \varphi(w) =_B 1$ we can rewrite w as part of the kernel.

$$w =_G \prod_{i=1}^{n-1} g_i^{(v_i)} \cdot \left(g_i^{(v_{i+1})}\right)^{-1}.$$

Next we define a finite subset $Y \subseteq X$, such that $w \in F(Y) \leq F(X)$. To achieve this we set

$$Y = \{g_i^{(v_i)}, g_i^{(v_{i+1})} \mid 1 \leq i \leq n-1\}.$$

From this definition it follows that $|Y| \leq 2(n-1)$. Note that two generators $g_i^v, g_j^u \in Y$ are equal if and only if $v = u$ and $g_i =_B g_j$. The following circuit can decide whether two generators are equal using the oracle for the power word problem in B.

$$Eq(v, i, u, j) \equiv \left(v = u \wedge g_i g_j^{-1} \in \mathrm{PowWP}(B)\right)$$

As a last step we simplify the basis by mapping it to the set $\hat{Y} = \{x_1, \dots x_{2(n-1)}\}$. We use the following map $\psi : Y \to \hat{Y}$.

$$g_i^v \mapsto \begin{cases} x_k, & \text{if } v = v_i \text{ and} \\ & k = \min\{j \leq i \mid g_i^{v_i} = g_j^{v_j}\}, \\ x_k, & \text{if } v = v_{i+1} \text{ and} \\ & k = \min\{1 \leq j \leq n \mid g_i^{v_{i+1}} = g_j^{v_j}\} \text{ exists}, \\ x_{k+n-1} & \text{if } v = v_{i+1} \text{ and} \\ & k = \min\{1 \leq j \leq n \mid g_i^{v_{i+1}} = g_j^{v_j}\} \text{ otherwise.} \end{cases}$$

The map ψ can be computed in uAC^0 with oracle gates for the power word problem in B and defines an isomorphism between $F(Y)$ and $F(\hat{Y})$. By [Kau17, Theorem 5.2.5] the uniform word problem of $\hat{w} = \psi(w)$ in $F(\hat{Y})$ can be reduced to the word problem in F_2 in uAC^0. ∎

The next step is to look at a specific amalgamated product. The following lemma is due to Jonathan Kausch [Kau17].

Lemma 7.27 *[Kau17, Lemma 5.5.2] Let $G = P *_A (B \times A)$. Given a retract $\pi : G \to P$, it holds that $G \simeq \ker \pi \rtimes P$ and*

$$\ker \pi \simeq *_{v \in P/A} B^{(v)}.$$ □

Lemma 7.28 *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ be a graph product of f.g. groups. For a subset $S \subseteq \mathcal{L}$ we define the induced subgroup $G_S = \mathrm{GP}(S, I_S; (G_\alpha)_{\alpha \in S})$, where $I_S = I \cap (S \times S)$. It holds that*

$$\mathrm{GSPowWP}(G_S, G) \in \mathsf{uAC}^0(\mathrm{SPowWP}(G)),$$

that is the generalized simple power word problem $\mathrm{GSPowWP}(G_S, G)$ can be decided in uAC^0 with an oracle for the simple power word problem in G. □

Proof Consider the projection $\pi : (\Gamma \times \mathbb{Z})^* \mapsto (\Gamma_S \times \mathbb{Z})$, with

$$\pi(a^x) = \begin{cases} a^x & \text{if } \mathrm{alph}(a) \in S, \\ 1 & \text{otherwise.} \end{cases}$$

Let $w = w_1^{x_1} \dots w_n^{x_n}$ be the input of the generalized simple power word problem. We have $w =_G \pi(w)$ if and only if $w \in G_S$. That is equivalent to $w^{-1}\pi(w) =_G 1$, the projection π can be computed in uAC^0. ∎

Lemma 7.29 *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. It holds that*

$$\mathrm{SPowWP}(G) \in \mathsf{uAC}^0(\{\mathrm{WP}(F_2)\} \cup \{\mathrm{PowWP}(G_\alpha) \mid \alpha \in \mathcal{L}\}),$$

that is the simple power word problem in G can be solved in uAC^0 with oracles for the power word problem in each base group G_α and the word problem of the free group F_2. □

Proof We proceed by induction on the cardinality of $\mathcal{L}$. If $|L| = 1$, we can solve the simple power word problem in G by solving the power word problem in the base group. Otherwise, let $\beta \in \mathcal{L}$ be fixed. We define $L' = \mathcal{L} \setminus \{\beta\}$, $I' = I \cap (L' \times L')$, $P = \mathrm{GP}(L', I'; (G_\alpha)_{\alpha\in L'})$, $\mathrm{link}(\beta) = \{\gamma \mid (\beta, \gamma) \in I\}$ and $\mathrm{link}(G_\beta) = \mathrm{GP}(\mathrm{link}(\beta), I \cap (\mathrm{link}(\beta) \times \mathrm{link}(\beta)); (G_\alpha)_{\alpha\in\mathrm{link}\,\beta})$. Now we can write G as amalgamated product $G = P *_{\mathrm{link}(G_\beta)} (\mathrm{link}(G_\beta) \times G_\beta)$. By the induction hypothesis we can solve the simple power word problem in P and $\mathrm{link}(G_\beta)$ with oracles for the power word problem in each base group G_α and the word problem of the free group F_2. By Lemma 7.28 we can reduce the generalized simple power word problem of $\mathrm{link}(G_\beta)$ in P to the simple power word problem in P. It remains to show how to solve the simple power word problem in the amalgamated product.

Let the input be $w = w_1^{x_1} \dots w_n^{x_n} \in (\Gamma \times \mathbb{Z})^*$. Recall that $\Gamma_P = \bigcup_{\alpha\in L'} \Gamma_\alpha$. We define the projections $\pi_P : \Gamma^* \to \Gamma_P^*$ and $\pi_\beta : \Gamma^* \to \Gamma_\beta^*$ by

$$\pi_P(a) = \begin{cases} a & \text{if } a \in \Sigma_P, \\ a & \text{if } a \in \Sigma_A, \\ 1 & \text{if } a \in \Sigma_B, \end{cases} \qquad \pi_\beta(a) = \begin{cases} 1 & \text{if } a \in \Sigma_P, \\ 1 & \text{if } a \in \Sigma_A, \\ a & \text{if } a \in \Sigma_B. \end{cases}$$

Let $p_i = \pi_P(w_i)$ and $b_i = \pi_B(w_i)$. For the following construction we assume that $\pi_P(w) = p_1^{x_1} \dots p_n^{x_n} =_P 1$, as otherwise $w \neq_G 1$. By Lemma 7.27 we have $G \simeq *_{v\in P/A} B^{(v)} \rtimes P$. We want to write w as part of the kernel of π_P. We define $g_i = p_1^{x_1} \dots p_i^{x_i}$. Note that the g_i can be computed in uAC^0. We have $w =_G g_1 b_1^{x_1} g_1^{-1} g_2 b_2^{x_2} \dots g_{n-1}^{-1} g_n b_n^{x_n}$. Observe that $g_n = \pi_P(w) =_G 1$ and thus $w =_G g_1 b_1^{x_1} g_1^{-1} \dots g_n b_n^{x_n} g_n^{-1}$.

We compute

$$\begin{aligned} v_i &= \min\{1 \le j \le n \mid g_i A = g_j A\} \\ &= \min\{1 \le j \le n \mid g_i g_j^{-1} \in A\}. \end{aligned}$$

The computation can be reduced to the simple power word problem in G in uAC^0 by Lemma 7.28.

Now we have $w =_G 1$ if and only if $\pi_P(w) =_G 1$ and $(b_1^{x_1})^{(v_1)} \dots (b_n^{x_n})^{(v_n)} = 1$ in $*_{v\in R} B^{(v)}$. The lemma follows with Lemma 7.26. ■

Proof (Proof of Theorem 7.3) The proof, that the power word problem in a graph product G of f.g. groups can be decided in uAC^0 with oracles for the word problem in the free group F_2 and the power word problem in each base group G_α follows from the previous lemmata. By Lemma 7.24, the power word problem in G is uAC^0-Turing reducible to the simple power word problem in G and the word problem in the free group F_2. By Lemma 7.29, the simple power word problem in G is uAC^0-Turing reducible to the word problem in the free group F_2 and the power word problem in each base group G_α. ■

7.5 The uniform Case

In this section we look at the uniform power word problem in graph groups. First, we reduce the power word problem to the simple power word problem, where each power is restricted to a single base group, using the shortening process presented in the previous section. Then we directly solve the simple power word problem.

Lemma 7.30 *Let C be a non-trivial class of f.g. groups, $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$. The uniform power word problem* $\mathrm{PowWP}\,\mathrm{GP}\,C$*, where $G_1, \dots G_n$ and I are part of the input, can be solved in* uAC^0 *with oracles for* $\mathrm{C}_=\mathrm{L}$ *and the simple power word problem* $\mathrm{SPowWP}(\mathrm{GP}\,C)$. □

Proof We begin with the preprocessing described in Section 7.1. By Lemma 7.6 the preprocessing can be done in uAC^0 with oracles for $\mathrm{C}_=\mathrm{L}$ and for the uniform word problem of $\mathrm{GP}\,C$. We can use the oracle for the uniform simple power word problem $\mathrm{SPowWP}(\mathrm{GP}\,C)$ to solve the uniform word problem in $\mathrm{GP}\,C$. Following the preprocessing we have a word $w = u_0 p_1^{x_1} u_1 \dots p_n^{x_n} u_n$. We proceed with the shortening procedure for each $p \in \{p_i \mid 1 \le i \le n\}$. The shortening procedure can be computed in parallel for each p. The computation of the shortened word requires iterated additions which is in uTC^0, and therefore can be done in uAC^0 with oracle gates for $\mathrm{WP}(F_2)$. By [LZ77] the word problem of the free group F_2 can be solved in LOGSPACE and therefore in $\mathrm{C}_=\mathrm{L}$. By Lemma 7.23 the exponents of the shortened word are bounded by a polynomial. Thus, the shortened word can be represented as a word $\hat{w}$ of polynomial length over the alphabet $(\Gamma \times \mathbb{Z})$. We can solve the power word problem of w by solving the simple power word problem of $\hat{w}$. ■

Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f.g. groups. The following embedding of G into a (possibly infinite-dimensional) linear group has been presented in [Kau17]. The mapping $\sigma : G \to \mathrm{GL}(\mathbb{Z}^\Gamma)$ is defined by $w \mapsto \sigma_w$, where $\sigma_w = \sigma_{w_1} \dots \sigma_{w_n}$ for a factorization $w = w_1 \dots w_n$. The mapping $\sigma_a : \mathbb{Z}^\Gamma \to \mathbb{Z}^\Gamma$ is defined as the linear extension of

$$\sigma_a(b) = \begin{cases} -a & \text{if } a, b \in \Gamma_\alpha \text{ for some } \alpha \text{ and } ab =_{G_\alpha} 1, \\ [ab] - a & \text{if } a, b \in \Gamma_\alpha \text{ for some } \alpha \text{ and } ab \neq_{G_\alpha} 1, \\ b + 2a & \text{if } a \in \Gamma_\alpha, b \in \Gamma_\beta \text{ for some } \alpha \neq \beta \text{ and } (\alpha, \beta) \notin I, \\ b & \text{if } a \in \Gamma_\alpha, b \in \Gamma_\beta \text{ for some } \alpha \neq \beta \text{ and } (\alpha, \beta) \in I. \end{cases}$$

The following lemma is due to Jonathan Kausch [Kau17].

Lemma 7.31 *[Kau17, Lemma 3.3.4] Let $w \in \Gamma^*$ be a reduced trace and $we =_G uev$ such that $e \in \Gamma_\varepsilon$, $u, v \in \Gamma^*$ and e is the single maximal element of ue. Moreover, let*

$$\sigma_w(e) = \sum_{c\in\Gamma} \lambda_c \cdot c,$$

and let $u = u_0 a_1 u_1 \dots a_n u_n$ be the α-factorization of u, then for $c \in \Gamma_\alpha$ we have $\lambda_c \ge 0$ and

$$\lambda_c > 0 \iff \text{For some } i \text{ with } 1 \le i \le n : c =_G \begin{cases} a_i \dots a_n & \text{if } \alpha \neq \varepsilon, \\ a_i \dots a_n \cdot c & \text{if } \alpha = \varepsilon. \end{cases}$$ □

Algorithm 7.1 Computing the coefficient $a \in \Gamma_\alpha$ of $\sigma_w(x)$ in GapL

Input: $a, w_1 \dots w_n$; where $w_i = g_i^{x_i}$
$(k, \ell, s) \leftarrow (n+1, n+1, 1)$
for i in $[n, \dots, 1]$ **do**
  if $k = n+1 \wedge \ell = n+1$ **then** // $\sigma_{w_i}(x) = 2w_i + x$
    Guess "Branch 1", "Branch 2" or "Branch 3"
    if "Branch 1" or "Branch 2" **then** $(k, \ell, s) \leftarrow (i, i, 1)$
    if "Branch 3" **then** $(k, \ell, s) \leftarrow (n+1, n+1, s)$
  else if $\text{alph}(g_i) = \text{alph}(g_k)$ **then** // $\sigma_{w_i}(\pi_\alpha(w_k \dots w_\ell)) = [w_i \pi_\alpha(w_k \dots w_\ell)] - w_i$
    Guess "Branch 1" or "Branch 2"
    if "Branch 1" **then** $(k, \ell, s) \leftarrow (i, \ell, s)$
    if "Branch 2" **then** $(k, \ell, s) \leftarrow (i, i, -s)$
  else if $(\text{alph}(g_i), \text{alph}(g_k)) \notin I$ **then** // $\sigma_{w_i}(\pi_\alpha(w_k \dots w_\ell)) = \pi_\alpha(w_k \dots w_\ell) + 2w_i$
    Guess "Branch 1", "Branch 2" or "Branch 3"
    if "Branch 1" **then** $(k, \ell, s) \leftarrow (k, \ell, s)$
    if "Branch 2" or "Branch 3" **then** $(k, \ell, s) \leftarrow (i, i, s)$
  else // $\sigma_{w_i}(\pi_\alpha(w_k \dots w_\ell)) = \pi_\alpha(w_k \dots w_\ell)$
    $(k, \ell, s) \leftarrow (k, \ell, s)$
  end if
end for
if $k \neq n+1 \wedge a =_{G_\alpha} \pi_\alpha(w_k \dots w_\ell)$ **then** // Use oracle for PowWP C
  if $s = 1$ **then** accept
  if $s = -1$ **then** reject
else
  Guess "Branch 1" or "Branch 2"
  if "Branch 1" **then** accept
  if "Branch 2" **then** reject
end if

Our solution to the uniform simple power word problem is based on the solution to the word problem presented in [Kau17]. The underlying idea is to add an additional free group $\langle x \rangle$ to the graph product, which is dependent on all other groups. Let π_α be the projection to Γ_α, defined by $\pi_\alpha(a) = a$ for $a \in \Gamma_\alpha$ and $\pi_\alpha(a) = 1$ for $a \notin \Gamma_\alpha$. As a consequence of Lemma 7.31 we have $\sigma_w(x) = x$ if and only if $w =_G 1$. Non-zero coefficients of $\sigma_w(x)$ are a subword of $\pi_\alpha(w)$ for some $\alpha \in \mathcal{L}$.

Lemma 7.32 *Let C be a non-trivial class of f.g. groups, $G = \text{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$. The uniform simple power word problem in G, where $G_1, \dots G_n$ and I are part of the input, can be solved in* $\text{C}_=\text{L}$ *with oracle gates for the uniform power word problem* PowWP C. □

Proof Let $w = p_1^{x_1} \dots p_n^{x_n} \in G$, where $p_i \in \Gamma$ and $x_i \in \mathbb{Z}$. If $w \neq_G 1$ then there are $\alpha \in \mathcal{L}$ and $1 \leq k \leq \ell \leq n$ such that the coefficient $\pi_\alpha(p_k^{x_k} \dots p_\ell^{x_\ell})$ is not zero.

To compute the coefficients we use Algorithm 7.1. The computation of $\sigma_w(x)$ can be regarded as a tree. Each inner node is annotated with a coefficient and a sign, and corresponds to a contribution of either $+1$ or -1 to that coefficient. The coefficient is stored using two indices k and ℓ into the input. We use $(k, \ell) = (n+1, n+1)$ to represent x.

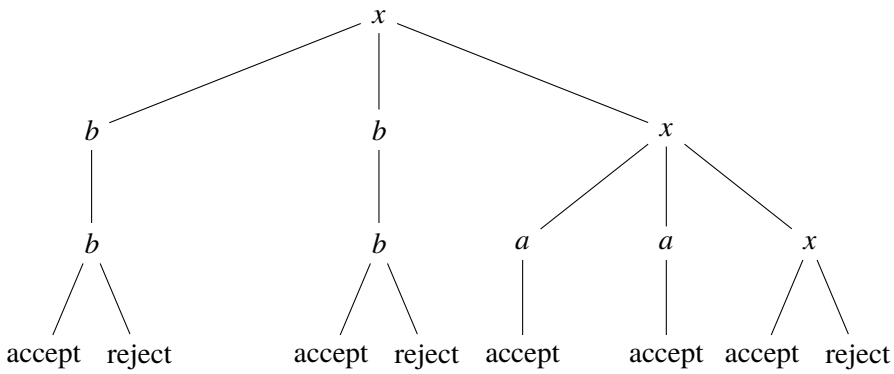


Figure 7.1: Computation of the coefficient λ_a of $\sigma_{ab}(x)$ by Algorithm 7.1. Each inner node is labeled with the coefficient it contributes to. The algorithm stores the coefficient using two indices k and ℓ. The nodes on the second level correspond to $\sigma_b(x)$, the nodes on the third level correspond to $\sigma_{ab}(x)$. If they are labeled with a they have one leaf node as a child which is an accepting path (or a rejecting path if the sign is negative). If they are not labeled with a, then there are two leaf node children, one is an accepting path, the other a rejecting path, so they do not affect the result of the computation.

The root node is x. Let $w = w'a$, with $a \in \Gamma$. Then $\sigma_w(x) = \sigma_{w'}(\sigma_a(x))$. The nodes on the second level, that is the children of the root node, correspond to σ_a. The last level made up of inner nodes corresponds to σ_w. At that point the algorithm checks if the node is annotated with the coefficient that should be computed. That is done using the oracle for the uniform power word problem in C. If the node is annotated with the coefficient that should be computed, then the computation will accept the input if the sign is positive, and reject if the sign is negative. The coefficient is computed as the difference of the number of accepting paths and the number of rejecting paths. Nodes not labeled with the coefficient that should be computed will have two leaf nodes as children, one accepting and one rejecting computation path, thus they do not affect the result. Therefore, the computation of a coefficient is in $\mathsf{GapL}^{\mathrm{PowWP}\,C}$, and we can check in $\mathrm{C}_=\mathrm{L}^{\mathrm{PowWP}\,C}$ whether a coefficient is zero. An example of a computation tree is presented in Figure 7.1.

We can check whether all coefficients are zero, as $\mathrm{C}_=\mathrm{L}^{\mathrm{PowWP}\,C}$ is closed under finite conjunctions [CMTV98]. ∎

Proof (of Theorem 7.4) Let C be a non-trivial class of f.g. groups, $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$. The uniform power word problem in G, where $G_1, \ldots G_n$ and I are part of the input, can be solved in uAC^0 with oracle gates for $\mathrm{C}_=\mathrm{L}$ and the uniform power word problem $\mathrm{PowWP}\,C$. By Lemma 7.30, the uniform power word problem in G is uAC^0-Turing reducible to the simple power word problem in G and the word problem of $\mathrm{C}_=\mathrm{L}$. By Lemma 7.32, the simple power word problem in G is $\mathrm{C}_=\mathrm{L}$-Turing reducible to the uniform power word problem $\mathrm{PowWP}\,C$. ∎

8 Consequences for the Knapsack Problem in Graph Groups

The knapsack problem (over $\mathbb{Z}$) is a classical optimization problem. It is defined as follows. There are n items, with weight w_i and value v_i. There is a maximum weight W. The goal is to maximize the total value $\sum x_i v_i$, while not exceeding the weight bound $\sum x_i w_i \leq W$, where $x_i \in \mathbb{Z}, x_i \geq 0$.

When posed as a decision problem, there is a target value V, and the question is whether there are x_i such that $\sum x_i v_i \geq V$, while not exceeding the weight bound. This decision variant of the knapsack problem is one of the 21 problems shown to be NP-complete by Karp in 1972 [Kar72].

Myasnikov et al. have formulated the classical knapsack problem for arbitrary groups [MNU15]. Their definition is a follows.

Definition 8.1 Let G be a group. Let $g_1, \ldots, g_n, g \in G$. Then the knapsack problem is to decide whether there are $x_1, \ldots, x_n \in \mathbb{Z}$, with $x_i \geq 0$, such that

$$g_1^{x_1} \cdots g_n^{x_n} =_G g.$$ □

Using the above definition, with the group $\mathbb{Z}$ the input is encoded unary and not binary (as with the classical knapsack problem). Thus the problem becomes easier. However, if we choose a group G that is sufficiently complex, the problem will be NP-complete again, even for groups with the word problem in P.

M. Lohrey and G. Zetzsche have studied the knapsack problem for fixed graph groups. They have shown the following results for a fixed graph group G with independence relation I [LZ16].

- If I is a complete graph, then the knapsack problem for G is uTC^0-complete.
- If I does not contain an induced cycle of four nodes (C_4) or an induced path of length four (P_4), then the knapsack problem for G is LogCFL-complete.
- If I contains an induced C_4 or P_4, then the knapsack problem for G is NP-complete.

We consider the uniform knapsack problem for graph groups. That is, the variant of the knapsack problem where the group, most importantly the independence graph I, is part of the input. We use our result on the uniform power word problem in graph groups to give a crude algorithm for solving the uniform knapsack problem in graph groups.

Theorem 8.2 *The uniform knapsack problem for graph groups is* NP*-complete. On the input of an alphabet Σ, an independence relation $I \subseteq \Sigma$ and group elements $g_1, \ldots, g_n, g$, it can be decided in* NP *whether there are $x_1, \ldots, x_n \in \mathbb{Z}$ with $x_i \geq 0$ such that*

$$g_1^{x_1} \cdots g_n^{x_n} =_G g.$$ □

Proof The proof idea is to guess a solution and then verify it using the algorithm for solving the power word problem.

By [LZ18, Theorem 3.11], there is an exponential bound on the solution. More precisely there is a polynomial $p(n)$, such that if there is a solution, then there is a solution $x_1, \ldots, x_n$ with $x_i \leq 2^{p(n)}$. Therefore, we can guess all potential solutions within the bound in NP, and if there is a solution, we will guess a solution.

From Theorem 7.4 it follows that the uniform power word problem in graph groups can be decided in P. Hence, the uniform knapsack problem can be decided in NP. Finally, NP-completeness follows immediately from the NP-completeness of the knapsack problem for a certain fixed graph groups, which has been shown in [LZ16]. ∎

9 Conclusion

This thesis has studied the power word problem, a variant of the word problem, in graph groups and graph products. The underlying idea to our solutions to this problem is to replace the exponents with smaller ones, bounded by a polynomial in the size of the input. We call the resulting word the shortened word. Once we have the shortened word, we can expand the powers and then solve the word problem. The majority of this thesis is proving the correctness of that algorithm.

The core observation underlying our proof is that if there are two different powers and we apply free reduction, then the amount of characters that cancel is limited. In Chapter 5 we have formalized this observation and given an upper bound on the number of characters that can cancel. We used these bounds to calculate how much we can reduce the exponents when computing the shortened word.

To show that our solution for the power word problem is correct, we have shown that the shortened word is equal to the identity if and only if the original input word is equal to the identity. As a technical instrument to facilitate the proof, we have defined a symbolic rewriting system over an infinite alphabet.

We analysed the computational complexity of the power word problem in graph groups and graph products in Chapter 6 and Chapter 7. Following that, in Chapter 8 we have looked at the implications of our research to the knapsack problem. Our contribution can be summarized as follows.

- The power word problem in a fixed graph group is uAC^0-Turing reducible to the word problem in the free group with two generators F_2.
- The power word problem in a fixed graph product $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ is uAC^0-Turing reducible to the power word problem each base group G_α and the word problem in the free group F_2.
- Given a non-trivial class C of f.g. groups, the uniform power word problem for graph products can be decided in $\mathrm{C}_=\mathrm{L}$ with oracles for the uniform power word problem in C.
- We extended upon the work of M. Lohrey and G. Zetzsche by showing NP-completeness for the uniform knapsack problem in graph groups.

Bibliography

[Boo58] W. W. Boone. “THE WORD PROBLEM”. In: *Proceedings of the National Academy of Sciences* 44.10 (1958), pp. 1061–1065. ISSN: 0027-8424. DOI: 10.1073/pnas.44.10.1061. eprint: https://www.pnas.org/content/44/10/1061.full.pdf. URL: https://www.pnas.org/content/44/10/1061 (cit. on p. 7).

[CGW09] J. Crisp, E. Godelle, B. Wiest. “The conjugacy problem in subgroups of right-angled Artin groups”. In: *Journal of Topology* 2.3 (2009), pp. 442–460 (cit. on pp. 19, 20).

[CMTV98] H. Caussinus, P. McKenzie, D. Thérien, H. Vollmer. “NondeterministicNC1Computation”. In: *Journal of Computer and System Sciences* 57.2 (1998), pp. 200–212 (cit. on p. 60).

[Deh11] M. Dehn. “Über unendliche diskontinuierliche Gruppen”. In: *Mathematische Annalen* 71.1 (1911), pp. 116–144 (cit. on p. 7).

[Dub85] C. Duboc. “Some properties of commutation in free partially commutative monoids”. In: *Information Processing Letters* 20.1 (1985), pp. 1–4. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(85)90121-8 (cit. on p. 13).

[Dub86] C. Duboc. “On some equations in free partially commutative monoids”. In: *Theoretical Computer Science* 46 (1986), pp. 159–174 (cit. on pp. 13, 19).

[FW65] N. J. Fine, H. S. Wilf. “Uniqueness theorems for periodic functions”. In: *Proceedings of the American Mathematical Society* 16.1 (1965), pp. 109–114 (cit. on p. 23).

[Gre90] E. R. Green. “Graph products of groups”. PhD thesis. University of Leeds, 1990 (cit. on pp. 7, 17).

[HLM12] N. Haubold, M. Lohrey, C. Mathissen. “Compressed decision problems for graph products and applications to (outer) automorphism groups”. In: *International Journal of Algebra and Computation* 22.08 (2012), p. 1240007 (cit. on p. 7).

[Kar72] R. M. Karp. “Reducibility among combinatorial problems”. In: *Complexity of computer computations*. Springer, 1972, pp. 85–103 (cit. on pp. 8, 61).

[Kau17] J. Kausch. “The parallel complexity of certain algorithmic problems in group theory”. PhD thesis. University of Stuttgart, 2017. DOI: http://dx.doi.org/10.18419/opus-9152 (cit. on pp. 17–19, 22, 37, 55, 56, 58, 59).

[Loh04] M. Lohrey. “Word problems on compressed words”. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2004, pp. 906–918 (cit. on p. 7).

[LW19] M. Lohrey, A. Weiß. “The power word problem”. In: *arXiv preprint arXiv:1904.08343* (2019) (cit. on pp. 7, 27).

[LZ16] M. Lohrey, G. Zetzsche. “The complexity of knapsack in graph groups”. In: *arXiv preprint arXiv:1610.00373* (2016) (cit. on pp. 8, 61, 62).

[LZ18] M. Lohrey, G. Zetzsche. “Knapsack in graph groups”. In: *Theory of Computing Systems* 62.1 (2018), pp. 192–246 (cit. on p. 62).

[LZ77] R. J. Lipton, Y. Zalcstein. “Word problems solvable in logspace”. In: *Journal of the ACM (JACM)* 24.3 (1977), pp. 522–526 (cit. on p. 58).

[MNU15] A. Myasnikov, A. Nikolaev, A. Ushakov. “Knapsack problems in groups”. In: *Mathematics of Computation* 84.292 (2015), pp. 987–1016 (cit. on pp. 8, 61).

[Nov55] P. S. Novikov. “On the algorithmic unsolvability of the word problem in group theory”. Russian. In: (1955) (cit. on p. 7).

[Rei08] O. Reingold. “Undirected connectivity in log-space”. In: *Journal of the ACM (JACM)* 55.4 (2008), pp. 1–24 (cit. on p. 45).

[Wra88] C. Wrathall. “The word problem for free partially commutative groups”. In: *Journal of Symbolic Computation* 6.1 (1988), pp. 99–104 (cit. on p. 13).

All links were last followed on June 28, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature