The parallel complexity of certain algorithmic problems in group theory

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Jonathan Kausch

aus Stuttgart

Hauptberichter: Prof. Dr. Volker Diekert

Mitberichter: Prof. Dr. Markus Lohrey

Tag der mündlichen Prüfung: 1. Februar 2017

Institut für Formale Methoden der Informatik

2017

Contents

List of Figures

List of Algorithms

Abstract

In this thesis, we study the parallel complexity of certain problems in algorithmic group theory. These problems are the word problem, the geodesic and normal form problem and the conjugacy problem. We study these problems for some products of groups, namely direct products, free products and graph products. For all those we consider the problems for a fixed group as well as the uniform versions. Uniform means that the group is part of the input. Among the studied problems, the word problem is the most important one and necessary to solve any of the other problems.

For direct products, solving any of the mentioned problems reduces directly to the problems in the base groups of the product. Some of the solutions for the direct product are required for solving the problems of the more complicated products.

For free products, we show that the word problem reduces in AC^0 to the word problem of the base groups and the word problem of the free group of rank two. This does hold for the word problem of a fixed free product as well as for the uniform version. For the geodesic and normal form problem of free products, we introduce an equivalence relation. This relation can be decided in AC^0 by using oracle calls to the word problems of the base groups. The solution of the word and geodesic problem can then be used to solve the conjugacy problem. In free products, two cyclically reduced words are conjugate if and only if they are transposed.

Direct products and free products are special cases of graph products. A graph product can be written as an amalgamated product of smaller graph products. We first solve the word problem of some restricted amalgamated product. This solution can then be used to solve the word problem of a fixed graph product inductively. We obtain that the word problem of a fixed graph product is AC^0-reducible to the word problem of its base groups and the word problem of the free group of rank two. Unfortunately, this method cannot be used to solve the uniform word problem. We show that the uniform word problem of graph products is NL-hard. For solving it, we introduce an embedding of the graph product into the automorphism group of some (possibly infinite dimensional) vector space. We show that the evaluation of these automorphisms can be realized in GapL and that verifying its result is in $\mathsf{C}_=\mathsf{L}$ by using oracle calls to the word problem in the base groups. The uniform word problem of graph products can be reduced to the evaluation of these automorphisms. For the geodesic problem, we introduce

another equivalence relation. As for free products, this relation can be decided in AC^0 by using oracle calls to the (uniform) word problem. In graph products the normal form of some word is the length-lexicographic first equivalent word. For solving the normal form problem, first a geodesic and then the lexicographic normal form of this geodesic is computed. We show that for a fixed graph product the computation of the lexicographic normal form is in TC^0 and TC^0-complete for most graph products. We further show that the uniform version is FNL-complete. The solution of the word and geodesic problem can then be used to solve the conjugacy problem. First, we show how to compute cyclically reduced words in AC^0 by using oracle calls to the word problem. Then we show that in graph products two cyclically reduced words are conjugate if and only if they are obtained by a sequence of transpositions. This problem can then be solved by verifying whether the first word is a factor of some power of the second word. For a fixed graph product the factor problem can be decided in AC^0 by using oracle calls to the word problem. For the uniform factor problem we show that it can be decided in NL by using oracle calls to the uniform word problem. Combining all this gives a solution to the (uniform) conjugacy problem of graph products.

Parts of this thesis have already been published in the following two papers. My contribution to both papers is fifty percent.

- Volker Diekert and Jonathan Kausch. Logspace computations in graph products. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *ISSAC*, pages 138–145. ACM, 2014.

- Volker Diekert and Jonathan Kausch. Logspace computations in graph products. *Journal of Symbolic Computation*, 75(C):94–109, July 2016.

Despite the above, we published the following papers.

- Volker Diekert, Jonathan Kausch, and Markus Lohrey. Logspace computations in graph groups and Coxeter groups. In D. Fernãndez-Baca, editor, *Proceedings of LATIN 2012*, volume 7256 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2012.

- Volker Diekert, Jonathan Kausch, and Markus Lohrey. Logspace computations in Coxeter groups and graph groups. In *Computational and Combinatorial Group Theory and Cryptography*, volume 582 of *Contemporary Mathematics*, pages 77–94. Amer. Math. Soc., 2012. Journal version of LATIN 2012, 243–254, LNCS 7256, 2012.

Zusammenfassung

In dieser Arbeit untersuchen wir die parallele Komplexität gewisser Probleme in der algorithmischen Gruppentheorie. Diese Probleme sind das Wortproblem (engl. word problem), das Geodäten- und Normalformenproblem (engl. geodesic and normal form problem) und das Konjugationsproblem (engl. conjugacy problem). Wir untersuchen diese Probleme für Produkte von Gruppen, genauer für direkte Produkte, freie Produkte und Graphprodukte. Für all jene betrachten wir die Probleme sowohl für eine feste Gruppe als auch ihre uniforme Variante. Uniform bedeutet, dass die Gruppe Teil der Eingabe ist. Unter den untersuchten Problemen ist das Wortproblem das wichtigste und notwendig für die Lösung der anderen Probleme.

Die erwähnten Probleme lassen sich für direkte Produkte unmittelbar durch reduzieren auf die Probleme in den Basisgruppen lösen. Manche der Lösungen für direkte Produkte werden benötigt, um die Probleme in den komplizierteren Produkten zu lösen.

Für freie Produkte können wir zeigen, dass sich das Wortproblem in AC^0 auf das Wortproblem der Basisgruppen und das Wortproblem der freien Gruppe vom Rang zwei reduzieren lässt. Dies gilt sowohl für das Wortproblem eines festen freien Produkts als auch für die uniforme Variante. Für das Geodäten- und Normalformenproblem freier Produkte führen wir eine Äquivalenzrelation ein. Diese Relation kann in AC^0 durch Orakelanfragen an das Wortproblem des freien Produkts entschieden werden. Die Lösung des Wort- und Geodätenproblems kann schließlich genutzt werden, um das Konjugationsproblem zu lösen. In freien Produkten sind zwei zyklisch reduzierte (engl. cyclically reduced) Wörter genau dann konjugiert, wenn sie transponiert zueinander sind.

Direkte Produkte und freie Produkte sind Spezialfälle des Graphprodukts. Ein Graphprodukt kann als amalgamiertes Produkt kleinerer Graphprodukte aufgefasst werden. Wir lösen zuerst das Wortproblem dieser eingeschränkten amalgamierten Produkte. Diese Lösung kann schließlich genutzt werden, um das Wortproblem eines festen Graphprodukts induktiv zu lösen. Wir erhalten, dass das Wortproblem eines festen Graphprodukts AC^0-reduzierbar auf das Wortproblem in den Basisgruppen und das Wortproblem der freien Gruppe vom Rang zwei ist. Diese Methode lässt sich nicht für das uniforme Wortproblem in Graphprodukten nutzen. Wir zeigen, dass das uniforme Wortproblem von Graphprodukten NL-schwer ist. Um dieses zu lösen führen wir eine Einbettung des Graphprodukts in

die Automorphismengruppe eines (möglicherweise unendlich dimensionalen) Vektorraums ein. Wir zeigen, dass durch Orakelanfragen an das Wortproblem in den Basisgruppen, die Auswertung dieser Automorphismen in GapL realisiert werden kann und, dass die Verifikation des Ergebnisses in $\mathsf{C_=L}$ ist. Das uniforme Wortproblem kann auf die Auswertung dieser Automorphismen reduziert werden. Für das Geodätenproblem führen wir eine weitere Äquivalenzrelation ein. Wie schon für freie Produkte kann diese Relation in AC^0 durch Orakelanfragen an das (uniforme) Wortproblem entschieden werden. Die Normalform eines Wortes ist das längenlexikographisch erste äquivalente Wort. Um das Normalformenproblem zu lösen wird zuerst eine Geodätische und anschließend die lexikographische Normalform dieser Geodätischen berechnet. Wir zeigen, dass die Berechnung der lexiographischen Normalform in TC^0 möglich ist und TC^0-vollständig für die meisten Graphprodukte ist. Wir zeigen außerdem, dass die uniforme Variante FNL-vollständig ist. Die Lösung des Wort- und Geodätenproblems kann schließlich genutzt werden, um das Konjugationsproblem zu lösen. Wir zeigen zuerst, wie sich zyklisch reduzierte Wörter in AC^0 durch Orakelanfragen an das Wortproblem berechnen lassen. Anschließend zeigen wir, dass in Graphprodukten zwei zyklisch reduzierte Wörter genau dann konjugiert sind, wenn sie durch eine Folge von Transpositionen auseinander hervorgehen. Dieses Problem kann schließlich gelöst werden, indem geprüft wird, ob das erste Wort ein Faktor einer Potenz des zweiten Worts ist. Für ein festes Graphprodukt kann das Faktorproblem in AC^0 durch Orakelanfragen an das Wortproblem entschieden werden. Für das uniforme Faktorproblem zeigen wir, dass es in NL durch Orakelanfragen an das uniforme Wortproblem entschieden werden kann. Zusammengesetzt ergibt sich eine Lösung des (uniformen) Konjugationsproblems.

Chapter 1

Introduction

Algorithmic problems in group theory have been considered for over 100 years. Research started with the important work of Titze and Dehn back in the beginning of the 20th century. Already in 1911 Max Dehn formulated the three fundamental questions in algorithmic group theory. The word problem, the conjugacy problem and the isomorphism problem. A detailed overview on these decision problems can be found in [Mil92]. In this work we are interested in the word and conjugacy problem. Moreover, we are interested in the geodesic and the normal form problem.

1. **Word problem**: Is a given word equal to the identity in the group?
2. **Conjugacy problem**: Are two given words conjugate in the group?
3. **Geodesic problem**: Find a shortest representative of a given word.
4. **Normal form problem**: Find a unique representative of a given word.

With the emergence of parallel complexity classes it became increasingly interesting to study the parallel complexity of those problems. The parallel complexity classes of interest in this work lie between the first and second level of the NC-hierarchy. For reductions, whenever possible AC^0 is used. In some cases, the power of TC^0 is necessary.

$$\mathsf{AC}^0 \subsetneq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{C}_=\mathsf{NC}^1 \subseteq \mathsf{LOGSPACE} \subseteq \mathsf{NL} \subseteq \mathsf{C}_=\mathsf{L} \subseteq \mathsf{AC}^1 \subseteq \mathsf{TC}^1 \subseteq \mathsf{NC}^2$$

In the above chain, only the first inclusion is known to be strict. For all other inclusions it is unknown if they are strict. All those complexity classes are contained in P. There seem to exist inherently sequential problems in P and it is widely believed that P and the above complexity classes differ.

The focus in this work is on certain group products, namely direct products, free products and graph products. Furthermore, for the word problem some restricted amalgamated products are considered. Graph products are a combination of direct and free products. They naturally extend graph groups and were introduced in the PhD thesis of Elisabeth Green [Gre90]. Graph groups have a close relationship to

trace monoids. Trace monoids are also known as free partially commutative semigroups and were introduced in the 1970's for the study of concurrent systems. An extensive overview on trace theory may be found in [DR95, Die90]. Many results on the algorithmic complexity in trace monoids are linear time algorithms. In case of the word problem of trace monoids, it is known to be TC^0-complete [ÀG91]. The precise complexity of the word problem in graph groups is not resolved by now for the following reason. Except for the special case of a free abelian group, graph groups contain a free group. While the word problem of the free group of rank two is NC^1-hard by [Rob93], it is only known to be contained in the counting complexity class $\mathsf{C}_=\mathsf{NC}^1$. Anyway, the two complexity classes are very close in terms of circuit depth. Nonetheless, the question, whether the word problem of the free group can actually be solved in NC^1 remains open.

Graph groups are also known as free partially commutative groups and as right angled Artin groups. They attracted much attention lately, as they are very important for some problems in three dimensional topology. For a survey on right angled Artin groups and their connection to topology see [Cha07]. Related to graph groups are right angled Coxeter groups. Right angled Artin groups and right angled Coxeter groups are linear groups. Therefore, their word problem is contained in $\mathsf{C}_=\mathsf{NC}^1$. Linear groups on their own are a very important class of groups. By the work of Lipton, Zalcstein [LZ77] and Simon [Sim79] finitely generated linear groups posses a word problem in $\mathsf{LOGSPACE}$. Furthermore, the word problem of linear groups over the integers is contained in $\mathsf{C}_=\mathsf{NC}^1$ by [CMTV98].

Recently the computation of normal forms in $\mathsf{LOGSPACE}$ attracted attention [EEO13, DKL12a, DKL12b]. In the context of normal forms the following question was raised in [EEO13]. Is the word problem of a free product over two groups each having a word problem in $\mathsf{LOGSPACE}$ again in $\mathsf{LOGSPACE}$? We answer this question positively. In fact, it already follows as a corollary of [Waa90] dating back to 1990. Waack showed, that the word problem of a free product is NC^1-reducible to the word problem of the base groups and the word problem of the free group on two generators. We generalize the result of Waack to graph products. More precisely, we show that the word problem of graph products is AC^0-reducible to the word problem of the base groups and the word problem of the free group on two generators. Having a solution to the word problem is necessary for all the other problems. The mentioned solution of graph products is for a fixed graph product. Besides considering the problems for a fixed group, we further consider the uniform problems. Uniform means that the particular group to solve the problem for is part of the input. For a fixed group, all problems turn out to have a solution below $\mathsf{LOGSPACE}$, when using an oracle for the problem in the base group. For free products, the complexity of the non-uniform and uniform problem turns out to be the same. In case of graph products, we show that the the uniform problems are NL-hard and contained in $\mathsf{C}_=\mathsf{L}$ or $\mathsf{AC}^0(\mathsf{C}_=\mathsf{L})$ with oracles for the base groups.

Outline

The outline of this thesis is as follows. In Chapter 2 an introduction to the basic notation and an overview on the parallel and counting complexity classes considered in this thesis is given. Chapter 3 gives a introduction to graph products. The graph product is the most flexible product among the considered group products in this thesis. It is unknown, whether a graph product of linear groups is linear again. However, we can show that any graph product is linear over some infinite dimensional vector space. This result is in particular important to solve the uniform word problem of graph products, which then serves as a basis for the other uniform problems of graph products. As a corollary of this linearity result, we additionally recover a result of [HW99]. It states, that graph products over finite groups are linear. In Chapter 4 an overview on the considered problems from algorithmic group theory is given. Among them are the (uniform) word, conjugacy, geodesic and normal form problem. Furthermore, the encoding of the input to work with parallel circuits is specified.

The next three chapters are devoted to the word problem, the geodesic and normal form problem and the conjugacy problem. Chapter 5 is divided into the word problem of direct products, free products, some restricted amalgamated products and graph products. For free products, we show, that the word problem reduces to the base groups and some free group. This result was known due to Waack [Waa90] before. However, our proof is more general and does solve the word problem without using induction for an arbitrary number of base groups in the free product. Therefore, it can also be used to solve the uniform word problem of free products. The results on amalgamated products mainly serve as a tool to solve the word problem of a fixed graph product. Graph products can inductively be written as amalgamated product of smaller graph products. For the uniform word problem of graph groups, we show that it is NL-hard. Moreover, for the uniform word problem in graph products, we show, that it can be solved in $\mathsf{C}_{=}\mathsf{L}$. The next chapter is on geodesics and normal forms. For free products we give an equivalence relation to solve the geodesic problem. Since geodesics in free products are unique, the normal form problem is then a direct consequence. The equivalence relation for free products can be extended to an equivalence relation for graph products. The equivalence classes yield again a geodesic. The normal form in graph products involves computing a lexicographic ordering. For geodesics in a fixed graph product, we show that this ordering can be computed in TC^0. For the uniform problem, we show that computing the lexicographic ordering of a geodesic is NL-complete. Chapter 7 is divided into the conjugacy problem of direct products, free products and graph products. For free products the conjugacy problem reduces to the transposition problem and can therefore be directly translated into a circuit. For graph products, the conjugacy problem can be reduced to the transitive closure of

the transposition problem, which then reduces to a pattern matching problem in dependence graphs. For a fixed graph product, this pattern matching can be done in AC^0 by using the word problem as oracle. For the uniform conjugacy problem, we show, that this pattern matching can be done in NL with oracles for the word problem.

Finally, in Chapter 8 some remaining open problems and consequences are discussed. Moreover, all results achieved in this thesis are summarized in a table.

Chapter 2

Preliminaries

This chapter introduces the notation and definitions used throughout this thesis. Moreover, an overview on the relation of these definitions is given and some of their basic properties are stated. Readers familiar with these definitions may skip this chapter.

We let $\mathbb{N}$ contain the number 0. The disjoint union of two sets is denoted by $\uplus$. The logarithm log is always meant to be the logarithm of base 2. The cardinality of some set A is denoted by $|A|$. We denote a subgroup H of some group G by $H \leq G$. A homomorphism $\pi : G \to H$ of some group G onto a subgroup H is called a *retract* if $\pi(h) = h$ for all $h \in H$.

2.1 Graphs

A basic concept of describing relations between objects are graphs. In our settings a finite graph G is a pair (V, E). The vertices V are a finite set and the edges E are a subset of $V \times V$. In general graphs are directed. If we consider undirected graphs, then $(v, w) \in E$ if and only if $(w, v) \in E$. In this case, we sometimes write $\{v, w\} \in E$. The graph $G' = (V', E')$ is called a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq V' \times V' \cap E$. Let V' be a subset of V. Then the graph $G' = (V', E')$ with $E' = (V' \times V') \cap E$ is called the *induced subgraph* of G. Sometimes, the vertices of the graph are labeled with letters of a finite alphabet Σ. The labeling is achieved via some function $\lambda : V \to \Sigma$ and the graph G is extended by this function, that is $G = (V, E, \lambda)$. In this case, the subgraph G' would be $(V', E', \lambda|_{V'})$. The (out) degree of a vertex $v \in V$ is $d_v = |\{w \mid (v, w) \in E\}|$. A path of length n in the graph is a sequence of vertices $v_0, \ldots, v_n \in V$ with $(v_i, v_{i+1}) \in E$ for all $0 \leq i < n$. To say that there exists a path from $s \in V$ to $t \in V$, we write there exists an $s \xrightarrow{*} t$ *path*. The path is called simple, if it does not contain any repeated vertices, that is $v_i \neq v_j$ for all $0 \leq i \neq j \leq n$. A cycle is a path with $v_0 = v_n$ and all other vertices on that path are distinct, that is $v_i \neq v_j$ for all $0 < i \neq j < n$ and $v_0 \neq v_i \neq v_n$ for all $0 < i < n$. A graph is called *acyclic*, if it does not contain

any cycles. On an *acyclic graph*, a *topological ordering* of the vertices is possible. Let $V = \{v_0, \ldots, v_n\}$. Then the vertices are ordered topological, if there is no path from v_j to v_i for $j > i$. The *link* of a node $v \in V$ is defined as the set of all neighbors, that is $\text{link}(v) = \{u \in V \mid (v, u) \in E\}$.

2.2 Complexity

This section gives a general overview on the complexity classes considered in this thesis. An overview over the relation of the mentioned complexity classes is given in Figure 2.2 on page 27. Sometimes it is convenient to relate complexity classes and function classes. Let $\mathcal{F}$ be a function class and $\mathcal{C}$ a complexity class. We say $\mathcal{C}$ is contained in $\mathcal{F}$, if for any $L \in \mathcal{C}$ its characteristic function χ_L is in $\mathcal{F}$. For an overview on complexity theory, see for example the textbooks [HO02, HU79, Pap94, Vol99].

Complexity on a Turing machine

In this thesis we use the standard complexity classes LOGSPACE and NL . For the following definitions we assume the input to have length n A formal language L is in LOGSPACE if there exists some deterministic Turing machine with a read only input tape and a $\mathcal{O}(\log n)$ space-bounded work tape deciding L. Cook introduced in [Coo85] functional complexity classes $\mathcal{C}^*$ as the NC^1 closure of the complexity class $\mathcal{C}$. For the NC^1 closure see the paragraph on Boolean circuits. FL is the class of functions computable by a deterministic Turing machine with a read only input tape, a $\mathcal{O}(\log n)$ space-bounded work tape and a write only output tape. We do have

Lemma 2.2.1. $\mathsf{FL} = \mathsf{NC}^1(\mathsf{LOGSPACE}) = \mathsf{LOGSPACE}^*$

The complexity class NL consists of languages decidable by a non-deterministic Turing machine with a read only input tape and a $\mathcal{O}(\log n)$ space-bounded work tape. Let Σ be some finite alphabet. A function $f : \Sigma^* \to \Sigma^*$ is in FNL if the language $L = \{(x, i, b) \mid \text{The } i\text{-th symbol of } f(x) \text{ is } b\}$ is decidable in NL. For FNL we have

Lemma 2.2.2 ([ABJ95])**.** $\mathsf{FNL} = \mathsf{FL}^{\mathsf{NL}} = \mathsf{NC}^1(\mathsf{NL}) = \mathsf{NL}^*$

Oracle Turing machines.

Turing machines can be equipped with an oracle for any language L. An *oracle Turing machine* has an extra *oracle tape* , which is write-only and can be written on from left to right. Let w be the word on the oracle tape. The Turing machine

queries the oracle for $w \in L$. The oracle deletes the content of the oracle tape and writes its answer onto the oracle tape. Let $\mathcal{C}$ be a complexity class defined in terms of Turing machines and $\mathcal{D}$ be a set of languages. We write $K \in \mathcal{C}^{\mathcal{D}}$ if K can be decided by a Turing machine in $\mathcal{C}$ which additionally uses a finite subset of languages in $\mathcal{D}$ as oracle.

One has to be careful, if the Turing machine is non-deterministic. We use the Ruzzo-Simon-Tompa oracle access mechanism, which demands the whole query to the oracle to be deterministic [RST84]. More precisely, the Turing machine has to behave deterministically from the first symbol written onto the oracle tape until the query to the oracle.

We will use the following well known fact.

Lemma 2.2.3. $\mathsf{LOGSPACE}^{\mathsf{LOGSPACE}} = \mathsf{LOGSPACE}$ *and* $\mathsf{NL}^{\mathsf{LOGSPACE}} = \mathsf{NL}$.

Boolean circuits

A *circuit* is a directed acyclic graph. The vertices are called gates. The in degree (respectively out degree) of a vertex is called *fan-in* (respectively *fan-out*). There are basically three types of gates. The input gates have fan-in 0 and fan-out 1, the output gates have fan-in 1 and fan-out 0. All inner nodes have fan-in > 0 and fan-out > 0. For Boolean circuits the boolean functions $\mathsf{AND}\ \wedge, \mathsf{OR}\ \vee$ and $\mathsf{NOT}\ \neg$ are allowed as inner nodes. The gates AND and OR can have unbounded fan-in or bounded fan-in two. The size of the circuit is the number of nodes and the depth is the length of a longest path from an input to an output node. Every *Boolean circuit* C describes a boolean function $f_C : \mathbb{B}^n \to \mathbb{B}$, where $f_C(x_1, \ldots, x_n)$ is the result of the circuit with $x_1, \ldots, x_n$ attached to the input gates. We define the circuit classes NC^i and AC^i as follows. NC^i, $i \geq 0$, is the class of Boolean circuits with polynomial size and depth $\mathcal{O}(\log^i(n))$, where all gates have bounded fan-in two. Similarly, AC^i, $i \geq 0$, is the class of Boolean circuits with polynomial size and depth $\mathcal{O}(\log^i(n))$, where all gates have unbounded fan-in. Furthermore, one obtains the classes TC^i from AC^i by adding MAJORITY gates to the circuits. The gate MAJORITY has unbounded fan-in and decides, if the majority of the input bits is one.

A *circuit family* is a sequence $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$ of circuits, where each circuit C_n has exactly n input gates. Sometimes, there are limitations on the number of inputs, like n has to be a square number. We then define that C_n is the constant 0 function, if n is not a square number. In general, it is not enough to only describe the existence of a circuit family for a given problem. The circuits should be constructible in some manner. For this, we define DLOGTIME , the direct connection language and the extended connection language. A DLOGTIME Turing machine is a deterministic Turing machine with an input tape and a finite number

of work tapes. The time bound for this Turing machine is $\mathcal{O}(\log n)$ and the space bound for the work tapes is $\mathcal{O}(\log n)$ for all work tapes altogether. Additionally the Turing machine is equipped with an address tape. Writing $\mathrm{bin}(i)$ on the address tape gives access to the i-th symbol of the input or *undefined* if i is not in the range of the input.

For the *direct connection language* let $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ be a circuit family. Then the direct connection language $L_{\mathrm{DC}}(\mathcal{C})$ of the circuit family $\mathcal{C}$ is the set of tuples $\langle y, g, p, b\rangle$ where for $n = |y|$

- g is a number of a gate v in C_n and
- for $p \in \mathbb{B}^*$:
  - if $p = \varepsilon$, then b is the type of gate v,
  - if $p = \mathrm{bin}(k)$, then b is the number of the k-th predecessor gate to v.

The *extended connection language* is defined similar. Let $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ be a circuit family with fan-in at most 2. The difference is, that $p \in \mathbb{B}^*$ is used to describe a path. Let v be a vertex and $p = 0$ (respectively $p = 1$), then $p(v)$ is the left (respectively right) predecessor of v. For $p = ap'$, we inductively define $p(v) = a(p'(v))$. Now, the extended connection language $L_{\mathrm{EC}}(\mathcal{C})$ of the circuit family $\mathcal{C}$ is the set of tuples $\langle y, g, p, b\rangle$ where for $n = |y|$

- g is a number of a gate v in C_n and
- for $p \in \mathbb{B}^*$ with $|p| \leq \log n$:
  - if $p = \varepsilon$, then b is the type of gate v,
  - if $p \neq \varepsilon$, then b is the number of the gate $p(v)$.

A circuit family $\mathcal{C}$ of constant-depth circuits is DLOGTIME-*uniform*[1] if its direct connection language $L_{\mathrm{DC}}(\mathcal{C})$ is recognized in DLOGTIME. A circuit family $\mathcal{C}$ of logarithmic-depth, bounded fan-in circuits is DLOGTIME-*uniform* if its extended connection language $L_{\mathrm{EC}}(\mathcal{C})$ is recognized in DLOGTIME.

Similarly, a circuit family is LOGSPACE-*uniform*, if its direct connection language L_{DC} is in LOGSPACE. Equivalently, a circuit family is LOGSPACE-uniform, if there is a logspace machine, which on input 1^n computes the circuit C_n (for details see e.g. [Vol99]). Whenever we write a circuit family is *uniform*, we mean DLOGTIME-uniform for $\mathsf{AC}^0, \mathsf{TC}^0$ and NC^1 and LOGSPACE-uniform for $\mathsf{AC}^i, \mathsf{NC}^{i+1}$ with $i \geq 1$. We also shorten the uniformity of a circuit by $\mathsf{uAC}^i, \mathsf{uNC}^i$ and uTC^i.

Sometimes, we express functions $f : \mathbb{B}^n \to \mathbb{B}^m$ by circuits. We say f is in (uniform) AC^i (respectively TC^i, NC^i) if each output bit of f is in (uniform) AC^i

[1] Originally defined in [BIS90].

(respectively TC^i, NC^i). A function f is AC^0-(Turing-)reducible to a function f', if f is computable by a uniform AC^0-circuit with gates for the function f'. The gates for f' are then called *oracle gates* and we write $f \in \mathsf{AC}^0(f')$. We can also use oracle gates from a set of languages (respectively functions) $\mathcal{C}$. We write $L \in \mathsf{AC}^0(\mathcal{C})$ if L is computable by an AC^0-circuit using a finite subset of $\mathcal{C}$ as oracle gates. Similarly, we say $L \in \mathsf{NC}^1(\mathcal{C})$ if L is computable by an NC^1-circuit using a finite subset of $\mathcal{C}$ as oracle gates. Note that in case of NC^1 the oracle gate counts as depth $\log n$ if it has n input bits. For more details on (Boolean) circuit complexity see e.g. the textbook [Vol99].

Boolean circuits and first-order formulae

There is a tight relationship between the complexity classes described by uniform Boolean circuits and *first-order* formulae. Therefore, it is often convenient to show the existence of a uniform circuit by giving a first-order formula. A detailed introduction is given in [BIS90]. In this setting first-order includes the following predicates and constants:

- the minimal and maximal positions in the input **1** and **n**,
- the BIT predicate, where $BIT(i, j)$ holds iff the i-th bit of $\mathrm{bin}(j)$ is one,
- the predicate X, where $X(i)$ is the i-th bit of the input
- the predicate $=$ and $\leq$ on numbers.

We denote first-order with these predicates by FO . In particular, the following results follow from [BIS90, Theorem 9.1]. First of all, we have $\mathsf{FO} = \mathsf{AC}^0$. By $\#i : \varphi(i)$ we denote the counting operator. We define $\#i : \varphi(i)$ to be the number of i that satisfy $\varphi(i)$. The problem BCount of counting the ones in a binary sequence is uTC^0-complete, see for example [Vol99].

Problem: BCount
Input: $a_1, \ldots, a_n \in \{0, 1\}$
Output: $\sum a_i$ in binary notation.

Hence the counting operator can be realized in uTC^0. Thus, uniform TC^0 equals first-order with counting operator, that is $\mathsf{uTC}^0 = \mathsf{uAC}^0(\mathrm{BCount}) = \mathsf{FO}[\#]$. Moreover, verifying that some number is the minimum of some set or selecting a certain element satisfying a property can easily be expressed using first order logic. Let $P(i)$ be some property, then $k = \min\{i \mid P(i)\}$ if $P(k) \wedge \forall j : (j \geq k) \vee \neg P(j)$ and selecting the j-th bit of the only element X_k satisfying P is $\bigvee_k P(k) \wedge (X_k)_j$. Thus, these two properties are in uniform AC^0. We will use them frequently in this thesis.

Arithmetic circuits

Arithmetic circuits are similar to Boolean circuits. The only difference are the gates, which in case of arithmetic circuits are $+$ for addition and $\times$ for multiplication of integers. The input gates are allowed to have $-1, 0$ and 1 as input. Thus the function corresponding to an arithmetic circuit $\mathcal{C}$ with n input gates is $f_C : \{-1, 0, 1\}^n \to \mathbb{Z}$. Uniformity for arithmetic circuits is defined the same way as uniformity for Boolean circuits. The class GapNC^1 is the class of $\mathsf{DLOGTIME}$-uniform arithmetic circuits where $+$ and $\times$ have fan-in two, the circuit has polynomially many gates in n and its depth is $\mathcal{O}(\log n)$. It was first introduced in [CMTV98] and shown to be equivalent to the above definition. More on arithmetic circuits may be found in [All04, DMR$^+$12]. In most cases we are interested in decision problems. The class $\mathsf{C_=NC}^1$ is the decision variant of GapNC^1 and defined as

$$\mathsf{C_=NC}^1 = \{L \subseteq \{0, \pm 1\}^* \mid \exists f \in \mathsf{GapNC}^1 \; \forall w \in \{0, \pm 1\}^* : w \in L \iff f(w) = 0\}.$$

To express a language in $\mathsf{C_=NC}^1$ as a circuit we introduce a test gate "$= 0$?", which outputs 1 if the input equals 0 and 0 otherwise. The class $\mathsf{C_=NC}^1$ consists of uniform arithmetic circuits with polynomially many processors, depth $\mathcal{O}(\log n)$ and with additionally one "$= 0$?" test gate as output gate. In the following we state some useful properties of $\mathsf{C_=NC}^1$.

Lemma 2.2.4 ([CMTV98])**.** *The complexity class* $\mathsf{C_=NC}^1$ *is closed under*

(a) finite conjunctions, and

(b) finite disjunctions.

Consequently, $\mathsf{C_=NC}^1$ *is closed under union and intersection.*

Proof. (a) Let $L_1, \ldots, L_k \in \mathsf{C_=NC}^1$ with functions f_i corresponding to L_i. Then, by definition $g(w) = \sum_{i=1}^{k} f_i(w)^2$ is in GapNC^1 and $g(w) = 0 \iff w \in L_i$ for all i. For (b), let $L_1, \ldots, L_k \in \mathsf{C_=NC}^1$ and $h(w) = \prod_{i=1}^{k} f_i(w)$. Then $h(w) = 0$ if and only if there exists an i with $f_i(w) = 0$ and therefore, $w \in L_i$. □

Lemma 2.2.5 ([CMTV98])**.** *We do have* $\mathsf{NC}^1 \subseteq \mathsf{C_=NC}^1 \subseteq \mathsf{LOGSPACE}$.

As a direct consequence we obtain that $\mathsf{C_=NC}^1$ is closed under NC^1 many-one reductions. The next result is important for us and characterizes $\mathsf{C_=NC}^1$ in terms of some complete problem. For this completeness result we state the defintion of $\mathsf{DLOGTIME}$ *reducibility* as stated in [BIS90, Section 7]. A function f is a $\mathsf{DLOGTIME}$ *many-one reduction* of some language L to some language L', if f increases the length of strings only polynomially and the predicate $A_f(c, i, w)$ meaning ´the i-th symbol of $f(w)$ is c' is recognized by some $\mathsf{DLOGTIME}$ Turing machine.

Theorem 2.2.6 ([CMTV98]). *Computing a specific entry of an iterated product of integer matrices with constant dimension $k \geq 3$ encoded in binary (*ItMatProd$_N$*) is complete for* GapNC^1 *under* DLOGTIME *reductions*[2]*. Verifying a specific entry in the result of this matrix product (*V-ItMatProd$_N$*) is complete for* C$_=$NC^1 *under* DLOGTIME *reductions*[2]*.*

Since C$_=$NC^1 is closed under intersection, we obtain that verifying the resulting matrix of the iterated product is in C$_=$NC^1. We sketch how to compute the iterated product in GapNC^1. Computing the product of two matrices (with fixed dimension) is possible with an arithmetic circuit with constant depth, since only a fixed number of multiplications and additions are necessary. A simple divide and conquer approach then leads to an arithmetic circuit with logarithmic depth for the computation of the iterated matrix product.

Completeness for dimension $k = 2$ is an open problem. By the above the computation (respectively verification) of the iterated matrix product is contained in GapNC^1 (respectively C$_=$NC^1). However, it is only known to be hard for boolean NC^1 under DLOGTIME reductions by Robinson [Rob93].

The following chain shows the known inclusion relation of the first level of the parallel hierarchy.

$$\mathsf{AC}^0 \subsetneq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{C}_=\mathsf{NC}^1 \subseteq \mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1) \subseteq \mathsf{LOGSPACE}$$

Thereby, only the first inclusion is known to be strict.

Counting complexity classes

The idea behind the *counting complexity* classes is to count the number of accepting (and rejecting) paths in the computation of a non-deterministic Turing machine. We have seen in the previous subsection that computing (respectively verifying) the product of a sequence of integer matrices with a fixed dimension is GapNC^1- (respectively C$_=$NC^1-)complete. Our main interest in this subsection is the complexity of computing and verifying the product of a sequence of $n \times n$ integer matrices, where n depends on the input.

Let M be a non-deterministic, polynomially time-bounded Turing machine over the finite alphabet Σ. We define the following functions

$$\begin{aligned} \mathrm{accept}_M : \Sigma^* \to \mathbb{N}, &\quad \mathrm{accept}_M(w) = \text{number of accepting paths of } w \text{ in } M, \\ \mathrm{reject}_M : \Sigma^* \to \mathbb{N}, &\quad \mathrm{reject}_M(w) = \text{number of rejecting paths of } w \text{ in } M, \\ \mathrm{gap}_M : \Sigma^* \to \mathbb{Z}, &\quad \mathrm{gap}_M(w) = \mathrm{accept}_M(w) - \mathrm{reject}_M(w). \end{aligned}$$

[2] Actually, the problem is complete under the stronger "uniform projections" reduction. However, for our case complete under DLOGTIME reductions is sufficient.

In this thesis we consider only counting complexity classes over non-deterministic, polynomially time bounded and logarithmically (in the input) space-bounded Turing machines. With the above definition, the counting complexity classes can be described as

$$\#\mathsf{L} = \left\{ \text{accept}_M \;\middle|\; \begin{array}{c}\text{M a non-deterministic, polynomially time-bounded,}\\ \text{logarithmic space-bounded Turing machine}\end{array} \right\}$$

$$\mathsf{GapL} = \left\{ \text{gap}_M \;\middle|\; \begin{array}{c}\text{M a non-deterministic, polynomially time-bounded,}\\ \text{logarithmic space-bounded Turing machine}\end{array} \right\}$$

$$\mathsf{C_{=}L} = \left\{ L \subseteq \Sigma^* \;\middle|\; \begin{array}{c}\text{There exists } f \in \mathsf{GapL} \text{ such that}\\ \forall w \in \Sigma^* : w \in L \iff f(w) = 0\end{array} \right\}$$

There are some alternative characterizations of GapL. They can be expressed easily with the following notion. Let $\mathcal{C}$, $\mathcal{D}$ be function classes, then the difference of the two classes is

$$\mathcal{C} - \mathcal{D} = \{f - g \mid f \in \mathcal{C}, g \in \mathcal{D}\}.$$

Lemma 2.2.7 ([AO94])**.** *The function class* GapL *can be characterized by*

$$\mathsf{GapL} = \#\mathsf{L} - \#\mathsf{L} = \#\mathsf{L} - \mathsf{FL} = \mathsf{FL} - \#\mathsf{L}.$$

This characterization of GapL immediately yields $\#\mathsf{L} \subseteq \mathsf{GapL}$ and $\mathsf{LOGSPACE} \subseteq \mathsf{C_{=}L}$. Counting the number of $s \xrightarrow{*} t$ paths in a graph is contained in #L. The graph accessibility problem GAP is that there is at least one $s \xrightarrow{*} t$ path. Thus, its complement $\overline{\text{GAP}}$ is contained in $\mathsf{C_{=}L}$ and thus we have $\mathsf{NL} \subseteq \mathsf{C_{=}L}$ since $\overline{\text{GAP}}$ is NL-complete [Sav70].

The complexity class GapL enjoys many useful properties. We mention some of them in the following lemma.

Lemma 2.2.8 ([AO94])**.** *All of the following properties do hold for* GapL*:*

1. *Let $f : \Sigma^* \to \mathbb{Z}$ be a (total) logspace computable function. Then $f \in \mathsf{GapL}$.*
2. *Let $f \in \mathsf{GapL}$ and $g : \Sigma^* \to \Sigma^*$ be logspace computable. Then $(f \circ g) \in \mathsf{GapL}$.*
3. *Let $f, g \in \mathsf{GapL}$, then $(f + g)$ and $(f \cdot g)$ is in* GapL*.*

Some important problems of linear algebra are complete for GapL. We mention two of them in the following lemma.

Lemma 2.2.9 ([Vin91a, Vin91b, Tod91, Tod92, Dam91])**.** *The following problems are complete for* GapL *under logspace reductions.*

1. DET: *Given an $n \times n$ matrix A. Output* $\det(A)$.

2. MATPOWELEMENT: *Given an $n \times n$ integer matrix A, $1 \leq i, j \leq n$, $m \geq 0$ and $x \in \mathbb{Z}$. Is $(A^m)_{ij} = x$?*

Often, the problem MATPOW is listed for the GapL complete problems. The question is, given an $n \times n$ matrix A and an integer $m \geq 0$, what is A^m? Complete in this context means that the matrix A^m can be computed by n^2 many GapL functions, that is one function for each entry of the matrix A^m.

In many cases, one is not only interested in the computation of some function, but in verifying its result. Let f be any function. By V-f we denote the graph of f, that is

$$\text{V-}f = \{(x, y) \mid f(x) = y\}.$$

Complete functions for GapL automatically give complete sets for $\mathsf{C}_=\mathsf{L}$. In the context of GapL we say a function f' reduces to f, if there exist a logspace computable function g with $f'(x) = f(g(x))$. We use this definition for GapL completeness.

Theorem 2.2.10. *Let $f : \Sigma^* \to \mathbb{Z}$ be* GapL *complete. Then the verification language*

$$\text{V-}f = \{(x, y) \mid f(x) = y\}$$

of f is $\mathsf{C}_=\mathsf{L}$*-complete.*

Proof. We need to show that V-f is in $\mathsf{C}_=\mathsf{L}$ and that V-f is hard for $\mathsf{C}_=\mathsf{L}$. Let L be any language in $\mathsf{C}_=\mathsf{L}$. By definition, there exists a function $f_L \in \mathsf{GapL}$ with $w \in L \iff f_L(w) = 0$ for all $w \in \Sigma^*$. Since f is GapL complete, we have for any function $f' \in \mathsf{GapL}$ that $f' \leq f$. In particular $f_L \leq f$. Hence, there exists a logspace computable function $g : \Sigma^* \to \Sigma^*$ with $f_L(w) = f(g(w))$ for any $w \in \Sigma^*$. This leads to

$$w \in L \iff f_L(w) = 0 \iff f_L(w) = f(g(w)) = 0 \iff (g(w), 0) \in \text{V-}f.$$

The tuple $(g(w), 0)$ is logspace computable since $g(w)$ is.

It remains to show V-$f \in \mathsf{C}_=\mathsf{L}$. By the theorem above GapL is closed under addition. Hence, the function $f_{\text{V}} : \Sigma^* \times \mathbb{Z}$ with $f_{\text{V}}(x, y) = f(x) - y$ is in GapL. □

The complexity class $\mathsf{C}_=\mathsf{L}$ is closed under different kinds of reductions. We list them in the following theorem. It is unknown, whether $\mathsf{C}_=\mathsf{L}$ is closed under AC^0 Turing reductions. That is $\mathsf{AC}^0(\mathsf{C}_=\mathsf{L})$ might be more powerful than $\mathsf{C}_=\mathsf{L}$, since $\mathsf{C}_=\mathsf{L}$ is obviously contained in $\mathsf{AC}^0(\mathsf{C}_=\mathsf{L})$. One important property of complexity classes is the closure under complement. For $\mathsf{C}_=\mathsf{L}$ the question, whether $\mathsf{C}_=\mathsf{L} \stackrel{?}{=} \mathsf{coC}_=\mathsf{L}$ is an open problem.

Lemma 2.2.11 ([AO94]). $\mathsf{C_=L}$ *is closed under* $\leq_m^{\mathsf{FNL}}$, $\leq_{ctt}^{\mathsf{FNL}}$ *and* $\leq_{dtt}^{\mathsf{FNL}}$ *reductions.*

Theorem 2.2.12 ([ABO99]). $\mathsf{AC^0(C_=L)} = \mathsf{NC^1(C_=L)} = \mathsf{LOGSPACE}^{\mathsf{C_=L}}$.

Moreover, we will frequently use the following fact.

Lemma 2.2.13. $\mathsf{C_=L}^{\mathsf{LOGSPACE}} = \mathsf{C_=L}$.

Ranging on the same level of complexity is the computation of the determinant of an $n \times n$ integer matrix. In the following, we are only considering integer matrices unless otherwise stated. The class DET^* was originally introduced by Cook in 1985 [Coo85]. He defined the class DET^* as the set of functions, which are NC^1 reducible to the determinant function Det. It turns out that

Theorem 2.2.14 ([Vin91b], [Tod91]).

$$\mathsf{DET}^* = \mathsf{NC}^1(\#\mathsf{L}) = \mathsf{NC}^1(\mathsf{GapL})$$

Proof. The first equality is due to [Vin91b], [Tod91]. The second equality is due to the fact that $\mathsf{GapL} = \#\mathsf{L} - \#\mathsf{L}$. The values computed by $\#\mathsf{L}$ are at most exponential, thus their binary representation requires at most $n^{\mathcal{O}(1)}$ many bits. Computing the difference of two $n^{\mathcal{O}(1)}$-bit integer numbers is in NC^1. □

A similar class is DET , which is defined as the set of functions, which are logspace many-one reducible to the computation of the determinant. This difference led to some confusion and erroneous results, see [MV97]. Also, in the literature, DET^* is often denoted by DET. For the class DET, it turns out that

Theorem 2.2.15 ([MV97, Vin91a, Vin91b, Tod91, Tod92, Dam91]).

$$\mathsf{DET} = \mathsf{GapL}.$$

A complete proof of this result is given in [MV97]. This result was independently discovered by Vinay, Damm and Toda (see [Vin91a, Vin91b, Tod91, Tod92, Dam91]). One should be cautious with these papers, as they contain mistakes, see [MV97, Section 6.2] for details.

The verification of the result of some function was also of interest of [ST98]. Using the definition of V-f we obtain the verification of the determinant as

$$\text{V-Det} = \{(A, d) \mid \det(A) = d\}.$$

The complexity class $\mathsf{V\text{-}DET}$ is defined as the closure of V-Det under $\mathsf{AC^0}$ Turing reductions, that is $\mathsf{V\text{-}DET} = \mathsf{AC^0}(\text{V-Det})$.

Lemma 2.2.16. $\mathsf{V\text{-}DET} = \mathsf{AC^0(C_=L)}$

Proof. It is shown in [ST98] that the AC^0 many-one closure of V-DET equals $\mathsf{C_=L}$. Hence, by definition $\mathsf{V\text{-}DET} = \mathsf{AC}^0(\mathsf{C_=L})$. □

Lemma 2.2.17. *The following problems are complete for* $\mathsf{C_=L}$ *under logspace many-one reductions.*

1. V-DET*: Given an* $n \times n$ *integer matrix* A *and* $d \in \mathbb{Z}$*. Is* $\det(A) = d$*?*
2. V-MATPOW*: Given* $n \times n$ *integer matrices* A, B *and an* $m \geq 0$*. Is* $A^m = B$*?*
3. V-ITMATPROD*: Given* $n \times n$ *integer matrices* $A_1, \ldots, A_m$ *and a matrix* B*. Is* $A_1 \cdots A_m = B$*?*

Proof. V-DET is complete for $\mathsf{C_=L}$, since DET is complete for GapL.

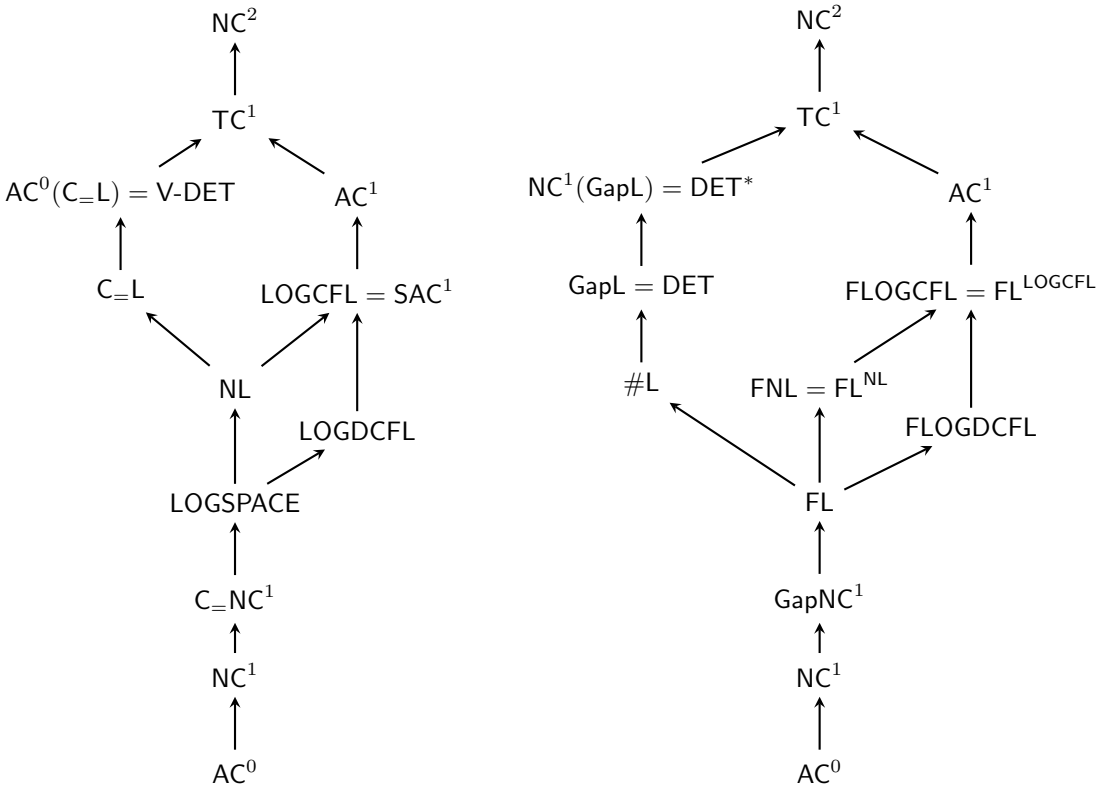
We first show the equivalence of *2* and *3*. The problem V-MATPOW is obviously logspace reducible to V-ITMATPROD. Hence, it remains to reduce the decision problem V-ITMATPROD to V-POWER. We follow the idea in [Coo85]. Let $A_1, \ldots, A_n$ be $n \times n$ matrices over $\mathbb{Z}$. We construct a $(n^2+n) \times (n^2+n)$ matrix A consisting of $n \times n$ blocks. The blocks are all zero except for those above the blocks on the diagonal. They are filled with $A_1, \ldots, A_n$. The n-th power A^n is the zero matrix, except for the block in the upper right corner. It contains $A_1 \cdots A_n$.

Now, for case *2* containment of the problem V-MATPOW in $\mathsf{C_=L}$ is as follows. The problem MATPOWELEMENT is complete for GapL and hence, the decision problem V-MATPOWELEMENT is complete for $\mathsf{C_=L}$. Since $\mathsf{C_=L}$ is closed under conjunctive truth table queries, we take n^2 queries to the verification problem V-MATPOWELEMENT. Each query corresponds to one entry in the resulting matrix A^m. Thus, we have V-MATPOW $\in \mathsf{C_=L}$. For the hardness, we follow the proof idea of [ST98] and reduce V-MATPOWELEMENT to V-MATPROD. Let $Z_{k,l}$ be an $n \times n$ matrix, with all entries 0, except for the (k,l)-th element, which is 1. The product $B = Z_{1,i} \cdot A^m \cdot Z_{j,n}$ is the zero matrix, except for the (i,j)-th element. The (i,j)-th entry of B equals the (i,j)-th entry of A^m, which completes the proof. □

The containment relation of all the logspace counting classes and determinant classes is depicted in Figure 2.1 on page 26 below. The equalities are shown above. All the inclusions from top to bottom are obvious, since $\mathsf{AC}^0 \subsetneq \mathsf{NC}^1$, but it is unknown, whether they are strict. Recall that a decision problem is a function problem via its characteristic function.

$$
\begin{array}{ccccccccc}
 & & \mathsf{C_{=}L} & & \#\mathsf{L} & \subseteq & \mathsf{GapL} & & \\
 & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \\
\mathsf{V\text{-}DET} & = & \mathsf{AC^0(C_{=}L)} & \subseteq & \mathsf{AC^0(\#L)} & = & \mathsf{AC^0(GapL)} & & \\
\| & & \| & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \\
\mathsf{LOGSPACE}^{\mathsf{C_{=}L}} & = & \mathsf{NC^1(C_{=}L)} & \subseteq & \mathsf{NC^1(\#L)} & = & \mathsf{NC^1(GapL)} & = & \mathsf{DET}^*
\end{array}
$$

Figure 2.1: Relation of counting complexity classes.



Decision classes *Functional classes*

Figure 2.2: Relation of (parallel) complexity classes. An arrow denotes containment. For details see the corresponding subsection.

2.3 Rewriting systems

A *rewriting system* on a set X is a binary relation $\Longrightarrow \subseteq X \times X$. We write $x \Longrightarrow y$, if $(x, y) \in \Longrightarrow$. The notation $x \Longrightarrow y$ means that we can rewrite x into y in one step. By $\overset{*}{\Longrightarrow}$ we denote the reflexive and transitive closure of $\Longrightarrow$. By $\overset{*}{\Longleftrightarrow}$ we denote the reflexive, transitive and symmetric closure of $\Longrightarrow$. We also write $y \Longleftarrow x$ instead of $x \Longrightarrow y$. We say the rewriting system is

- *confluent*, if $y \overset{*}{\Longleftarrow} x \overset{*}{\Longrightarrow} z$ implies $\exists w \in X : y \overset{*}{\Longrightarrow} w \overset{*}{\Longleftarrow} z$.
- *terminating*, if there are no infinite chains

$$x \Longrightarrow x_1 \Longrightarrow \ldots \Longrightarrow x_i \Longrightarrow \ldots$$

- *convergent*, if it is confluent and terminating.

The idea of a convergent rewriting system is that it does not matter in which order a rewriting process is performed.

2.4 Presentations of monoids and groups

The basic idea is to express monoid and group elements as words over some alphabet Σ. Thereby Σ is some arbitrary set. The union $\Sigma^* = \bigcup_{k \in \mathbb{N}} \Sigma^k$ is called the free monoid with multiplication

$$(a_1 \cdots a_k) \cdot (b_1 \cdots b_l) = (a_1 \cdots a_k b_1 \cdots b_l)$$

for $a_1 \cdots a_k, b_1 \cdots b_k \in \Sigma^*$. The neutral element of Σ^* is the empty word. We will denote the neutral element by 1.

Let M be some monoid and $S \subseteq M \times M$ be a relation over M. Instead of $(\ell, r) \in S$ we also write $\ell \longrightarrow r$. The relation S over M defines a rewriting system $\underset{S}{\Longrightarrow}$ through

$$x \underset{S}{\Longrightarrow} y \text{ if and only if } x = p\ell q, y = prq \text{ and } (\ell, r) \in S.$$

The relation $\overset{*}{\underset{S}{\Longleftrightarrow}} \subseteq M \times M$ is a congruence. We denote by $M/S = \{[x] \mid x \in M\}$ the set of equivalence classes $[x] = \{y \mid x \overset{*}{\underset{S}{\Longleftrightarrow}} y\}$. Therefore, M/S is a monoid with multiplication $[x] \cdot [y] = [xy]$. We call S confluent (respectively terminating, convergent) if $\underset{S}{\Longrightarrow}$ is confluent (respectively terminating, convergent). For $M = \Sigma^*$ we call S a semi-Thue system. Let N be some monoid. We say N is

- *finitely generated* (short *f.g.*), if N is presented by some Σ^*/S with Σ finite,
- *finitely presented* (short *f.p.*), if N is presented by some Σ^*/S with Σ and S finite.

Let $N = \Sigma^*/S$. We write $u =_N v$ if $[u] = [v]$ for words $u, v \in \Sigma^*$. Sometimes we write "$u = v$ in N" instead of $u =_N v$. Otherwise $u = v$ denotes the equality of words. We call Σ a generating set for N and $a \in \Sigma$ a generator.

Example 2.4.1. *Let Σ be an alphabet and $\Sigma^{\pm} = \Sigma \uplus \overline{\Sigma}$. Moreover, let $a \mapsto \bar{a}$ be an involution with $\bar{\bar{a}} = a$ for any $a \in \Sigma$. Then $F_\Sigma = (\Sigma^{\pm})^*/\{a\bar{a} \to 1 \mid a \in \Sigma^{\pm}\}$ is the free group generated by Σ.*

The free group is the basis for the presentation of groups. Let G be a group generated by $\Sigma \subseteq G$ as a group, then there exists some normal subgroup N with $G = F_\Sigma/N$. For some subset $\mathcal{R} \subseteq F_\Sigma$, where N is the normal closure of $\mathcal{R}$ we write

$$G = \langle \Sigma \mid \mathcal{R} \rangle = F_\Sigma \Big/ \mathcal{R}.$$

The set $\mathcal{R}$ defines a rewriting system S in the following way. For $r \in \mathcal{R}$ there exists some $w_r \in (\Sigma^{\pm})^*$ with $r =_{F_\Sigma} w_r$. Thus, with

$$S = \{a\bar{a} \to 1, w_r \to 1 \mid a \in \Sigma^{\pm}, r \in \mathcal{R}\}$$

we have $G = F_\Sigma/\mathcal{R} = (\Sigma^{\pm})^*/S$. We call G *finitely generated* (respectively *finitely presented*) if $(\Sigma^{\pm})^*/S$ is finitely generated (respectively finitely presented). We extend the above involution $a \mapsto \bar{a}$ to words by setting $\overline{uv} = \bar{v} \cdot \bar{u}$ for words $u, v \in (\Sigma^{\pm})^*$. Therefore, we obtain $w^{-1} =_G \overline{w}$ for $w \in (\Sigma^{\pm})^*$.

2.5 Group products

In this thesis we consider direct products, free products, amalgamated products and graph products. A detailed introduction to graph products is given in Chapter 3. The *direct product* of two groups G, H is denoted by $G \times H$ and for multiple groups G_i by $\prod G_i$. The *free product* of two groups G, H is denoted by $G * H$ and for multiple groups G_i by $* G_i$. Lastly, the *amalgamated product* of two groups G, H over isomorphic subgroups $A \leq G, B \leq H$ is denoted by $G *_{A=B} H$ or by $G *_A H$. These products are defined more precisely in the following. Let $G_1 = \langle \Sigma_1 \mid \mathcal{R}_1 \rangle$ and $G_2 = \langle \Sigma_2 \mid \mathcal{R}_2 \rangle$ be groups. Then the direct product of G_1 and G_2 is

$$G_1 \times G_2 = \langle \Sigma_1, \Sigma_2 \mid \mathcal{R}_1, \mathcal{R}_2, a_1 a_2 = a_2 a_1 \text{ for } a_i \in \Sigma_i \rangle.$$

and the free product of G_1 and G_2 is

$$G_1 * G_2 = \langle \Sigma_1, \Sigma_2 \mid \mathcal{R}_1, \mathcal{R}_2 \rangle.$$

This definition can be extended to amalgamated products. Let A be a some group and $\iota_1 : A \to G_1, \iota_2 : A \to G_2$ be embeddings of A into G_1 and G_2. Then the amalgamated product of G_1 and G_2 over A is

$$G_1 *_A G_2 = \langle \Sigma_1, \Sigma_2 \mid \mathcal{R}_1, \mathcal{R}_2, \iota_1(a) = \iota_2(a) \text{ for } a \in A \rangle.$$

If G is any group product over the groups G_α, then the groups G_α are called *base groups*. For the groups G_α we set $\Gamma_\alpha = G_\alpha \setminus \{1\}$. For any group product G over the groups G_α we let $\Gamma = G \setminus \{1\}$. We call a decomposition $w = w_1 \cdots w_n$ of a word $w \in \Gamma^*$ a *factorization*, if for every i there is some α_i with $w_i \in \Gamma_{\alpha_i}$. The length of w is denoted by $|w| = n$ if $w = w_1 \cdots w_n$ is a factorization of w. The α-length $|w|_\alpha$ of w is the number of $w_i \in \Gamma_\alpha$. The *geodesic length* of w is denoted by $\|w\|$, where

$$\|w\| = \min \{k \mid w =_G w_1 \cdots w_k \text{ with } w_1 \cdots w_k \text{ is a factorization}\}.$$

A word $w \in \Gamma^*$ is called *geodesic* if $\|w\| = |w|$. For direct, free and graph products we can define the geodesic α-length. Let $u \in \Gamma^*$ with $w =_G u$ and u geodesic. Then the geodesic α-length of w is $\|w\|_\alpha = |u|_\alpha$. The geodesic-length and geodesic alpha-length is unique for direct, free and graph products. We write $\mathrm{alph}(u) = \alpha$ for $u \in \Gamma_\alpha$. For words $w \in \Gamma^*$ with factorization $w = w_1 \cdots w_n$ we denote by $\mathrm{alph}(w)$ the set $\{\mathrm{alph}(w_i) \mid 1 \leq i \leq n\}$.

Chapter 3

Graph products

Graph products were originally introduced by Elisabeth Ruth Green in her PhD thesis [Gre90] as an extension of graph groups. Another similar concept are trace monoids, also known as free partially commutative monoids. They were introduced by Mazurkiewicz [Maz77, Maz87]. The connection of trace monoids to dependence graphs was initiated and established by R. Keller [Kel73].

Graph products are, like free and direct products, a product over some base groups. Moreover, graph products generalize direct and free products. Some special cases of graph products are right angled Artin groups and right angled Coxeter groups. Right angled Artin groups are also known as graph groups. The term 'graph group' was established by Carl Droms [Dro87] and emphasizes their description by graphs.

Graph products preserve important properties in algorithmic group theory, like having a solvable word problem, having a solvable conjugacy problem, having an automatic structure [HM95] and being residually finite [Gre90].

While a solution to the word problem is preserved, this is not the case for the more general problem of the generalized word problem. Already the generalized word problem of the graph group $(F_2 \times F_2)$ is undecidable by Mihailova [Mih58]. In this thesis, we do not need the full power of the generalized word problem in graph products. For the restricted case we need, graph products offer an efficient solution (see Lemma 5.6.3). Linear groups are residually finite and graph products do preserve this property. However, it is not known if a graph product of linear groups is linear again. Being linear is trivially true for direct products of linear groups. Also free products of linear groups are linear again, which was shown by [Weh73, Mar85]. For right angled Coxeter groups a linear embedding is known, see Section 3.2. Right angled Artin groups are linear as well since they do embed into right angled Coxeter groups. As stated above, linearity of graph products of linear groups is an open problem. However, we can show that any graph product embeds into the automorphism group over some infinite dimensional vector space (see Section 3.3). Fortunately, for the algorithmic problems considered in this thesis only a finite dimensional subspace has to be considered. The embedding

also gives a new and elementary proof that graph products over finite groups are linear.

3.1 Definition

In this section we first define trace monoids and some basic concepts and related notation. Based on trace monoids we then define graph groups and graph products.

3.1.1 Trace monoids

We first define trace monoids and related terms. Trace theory is useful for studying graph groups and graph products, since as soon as we deal with geodesics we can apply trace theory.

Let Σ be some finite alphabet and $I \subseteq \Sigma \times \Sigma$ a symmetric and irreflexive relation. The letter I of the relation refers to *independence relation*. The complement of this relation is the *dependence relation* $D = (\Sigma \times \Sigma) \setminus I$. Now, the *trace monoid* $M(\Sigma, I)$ is given by

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

Often the independence relation is depicted as the independence graph, where Σ is the set of vertices and two vertices $a, b \in \Sigma$ are adjacent if and only if $(a, b) \in I$. Similarly, the dependence relation can be depicted as the dependence graph, where again Σ is the set of vertices and two vertices $a, b \in \Sigma$ are adjacent if and only if $(a, b) \in D$.

An element of the trace monoid is called *trace*. Let $M = M(\Sigma, I)$ be a trace monoid and $w = a_1 \cdots a_n$ with $a_i \in \Sigma$ be a word representing a trace. Then we can assign a directed acyclic graph to the word w. The *dependence graph* of w is $D(w) = (V, E, \lambda)$ with $V = \{v_1, \ldots, v_n\}$ and $\lambda(v_i) = a_i$. The edge set is given by $(v_i, v_j) \in E$ if and only if $i < j$ and $(\lambda(v_i), \lambda(v_j)) \in D$. Contrary, if we have any graph with labeling function $\lambda : V \to \Sigma$ satisfying the condition on the edges, then we can uniquely assign a trace to it by taking any topological ordering of the graph. Thus, we do have a one-to-one correspondence of traces and dependence graphs, see [Die90, Proposition 1.2.4]. The *Hasse diagram* of the dependence graph is obtained by removing transitive edges. More precisely, the edge set of the Hasse diagram is given by $(v_i, v_j) \in E$ if and only if $i < j$ and $(\lambda(v_i), \lambda(v_j)) \in D$ and there is no k with $i < k < j$ and $(\lambda(v_i), \lambda(v_k)), (\lambda(v_k), \lambda(v_j)) \in D$. Thus, the out-degree of any vertex in the Hasse diagram is bounded by $|\Sigma|$. For two traces $u, v \in M$ we write $u \equiv v$ if their dependence graphs are isomorphic. A generator $a \in \Sigma$ is called *minimal* (respectively *maximal*) in a trace w, if there is some trace

u such that we have $w \equiv au$ (respectively $w \equiv ua$). We say a trace u is a *subtrace* (or *factor*) of a trace w, if there exists traces v_1, v_2 such that $w \equiv v_1 u v_2$. Let w be a trace which is represented by $w = a_1 \cdots a_n \in \Sigma^*$. We define the alphabet of w by $\mathrm{alph}(w) = \{a \in \Sigma \mid |w|_a \geq 1\}$. We extend the independence relation to traces by letting $(u, v) \in I$ if $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$.

For a general overview and introduction to trace theory, see for example the textbooks [Die90] and [DR95].

3.1.2 Graph groups

Graph groups arise from trace monoids by adding inverses to the generators. Let Σ be some finite alphabet and $\overline{\Sigma}$ be some disjoint copy of Σ. Further, let $I \subseteq \Sigma \times \Sigma$ a symmetric and irreflexive relation which satisfies $(a, b) \in I$ if and only if $(\bar{a}, b) \in I$. We set $\Sigma^{\pm} = \Sigma \uplus \overline{\Sigma}$. Now, the *graph group* $G(\Sigma, I)$ is defined as.

$$\begin{aligned} G(\Sigma, I) &= M(\Sigma^{\pm}, I) \Big/ \{a\bar{a} = 1 \mid a \in \Sigma^{\pm}\} \\ &\simeq \left(\Sigma^{\pm}\right)^* \Big/ \{bc = cb, a\bar{a} = 1 \mid a \in \Sigma^{\pm}, (b, c) \in I\} \\ &\simeq F(\Sigma) \Big/ \{ab = ba \mid (a, b) \in I\} \end{aligned}$$

A word $w \in (\Sigma^{\pm})^*$ (respectively a trace $w \in M(\Sigma^{\pm}, I)$ is called *reduced* or *geodesic*, if no traces $u, v \in M(\Sigma^{\pm}, I)$ and $a \in \Sigma^{\pm}$ do exist with $w \equiv ua\bar{a}v$. For graph groups we define the alphabet of some word w as

$$\mathrm{alph}(w) = \{a \in \Sigma \mid |w|_a \geq 1 \text{ or } |w|_{\bar{a}} \geq 1\}.$$

3.1.3 Graph products

For graph products we let $\mathcal{L}$ be some finite list and for each $\alpha \in \mathcal{L}$ we have some group G_α. We call the group G_α a *base group* of the graph product. Let $\langle \Sigma_\alpha \mid \mathcal{R}_\alpha \rangle$ be the presentation of G_α. We set $\Gamma_\alpha = G_\alpha \setminus \{1\}$ and $\Gamma = \biguplus_{\alpha \in \mathcal{L}} \Gamma_\alpha$. Note that we may consider Σ_α as a subset of Γ_α. Further, we set

$$I_\Gamma = \{(u, v) \mid u \in \Gamma_\alpha, v \in \Gamma_\beta \text{ for } (\alpha, \beta) \in I\}.$$

Let $u, v \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$. By $[uv]$ we denote the unique element in Γ_α equaling the product $u \cdot v$ in G_α. Using this definition, the *graph product* $\mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ can be defined as

$$\mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}}) = M(\Gamma, I_\Gamma) \Big/ \{uv = [uv] \mid u, v \in \Gamma_\alpha \text{ for some } \alpha \in \mathcal{L}\}$$

Example 3.1.1. *The direct product of some groups is the graph product of these groups with $I = \mathcal{L} \times \mathcal{L}$ as independence relation. Further, the free product of some groups is the graph product of these groups with $I = \emptyset$ as independence relation.*

A sequence $w = w_1 \cdots w_n \in \Gamma^*$ is called a *factorization* of w, if for each w_i there is some $\alpha \in \mathcal{L}$ with $w_i \in \Gamma_\alpha$. The alphabet of w is defined via this factorization as

$$\operatorname{alph}(w) = \{\alpha \in \mathcal{L} \mid \exists i : w_i \in \Gamma_\alpha\}.$$

A word $w \in \Gamma^*$ is called *reduced* or *geodesic*, if no traces $u, v \in \Gamma^*$ and $a_1, a_2 \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$ exist with $w \equiv u a_1 a_2 v$. For traces u, v we write $(u, v) \in I$ whenever we have $(u, v) \in I_\Gamma$. Let $G = \operatorname{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and $\mathcal{M} \subseteq \mathcal{L}$. The subgroup induced by the subset $\mathcal{M}$ is denoted $G_\mathcal{M} = \operatorname{GP}(\mathcal{M}, I \cap \mathcal{M} \times \mathcal{M}; (G_\alpha)_{\alpha\in\mathcal{M}})$.

Example 3.1.2. *Let G be the graph group with generating alphabet $\Sigma = \{a, b, c\}$ and independence relation $I = \{(a, b), (b, a)\}$. Then we have $G = (\langle a\rangle \times \langle b\rangle) * \langle c\rangle$. Figure 3.1 on page 34 shows two dependence graphs. On the left, the dependence graph of the word $u = ab\bar{a}ca\bar{b}$ is not reduced but the dependence graph of the word $v = bc\bar{b}a$ on the right is reduced. In the group G we have $u =_G v$. The dependence graph of u and v consists of the solid and dotted edges, whereas the Hasse diagram of u and v consists of the solid edges only.*

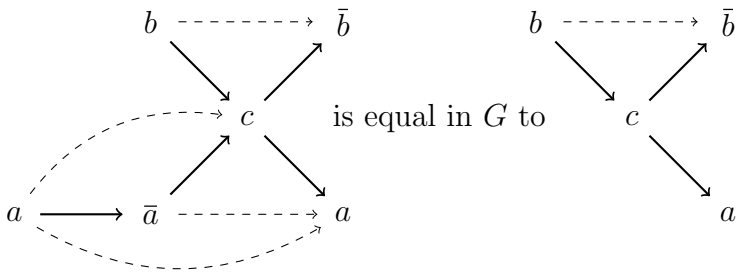


Figure 3.1: Dependence graph (Hasse diagram consists of the solid edges only) of $u = ab\bar{a}ca\bar{b}$ on the left and $v = bc\bar{b}a$ on the right (see Example 3.1.2).

In the following, let $G = \operatorname{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a fixed graph product. We define a rewriting procedure on dependence graphs. Let $D(w) = [V, E, \lambda]$ be the dependence graph of some word $w \in \Gamma^*$. As long as for some $\alpha \in \mathcal{L}$ there are vertices v_i, v_j adjacent in the Hasse diagram with $a_i = \lambda(v_i), a_j = \lambda(v_j) \in \Gamma_\alpha$ do the following.

- If $[a_i a_j] = 1$, then remove the vertices a_i, a_j from the dependence graph and their incident edges.
- If $[a_i a_j] \neq 1$, then remove the vertex v_j from the dependence graph and its incident edges. Moreover, set $\lambda(v_i) = [a_i a_j]$.

The rewriting procedure terminates after at most $|w|$ steps. Furthermore, it does not change the group element corresponding to the dependence graph. A dependence graph is called reduced, if we cannot apply any reduction step. Thus, a word $w \in \Gamma^*$ is reduced, if and only if its dependence graph is reduced.

Lemma 3.1.3 ([KL06, Lemma 6.1]). *The above rewriting procedure on dependence graphs is confluent.*

3.2 A linear embedding of graph groups

In this section, we first present the well known linear embedding of right angled Coxeter groups. We then show the construction of [HW99] to see that any graph group embeds into a right angled Coxeter groups. For the definition of right angled Coxeter groups, let Σ be some finite alphabet and I an independence relation. The associated right angled Coxeter group $CG(\Sigma, I)$ is defined as

$$CG(\Sigma, I) = M(\Sigma, I) \Big/ \{a^2 = 1, bc = cb \mid a \in \Sigma, (b, c) \in I\}$$

For $a \in \Sigma$ we define the mapping $\sigma_a : \mathbb{Z}^\Sigma \to \mathbb{Z}^\Sigma$ via

$$\sigma_a(b) = \begin{cases} -a, & \text{if } a = b \\ b, & \text{if } (a, b) \in I \\ b + 2a, & \text{if } (a, b) \in D \end{cases}$$

Let $w = a_1 \cdots a_n \in \Sigma^*$ be some word. We set $\sigma_w = \sigma_{a_1} \cdots \sigma_{a_n}$. Then the mapping $\sigma : CG(\Sigma, I) \to \mathrm{SL}(n, \mathbb{Z})$, $w \mapsto \sigma_w$ defines a linear embedding of the right angled Coxeter group. The correctness of this embedding follows by the next section. Alternatively and for more details on right angled Coxeter groups see e.g. the textbook [BB05].

In the following we give an embedding of graph groups into right angled Coxeter groups. The embedding presented here goes back to [HW99]. Let $G = G(\Sigma, I)$ be some graph group. Moreover, let $\Sigma_i = \{a_i \mid a \in \Sigma\}$ for $i = 1, 2$ be two disjoint copies of Σ. We set $\widetilde{\Sigma} = \Sigma_1 \uplus \Sigma_2$ and

$$\widetilde{I} = \{(a_i, b_j) \mid i = 1, 2 \text{ and } j = 1, 2 \text{ for } (a, b) \in I\}.$$

The graph group $G(\Sigma, I)$ embeds into the right angled Coxeter group $CG(\widetilde{\Sigma}, \widetilde{I})$ via the mapping induced by $a \mapsto a_1 a_2$ for $a \in \Sigma$.

3.3 A linear embedding of graph products

The goal of this section is to define a mapping of the graph product into some (possibly infinite-dimensional) linear group. For this section we fix the graph product $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$. Let $\Gamma_\alpha = G_\alpha \setminus \{1_\alpha\}$ for $\alpha \in \mathcal{L}$, where 1_α is the identity in G_α. Moreover, let $\Gamma = \biguplus_{\alpha\in\mathcal{L}} \Gamma_\alpha$ and $\Gamma^1 = \biguplus_{\alpha\in\mathcal{L}} G_\alpha$. Note that in Γ^1 we distinguish between the identity of $1_\alpha \in G_\alpha$ and $1_\beta \in G_\beta$ for $\alpha \neq \beta$. Before we define an embedding of G into some linear group, consider the monoid

$$M = \langle \Gamma^1 \rangle \Big/ \left\{ \begin{array}{ll} a_1a_2 = [a_1a_2] & \text{for } a_1, a_2 \in G_\alpha, \alpha \in \mathcal{L}, \\ ab = ba & \text{for } a \in G_\alpha, b \in G_\beta, (\alpha,\beta) \in I \end{array} \right\}.$$

In M we have $1_\alpha \neq 1_\beta$ for $\alpha \neq \beta$. We first define the mapping $\widetilde{\sigma}_a : \mathbb{Z}^{\Gamma_1} \to \mathbb{Z}^{\Gamma_1}$ for $a \in \Gamma_\alpha$ via the linear extension of

$$\widetilde{\sigma}_a(b) = \begin{cases} [a \cdot b] - a & \text{if } a, b \in \Gamma_\alpha \text{ for some } \alpha \\ b + 2a & \text{if } a \in \Gamma_\alpha, b \in \Gamma_\beta \text{ for some } \alpha \neq \beta \textbf{ and } (\alpha,\beta) \in D \\ b & \text{if } a \in \Gamma_\alpha, b \in \Gamma_\beta \text{ for some } \alpha \neq \beta \textbf{ and } (\alpha,\beta) \notin D. \end{cases}$$

Lemma 3.3.1. *The mapping $\widetilde{\sigma} : M \to \mathrm{GL}(\mathbb{Z}^{\Gamma_1})$ with $w \mapsto \widetilde{\sigma}_w$, where $\widetilde{\sigma}_w = \widetilde{\sigma}_{w_1} \cdots \widetilde{\sigma}_{w_n}$ for $w = w_1 \cdots w_n \in (\Gamma^1)^*$ is well-defined.*

Proof. To check that the mapping $\widetilde{\sigma}$ is well-defined, we need to verify that multiplication in G_α is preserved and that $\sigma_a\sigma_b = \sigma_b\sigma_a$ for all $a \in G_\alpha, b \in G_\beta$ and $(\alpha,\beta) \in I$. For the multiplication in G_α let $a_1, a_2 \in G_\alpha$ and $b \in G_\beta$.

$$\begin{aligned} (\widetilde{\sigma}_{a_1}\widetilde{\sigma}_{a_2})(b) = \widetilde{\sigma}_{a_1}(\widetilde{\sigma}_{a_2}(b)) &= \widetilde{\sigma}_{a_1} \left(\begin{cases} [a_2 \cdot b] - a_2 & \text{if } \alpha = \beta \\ b + 2a_2 & \text{if } \alpha \neq \beta \textbf{ and } (\alpha,\beta) \in D \\ b & \text{if } \alpha \neq \beta \textbf{ and } (\alpha,\beta) \notin D \end{cases} \right) \\ &= \begin{cases} [a_1 \cdot a_2 \cdot b] - [a_1 \cdot a_2] & \text{if } \alpha = \beta \\ b + 2a_1 + 2[a_1 \cdot a_2] - 2a_1 & \text{if } \alpha \neq \beta \textbf{ and } (\alpha,\beta) \in D \\ b & \text{if } \alpha \neq \beta \textbf{ and } (\alpha,\beta) \notin D \end{cases} \\ &= \begin{cases} [a_1a_2 \cdot b] - [a_1a_2] & \text{if } \alpha = \beta \\ b + 2[a_1a_2] & \text{if } \alpha \neq \beta \textbf{ and } (\alpha,\beta) \in D \\ b & \text{if } \alpha \neq \beta \textbf{ and } (\alpha,\beta) \notin D \end{cases} \\ &= \widetilde{\sigma}_{[a_1a_2]}(b) \end{aligned}$$

Hence, we do have $\widetilde{\sigma}_{a_1}\widetilde{\sigma}_{a_2} = \widetilde{\sigma}_{[a_1a_2]}$. For the independence relation, fix $(\alpha,\beta) \in I$

and let $a \in G_\alpha, b \in G_\beta$ and $c \in G_\gamma$.

$$(\widetilde{\sigma}_a\widetilde{\sigma}_b)(c) = \widetilde{\sigma}_a(\widetilde{\sigma}_b(c)) = \widetilde{\sigma}_a \left(\begin{cases} [bc] - b & \text{if } \beta = \gamma \\ c + 2b & \text{if } \beta \neq \gamma \textbf{ and } (\beta,\gamma) \in D \\ c & \text{if } \beta \neq \gamma \textbf{ and } (\beta,\gamma) \notin D \end{cases} \right)$$

$$= \begin{cases} [bc] - b & \text{if } \beta = \gamma \\ c + 2a + 2b & \text{if } \alpha,\beta \neq \gamma \textbf{ and } (\beta,\gamma) \in D \textbf{ and } (\alpha,\gamma) \in D \\ c + 2b & \text{if } \alpha,\beta \neq \gamma \textbf{ and } (\beta,\gamma) \in D \textbf{ and } (\alpha,\gamma) \notin D \\ c + 2a & \text{if } \alpha,\beta \neq \gamma \textbf{ and } (\beta,\gamma) \notin D \textbf{ and } (\alpha,\gamma) \in D \\ c & \text{if } \alpha,\beta \neq \gamma \textbf{ and } (\beta,\gamma) \notin D \textbf{ and } (\alpha,\gamma) \notin D \\ [ac] - a & \text{if } \alpha = \gamma \end{cases}$$

$$= (\widetilde{\sigma}_b\widetilde{\sigma}_a)(c)$$

□

By the last lemma, we do have a mapping of the monoid M into some (possibly infinite-dimensional) linear group. The graph product G is obtained from M by identifying the different group identities in M:

$$G = M \Big/ \triangleleft 1_\alpha = 1_\beta \text{ for } \alpha,\beta \in \mathcal{L} \triangleright$$

To obtain a well-defined mapping on G, let $E = \{1_\alpha \mid \alpha \in \mathcal{L}\}$. The mapping $\widetilde{\sigma}_a$ is also a mapping on the quotient space $\mathbb{Z}^{\Gamma_1}/\mathbb{Z}^E$. Hence, we also consider $\widetilde{\sigma}_a$ as the mapping $\widetilde{\sigma}_a : \mathbb{Z}^{\Gamma_1}/\mathbb{Z}^E \to \mathbb{Z}^{\Gamma_1}/\mathbb{Z}^E$.

Lemma 3.3.2. *The mapping $\widetilde{\sigma} : G \to \mathrm{GL}(\mathbb{Z}^{\Gamma_1}/\mathbb{Z}^E)$ with $w \mapsto \widetilde{\sigma}_w$, where $\widetilde{\sigma}_w = \widetilde{\sigma}_{w_1} \cdots \widetilde{\sigma}_{w_n}$ for $w = w_1 \cdots w_n \in \Gamma^*$ is well-defined.*

Proof. By definition, we do have $\mathrm{id} = \widetilde{\sigma}_{1_\alpha} = \widetilde{\sigma}_{1_\beta}$. □

Finally, we define $\sigma : G \to \mathrm{GL}(\mathbb{Z}^\Gamma)$ with $w \mapsto \sigma_w$, where $\sigma_w = \sigma_{w_1} \cdots \sigma_{w_n}$ for $w = w_1 \cdots w_n \in \Gamma^*$. Thereby, we define $\sigma_a : \mathbb{Z}^\Gamma \to \mathbb{Z}^\Gamma$ for $a \in \Gamma_\alpha$ via the linear extension of

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

Lemma 3.3.3. *The mapping σ is well-defined.*

Proof. We do have $\mathbb{Z}^{\Gamma_1}/\mathbb{Z}^E \simeq \mathbb{Z}^\Gamma$ by $\varphi : \mathbb{Z}^{\Gamma_1}/\mathbb{Z}^E \to \mathbb{Z}^\Gamma$ with

$$\varphi(a) = \begin{cases} 0 & \text{if } a = 1_\alpha \text{ for some } \alpha \in \mathcal{L} \\ a & \text{otherwise} \end{cases}.$$

The mapping φ is linear and for any $a \in \Gamma$ it holds $\sigma_a = \varphi \circ \widetilde{\sigma_a}$. □

The following two lemmas are a useful tool to show that σ defines an embedding into some (infinite-dimensional) linear group. They show how the mapping σ_w behaves in specific cases. For some word $u \in \Gamma^*$ we say that $u = v_0 a_1 \cdots a_n v_n$ is the *α-factorization* of u if we have $a_i \in \Gamma_\alpha$ for $1 \leq i \leq n$ and $v_i \in (\Gamma \setminus \Gamma_\alpha)^*$ for $0 \leq i \leq n$.

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

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

and let $u = v_0 a_1 \cdots a_n v_n$ be the α-factorization of u, then for $c \in \Gamma_\alpha$ we have $\lambda_c \geq 0$ and

$$\lambda_c > 0 \iff \text{For some } i \text{ with } 1 \leq i \leq n : c =_G \begin{cases} a_i \cdots a_n & \text{if } \alpha \neq \varepsilon \\ a_i \cdots a_n \cdot e & \text{if } \alpha = \varepsilon \end{cases}$$

Proof. Since $(e, v) \in I$ we have $\sigma_w(e) = \sigma_{uv}(e) = \sigma_u \sigma_v(e) = \sigma_u(e)$. We proof some slightly stronger result. Let $u = u_1 \cdots u_n$ and i_α be the minimal index with $u_i \in \Gamma_\alpha$. If no such index exists, we set $i_\alpha = \infty$. Then for $i_\beta < i_\alpha$ the inequality

$$\sum_{a \in \Gamma_\alpha} \lambda_\alpha < \sum_{b \in \Gamma_\beta} \lambda_b$$

holds. For $u = 1$ the claim holds. Now, let $|u| \geq 1$ with $u = a'u'$ and $a' \in \Gamma_\alpha$. Let

$$\sigma_{u'}(e) = \sum_{c \in \Gamma} \lambda_c \cdot c = \sum_{a \in \Gamma_\alpha} \lambda_a \cdot a + \sum_{\substack{b \in \Gamma_\beta, \\ (\alpha,\beta) \in D}} \lambda_b \cdot b + \sum_{\substack{c \in \Gamma_\gamma, \\ (\alpha,\gamma) \notin D}} \lambda_c \cdot c.$$

By applying $\sigma_{a'}$ we obtain

$$\begin{aligned}
\sigma_u(e) &= \sigma_{a'}(\sigma_{u'}(e)) \\
&= \sum_{c\in\Gamma} \mu_c \cdot c \\
&= \sum_{\substack{a\in\Gamma_\alpha,\\ a'a\neq_G 1}} \lambda_a \cdot [a'a] - \sum_{a\in\Gamma_\alpha} \lambda_a a' + \sum_{\substack{b\in\Gamma_\beta,\\ (\alpha,\beta)\in D}} \lambda_b \cdot (b + 2a') + \sum_{\substack{c\in\Gamma_\gamma,\\ (\alpha,\gamma)\notin D}} \lambda_c \cdot c \\
&= \sum_{\substack{a\in\Gamma_\alpha,\\ a'a\neq_G 1}} \lambda_a \cdot [a'a] + \left(\sum_{\substack{b\in\Gamma_\beta,\\ (\alpha,\beta)\in D}} 2\lambda_b - \sum_{a\in\Gamma_\alpha} \lambda_a \right) a' + \\
&\quad \sum_{\substack{b\in\Gamma_\beta,\\ (\alpha,\beta)\in D}} \lambda_b \cdot b + \sum_{\substack{c\in\Gamma_\gamma,\\ (\alpha,\gamma)\notin D}} \lambda_c \cdot c.
\end{aligned}$$

Since $u = a'u'$ is reduced, there must be some δ with $(\alpha, \delta) \in D$ and $i'_\delta < i'_\alpha$. Thereby i'_α, i'_δ is as above the minimal index in u'. Therefore, for this δ we have

$$\sum_{a\in\Gamma_\alpha} \lambda_\alpha < \sum_{d\in\Gamma_\delta} \lambda_d$$

and thus

$$\mu_{a'} = \sum_{\substack{b\in\Gamma_\beta,\\ (\alpha,\beta)\in D}} 2\lambda_b - \sum_{a\in\Gamma_\alpha} \lambda_a > \sum_{d\in\Gamma_\delta} \lambda_d > 0.$$

Moreover, for any δ with $(\alpha, \delta) \in D$ and $i'_\delta < i'_\alpha$ we have

$$\begin{aligned}
\sum_{a\in\Gamma_\alpha} \mu_a &= \sum_{\substack{a\in\Gamma_\alpha,\\ a'a\neq_G 1}} \lambda_a + \sum_{\substack{b\in\Gamma_\beta,\\ (\alpha,\beta)\in D}} 2\lambda_b - \sum_{a\in\Gamma_\alpha} \lambda_a \\
&\geq \sum_{\substack{b\in\Gamma_\beta,\\ (\alpha,\beta)\in D}} 2\lambda_b - \sum_{a\in\Gamma_\alpha} \lambda_a \\
&> \sum_{d\in\Gamma_\delta} \lambda_d = \sum_{d\in\Gamma_\delta} \mu_d
\end{aligned}$$

This completes the proof, since only the coefficients of $a \in \Gamma_\alpha$ did change under $\sigma_{a'}$. □

Lemma 3.3.5. *Let $1 \neq_G w \in G$. If $a \in G_\alpha$ is maximal in w, then all non-zero coefficients of $\sigma_w(\bar{a})$ are negative, whereby $\bar{a}$ is the inverse of a in G_α.*

Proof. Since a is maximal in w, we have $w =_G uav$ with uav reduced, a is the single maximal element of u and $(a, v) \in I$. Hence, we have $\sigma_{uav}(\bar{a}) = \sigma_{ua}(\bar{a}) = \sigma_u(-a) = -\sigma_u(a)$. Now, ua is reduced. Therefore, by Lemma 3.3.4 all non-zero coefficients of $\sigma_u(a)$ are positive. Taking all this together, we obtain that all non-zero coefficients of $\sigma_w(\bar{a})$ are negative. □

Theorem 3.3.6. *The mapping $\sigma : w \mapsto \sigma_w$ is injective.*

Proof. Let $1 \neq_G w \in G$, then there is some $\alpha \in \mathcal{L}$ and $a \in \Gamma_\alpha$ with a maximal in w. Let $\bar{a}$ be the inverse of a in G_α. By Lemma 3.3.5 all non-zero coefficients of $\sigma_w(\bar{a})$ are negative. On the other side, $\sigma_1(\bar{a}) = \mathrm{id}(\bar{a}) = \bar{a}$. Hence, we must have $\mathrm{id} \neq \sigma_w$. Therefore, the kernel of σ is trivial and thus, the mapping σ is injective. □

The next corollary recovers a result of Tim Hsu and Daniel Wise [HW99]. It follows immediately from the above construction.

Corollary 3.3.7. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of finite groups. Then G is a (finite-dimensional) linear group.*

Proof. The alphabet Γ is finite, since $\Gamma = \biguplus_{\alpha\in\mathcal{L}} G_\alpha \setminus \{1_\alpha\}$. Hence, $\sigma(G)$ is a subgroup of $\mathrm{GL}(n, \mathbb{Z})$ for $n := |\Gamma|$. □

Remark 3.3.8. When the graph product is actually a right angled Coxeter group, that is $G_\alpha \simeq \mathbb{Z}/2\mathbb{Z}$, then the embedding σ is the same as the well known embedding of right angled Coxeter groups into $\mathrm{GL}(n, \mathbb{Z})$ (see Section 3.2).

Chapter 4

Algorithmic group theory and circuits

Max Dehn formulated in 1911 the three fundamental problems in algorithmic group theory. The word problem, the conjugacy problem and the isomorphism problem. A solution to the word problem is required for both, the conjugacy problem and the isomorphism problem. This chapter gives a precise definition of the (uniform) word and conjugacy problem. Further, the uniform classes of the direct product, free product and graph product are defined. Moreover, an encoding of group products suitable to work with circuits is given. The geodesic and normal form problem is later defined in the beginning of Chapter 6.

4.1 Problems

In this section, first the problems for a fixed group and then the uniform problems are described.

Definition 4.1 (word problem)**.** Let G be a group generated by Σ as a monoid. The word problem is the following question: Given a word $w \in \Sigma^*$. Is $w = 1$ in G? We shorten this by

$$\mathrm{WP}(G) = \{w \in \Sigma^* \mid w =_G 1\}.$$

Definition 4.2 (conjugacy problem)**.** Let G be a group generated by Σ as a monoid. We say two words $v, w \in \Sigma^*$ are *conjugate*, if there is some $z \in \Sigma^*$ with $zw\bar{z} =_G v$. In this case we write $v \sim_G w$. The conjugacy problem is the following question: Given two words $v, w \in \Sigma^*$. Does $v \sim_G w$ hold? We shorten this by

$$\mathrm{CP}(G) = \{(v, w) \in \Sigma^* \times \Sigma^* \mid v \sim_G w\}.$$

Definition 4.3 (generalized word problem)**.** Let $H \leq G$ be two groups and let G be generated by Σ as a monoid. The generalized word problem is the following question: Given a word $w \in \Sigma^*$. Is $w \in H$? We shorten this by

$$\mathrm{GWP}(H, G) = \{w \in \Sigma^* \mid w \in H\}.$$

For the uniform problems, let $\mathcal{C}$ be a class of f.g. groups. For some group presentation $\mathcal{P}$ (of any kind) we denote by $\langle\mathcal{P}\rangle$ the group generated by the presentation $\mathcal{P}$.

Definition 4.4 (uniform word problem)**.** Let $\mathcal{C}$ be a class of f.g. groups. The uniform word problem of $\mathcal{C}$ is the following question: Given a group presentation $\mathcal{P} \in \mathcal{C}$ with generating set Σ and word $w \in (\Sigma^{\pm})^*$. Is $w = 1$ in $\langle\mathcal{P}\rangle$? We shorten this by

$$\mathrm{WP}(\mathcal{C}) = \{(\mathcal{P}, w) \in \mathcal{C} \times \left(\Sigma^{\pm}\right)^* \mid w = 1 \text{ in } \langle\mathcal{P}\rangle\}.$$

Definition 4.5 (uniform conjugacy problem)**.** Let $\mathcal{C}$ be a class of f.g. groups. The uniform conjugacy problem of $\mathcal{C}$ is the following question: Given a group presentation $\mathcal{P} \in \mathcal{C}$ with generating set Σ and two words $v, w \in (\Sigma^{\pm})^*$. Is $v \sim w$ in $\langle\mathcal{P}\rangle$? We shorten this by

$$\mathrm{WP}(\mathcal{C}) = \{(\mathcal{P}, v, w) \in \mathcal{C} \times \left(\Sigma^{\pm}\right)^* \times \left(\Sigma^{\pm}\right)^* \mid v \sim w \text{ in } \langle\mathcal{P}\rangle\}.$$

Next, we describe the classes of groups relating to the group products considered in this thesis. Let $\mathcal{C}$ be a class of group presentations. We say the class $\mathcal{C}$ is a *non-trivial class* if $\mathcal{C}$ contains a non-trivial group. We denote by $\mathbf{\Pi}\mathcal{C} = \bigcup_{n\in\mathbb{N}} \mathcal{C}^n$ the set of tuples over $\mathcal{C}$ and associate to a tuple $(\mathcal{P}_1, \ldots, \mathcal{P}_n) \in \mathbf{\Pi}\mathcal{C}$ the direct product $\prod_{i=1}^n \langle\mathcal{P}_i\rangle$ over the groups generated by the presentations $\mathcal{P}_i$.

Further, we denote by $\mathbf{F}\mathcal{C} = \bigcup_{n\in\mathbb{N}} \mathcal{C}^n$ the set of tuples over $\mathcal{C}$ and associate to a tuple $(\mathcal{P}_1, \ldots, \mathcal{P}_n) \in \mathbf{F}\mathcal{C}$ the free product $*_{i=1}^n \langle\mathcal{P}_i\rangle$ over the groups generated by the presentations $\mathcal{P}_i$.

Lastly, let $I_n \subseteq \mathbb{B}^{n\times n}$ be the subset of binary $n \times n$ matrices representing an independence relation. Then we denote by $\mathbf{GP}\mathcal{C}$ the set $\mathbf{GP}\mathcal{C} = \bigcup_{n\in\mathbb{N}}(\mathcal{C}^n \times I_n)$ and associate to a tuple $(\mathcal{P}_1, \ldots, \mathcal{P}_n, I) \in \mathbf{GP}\mathcal{C}$ the graph product $\mathrm{GP}\left(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}}\right)$, where $\mathcal{L} = \{1, \ldots, n\}$ and $G_i = \langle\mathcal{P}_i\rangle$. We remark that it is possible in AC^0 to verify, whether some matrix does correspond to an independence relation.

4.2 Presentation of products of groups

In most complexity classes the encoding of groups does not play an important role. Already in $\mathsf{LOGSPACE}$ the generators of the groups can be considered as part of the alphabet of the Turing machine. However, when talking about circuit complexity a proper encoding must be ensured. Likewise, for the uniform word problem, the required generators are not known beforehand and thus a proper encoding must be used.

Suppose we have a list $\mathcal{P}_1 = \langle\Sigma_1 \mid \mathcal{R}_1\rangle, \ldots, \mathcal{P}_n = \langle\Sigma_n \mid \mathcal{R}_n\rangle$ of group presentations. Let $\mathcal{P}$ be any product (for example a direct product, a free product, a graph

product). We set $\mathcal{P} = \langle \Sigma \mid \mathcal{R} \rangle$ with

$$\Sigma = \biguplus_{i=1}^{n} \mathrm{bin}(i)\Sigma_i.$$

The set of relations does depend on the kind of product. Let $\mathcal{R}_P$ be some set of additional relations, depending on the product. Let $\rho_i : \Sigma_i \to \mathrm{bin}(i)\Sigma_i$ be the homomorphism induced by $\rho_i(a) = \mathrm{bin}(i)a$ for $a \in \Sigma_i$. Then we set

$$\mathcal{R} = \mathcal{R}_{\mathcal{P}} \uplus \biguplus_{i=1}^{n} \rho_i(\mathcal{R}_i).$$

Now, for the set $\mathcal{R}_P$. If we have a free product, then $\mathcal{R}_P$ is the empty set. If the product is a direct product, then $\mathcal{R}_P$ is the set of all commutators, that is

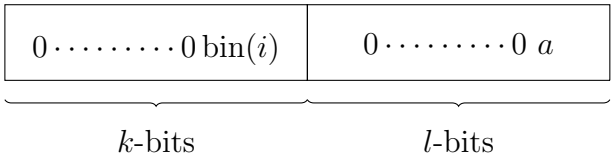
$$\mathcal{R}_P = \{[\mathrm{bin}(i)u, \mathrm{bin}(j)v] \mid 1 \leq i \neq j \leq m, u \in \Sigma_i, v \in \Sigma_j\}.$$

If the product is some graph product with dependence relation D, then we have

$$\mathcal{R}_P = \{[\mathrm{bin}(i)u, \mathrm{bin}(j)v] \mid 1 \leq i \neq j \leq m, u \in \Sigma_i, v \in \Sigma_j \textbf{ and } (i,j) \notin D\}.$$

4.3 Encoding of the input

Often, it is convenient to consider the input words to be given over the alphabet $\Gamma = G \setminus \{1\}$ of the group product G and to consider Γ to be *encoded* by Σ. Note, that the *encoding* of Γ is not unique. Now, for the circuit, each letter in Σ is encoded in a block of k-bits and a block of l-bits, where $k = \lceil \log n \rceil$ and l is the maximal bit-length required to encode any Σ_i in binary. If the encoding of $\mathrm{bin}(i)$ (respectively some letter $a \in \Sigma_i$) is shorter than k (respectively l), then it is padded with zeros to the left.



Extracting the group number $\mathrm{bin}(i)$ is a projection onto the first k-bits and extracting the letter a is a projection onto the last l-bits. Both projections can easily be done in AC^0. The identity of the group product can be encoded by filling both blocks completely with zeros. By convention, we may assume that the identity is encoded by a block of zeros in all groups. We can thus easily identify the identity in AC^0.

To simplify things, we can assume all blocks to have a size of n-bits, when the total input has length n and we may assume $n \geq k$. The circuit thus has $2n$ blocks of n-bit size. Superfluous bits are padded with zeros to the left.

For the uniform word (respectively conjugacy, etc.) problem, the encoding of the groups is (besides the input word) part of the input. For those we spend an extra n blocks of n-bit size for n group presentations. Since n has to be large enough to encode the input, we might have to many blocks for the input word or to many blocks for the group presentations. Therefore, the input word is padded to the right with the identity element that is the blocks are filled completely with zeros. The group presentations are padded to the right with a proper encoding for the trivial group. The trivial group can be encoded by just a block of zeros and if necessary transformed in AC^0 to another fixed representation. Finally, the encoding for multiple input words as in the conjugacy problem is similar.

$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$	$\cdots$	$\mathcal{P}_n$	w_1	w_2	w_3	$\cdots$	w_n

A nicer encoding in TC^0.

As soon as we have the power of TC^0, the encoding can be simplified in such a way that the padding with zeros can be omitted. We switch to an encoding over $\{0, 1, \$\}$. Each symbol can be encoded by a block of 2 bits, for example via

$$0 \mapsto 00$$
$$1 \mapsto 01$$
$$\$ \mapsto 10.$$

First, we describe the encoding for non-uniform group problems. Instead of using fixed-size blocks for the input, we separate each block in the input by the \$ symbol as follows

$$\mathrm{bin}(p_1)\$a_1\$ \cdots \$\,\mathrm{bin}(p_n)\$a_n =: b_1 \cdots b_{n'} \qquad \text{with } b_i \in \{0, 1, \$\}.$$

Hereby p_i denotes the letter a_i belongs to the group p_i. Recall that the letters a_i are encoded in binary. This encoding can be transformed in TC^0 into the fixed-size blocks described above. A symbol b_i belongs to block k if $b_j \neq \$$ and the number of \$ beforehand equals $k - 1$. Moreover it is the $(n' - j)$-th bit in this block if the next \$ symbol follows $(j + 1)$ bits later. Thus, bit $(n' - j)$ of block k is b_i, if

$$BIT(k, j, i) := \left[(b_{i+(j+1)} = \$) \wedge \bigwedge_{x=i+1}^{i+j} (b_x \neq \$) \right] \wedge (\#x < i : (b_x = \$)) = (k - 1)$$

This gives us bit $(n' - j)$ of block k through

$$BIT(k, j) := \bigvee_{x=1}^{n'} BIT(k, j, x) \wedge b_x.$$

For the uniform case, we introduce the additional symbol $\|$ and map it to the binary representation 11. An input word now has the form

$$\mathcal{P}_1\$ \cdots \$\mathcal{P}_m \parallel \mathrm{bin}(p_1)\$a_1\$ \cdots \$\,\mathrm{bin}(p_n)\$a_n =: b_1 \cdots b_{n'} \quad \text{with } b_i \in \{0, 1, \$, \|\}.$$

Hereby $\mathcal{P}_i$ is a presentation of some group. The formula for the transformation is the same as above, except that we have to distinguish on which side of the $\|$ symbol b_i occurs. This is achieved by

$$LEFT(i) := \bigvee_{j=i+1}^{n'} (b_j = \|)$$
$$RIGHT(i) := \bigvee_{j=1}^{i-1} (b_j = \|)$$

It is easy to extend this additional splitting of the input through $\|$ from two parts to an arbitrary number of parts. The encoding for multiple input words as in the conjugacy problem is similar. To conclude this section: As soon as we have the power of TC^0, the input can be given in a much more natural way. However, for the remaining part of this thesis, we always assume the block representation of the input to be given. It is more easy to process in circuits.

Chapter 5

The word problem

This chapter is devoted to the word problem in certain group products. We consider direct products, free products and graph products. Furthermore, the word problem for some restricted amalgamated products is solved. On the one side, it serves as a tool for the word problem of graph products. On the other side, it solves the word problem for a much larger class of groups. The results for direct products are easily derived. More interesting are the results for free products and graph products. They do generalize the results of [DKL12a, DKL12b] from graph groups to arbitrary graph products. Parts of these results are from [DK14, DK16], where the word problem of graph products is considered in the complexity class LOGSPACE. In this thesis, the complexity is further lowered from LOGSPACE with oracle to uAC^0 with oracle gates. Another interesting - and so far open - problem is the complexity of the uniform word problem in these group products. It is settled for direct products and free products. For graph products it is shown to be NL-hard and shown to be contained in $\mathsf{C}_=\mathsf{L}$ with an oracle for the word problem of the base groups.

For the correctness proofs of the solutions of the word problem we will make extensive use of the following lemma on semidirect products. A proof can be found in various textbooks.

Lemma 5.0.1 (Splitting Lemma)**.** *Let* $1 \to K \xrightarrow{\iota} G \xrightarrow{\pi} N \to 1$ *be a short exact sequence. If the sequence splits, that is, if one of the following conditions is met*

- *there exists* $\alpha : N \to G$ *with* $\pi \circ \alpha = \mathrm{id}_N$*, or*
- *there exists* $\beta : G \to K$ *with* $\beta \circ \iota = \mathrm{id}_K$*,*

then G *is isomorphic to the semidirect product* $N \rtimes K$*.*

5.1 Overview

Before we start with the word problem in group products,we give a short overview and some examples. It is important to note that the complexity of the word

problem of finitely generated groups is independent from the generating set. To see this, let $G \simeq H$ with

$$G = \langle a_1, \ldots, a_n \mid r_1, \ldots, r_k \rangle$$
$$H = \langle b_1, \ldots, b_m \mid s_1, \ldots, s_l \rangle$$

Since $G \simeq H$ there is an isomorphism $\varphi : G \to H$, which can be computed by an AC^0-transducer.

A major result in algorithmic group theory is the following result on linear groups over fields of characteristic zero by Lipton and Zalcstein in 1977. Simon extended this result in 1979 to linear groups over fields of prime characteristic. Each field has either characteristic zero or prime characteristic. Hence, combining both yields

Theorem 5.1.1 (Lipton, Zalcstein [LZ77] and Simon [Sim79])**.** *Let G be a finitely generated linear group. Then the word problem of G is in* LOGSPACE*.*

For linear groups over the integers this result has been improved to $\mathsf{C}_=\mathsf{NC}^1$, since multiplying together a sequence of integer matrices of constant dimension is complete for GapNC^1. Therefore, verifying the result is in $\mathsf{C}_=\mathsf{NC}^1$.

Examples

The following examples will serve as a base for the examples on the word problem of the group products.

Example 5.1.2. *The word problem of a fixed finite group G is contained in NC^1. Furthermore, if the finite group G is non-solvable, then the word problem is complete for NC^1 [Bar89].*

Example 5.1.3. *Let $\mathcal{C} = \{\langle a \mid a^n \rangle \mid n \in \mathbb{N} \cup \{0\}\}$ be the set of all cyclic groups. Then the word problem $\mathrm{WP}(\mathcal{C})$ is in uniform TC^0. More precisely, given a cyclic group and a word over $\{a\}^{\pm}$ as input, then the decision if $w = 1$ in this group is in uniform TC^0. In short we have*

$$\mathrm{WP}(\mathcal{C}) \in \mathsf{uTC}^0.$$

Proof. The input group G is given as $\langle a \mid a^n \rangle$. Hence, the order n of a is given in unary. If n is zero, then $G = \mathbb{Z} = \langle a \rangle$, whose word problem is uTC^0-complete. Otherwise, we use the uTC^0-complete problem BCount. Therefore, we can count the number of a and the number of $\bar{a}$ in TC^0. Now, let $w \in \left(\{a\}^{\pm}\right)^*$ be the input word. Then $w =_G 1$ if and only if $|w|_a - |w|_{\bar{a}} \mod n = 0$. By Hesse, integer division is in uniform TC^0 [Hes01]. Thus, the uniform word problem of

$\mathcal{C} = \{\langle a \mid a^n \rangle \mid n \in \mathbb{N}\}$ can be solved in uTC^0. For the hardness, the word problem of $\mathbb{Z} = \langle a \rangle$ is complete for uTC^0. Hence, uTC^0-completeness for the uniform word problem of cyclic groups follows. □

Example 5.1.4. *The uniform word problem of finite groups is the following problem. Given a finite group G via its multiplication table and a word w over elements of G. Is the word w equal to the identity in G? Following the work of Cook and McKenzie [CM87], this problem is complete for* LOGSPACE.

5.2 in free groups

Many reductions in this thesis reduce the word problem to at least the word problem of the free group. Moreover, many groups contain a free group. It is therefore of importance to understand the complexity of the word problem of the free group. Any free group F_n embeds into the free group F_2 over two generators. To see this, let $F_n = \langle x_1, \ldots, x_n \rangle$ and $F_2 = \langle s, t \rangle$. The mapping $x_i \mapsto t^i s t^{-i}$ defines an injective homomorphism, since the rank of the subgroup $\langle t^i s t^{-i} \mid 1 \leq i \leq n \rangle$ is n. In addition, the free group on two generators embeds into $\mathrm{SL}(2, \mathbb{Z})$. Thus, every free group is a linear group. The next theorem states a linear embedding of the free group. The very first linear representation of the free group on two generators is attributed to Sanov [San47].

Theorem 5.2.1. *The free group on two generators is a linear group. More precisely, it embeds into* $\mathrm{SL}(2, \mathbb{Z})$ *via the embedding* $\varphi : F_2 = \langle s, t \rangle \to \mathrm{SL}(2, \mathbb{Z})$ *which is induced by*

$$s \mapsto \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \quad t \mapsto \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}.$$

Since the free group is linear, its word problem is solvable in LOGSPACE by Theorem 5.1.1. However, the precise complexity of its word problem is unknown. A lower bound is known due to Robinson:

Theorem 5.2.2 (Robinson [Rob93])**.** *The word problem of the free group F_2 is NC^1-hard.*

Hence, the best one can hope for is NC^1. The following theorem gives an upper bound on the complexity in terms of arithmetic circuits. By a result of Jung, the arithmetic bound transfers to an upper bound in terms of Boolean circuits.

Theorem 5.2.3. *The word problem of the free group F_2 is in $\mathsf{C}_=\mathsf{NC}^1$.*

Proof. By Theorem 5.2.1 the free group F_2 is generated by 2×2 matrices over $\mathbb{Z}$. Iterated matrix multiplication of matrices of constant dimension is in GapNC^1, since matrix multiplication of matrices with constant dimension is in GapNC^0. The final step is to compare the resulting matrix with the identity matrix. Hence, the word problem of the free group F_2 can be solved by an arithmetic circuit of depth $\mathcal{O}(\log n)$. □

The following result on the complexity of GapNC^1 is by Jung. A simple proof may be found in [All04].

Theorem 5.2.4 (Jung [Jun85])**.** *Let $f \in \mathsf{GapNC}^1$. Then the function f is computed by a family of Boolean circuits having bounded fan-in, polynomial size and depth $\mathcal{O}(\log n \log^* n)$.*

Hence the word problem of the free group can be solved by a Boolean circuit of depth almost $\mathcal{O}(\log n)$. However, it is an open question, whether the equality $\mathsf{NC}^1 \stackrel{?}{=} \mathsf{C}_=\mathsf{NC}^1$ (respectively $\mathsf{NC}^1 \stackrel{?}{=} \mathsf{GapNC}^1$) holds.

Uniform solution

We close this section with the uniform word problem of the free group. That is, we consider the free group as part of the input.

Theorem 5.2.5. *The uniform word problem of the free group F_n with n generators is in $\mathsf{C}_=\mathsf{NC}^1$. More precisely, there is an uAC^0 many-one reduction of $\mathrm{WP}(F_n)$ onto $\mathrm{WP}(F_2)$.*

Proof. Let F_n be given in the input as $F_n = \langle x_1, \ldots, x_n \rangle$. Hence, the number n of generators is given in unary. As noted above F_n embeds into $F_2 = \langle s, t \rangle$ via $x_i \mapsto s^i t s^{-i}$. This mapping can be computed effectively in AC^0. For this, we do need that n is given in unary. Thus, we do have a uAC^0 many-one reduction of $\mathrm{WP}(F_n)$ onto $\mathrm{WP}(F_2)$. Moreover, the word problem of F_2 is contained in $\mathsf{C}_=\mathsf{NC}^1$. Now, the result follows since $\mathsf{C}_=\mathsf{NC}^1$ is closed under AC^0 many-one reductions. □

5.3 in direct products

Among the considered group products in this thesis, the direct is the most simple one. We list it for completeness here, and to show differences in the complexity to other group products. For the word problem, the decision if some word equals the identity transfers directly to the decision in the base groups. Moreover, this transformation can be performed in uniform AC^0. This does hold for the non-uniform and the uniform variant of the word problem in direct products.

Theorem 5.3.1. *The word problem in direct products is* AC^0 *Turing reducible to the word problem of the base groups.*

1. *Let* $G = \prod_{\alpha\in\mathcal{L}} G_\alpha$ *be a direct product of f. g. groups, then*

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

   that is there is a uniform AC^0 *Turing reduction of the word problem* $\mathrm{WP}(G)$ *onto the word problems* $\mathrm{WP}(G_\alpha)$*.*

2. *Let* $\mathcal{C}$ *be a non-trivial class of f. g. groups. Then*

$$\mathrm{WP}(\mathbf{\Pi}\mathcal{C}) \in \mathsf{uAC}^0(\mathrm{WP}(\mathcal{C})),$$

   that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$ *and a word* $w \in \Gamma^*$ *the word problem* $w = 1$ *in* $G = \prod_{i=1}^n G_i$ *can be solved in uniform* AC^0 *with oracle gates for the word problem of* $\mathcal{C}$*.*

Proof. Let the factorization of the input word be $w = w_1 \cdots w_n \in \Gamma^*$. We fix an enumeration of the $\alpha \in \mathcal{L}$. Hence, we may assume $\mathcal{L} = \{1, \ldots, |\mathcal{L}|\}$ and in *case 2.* we have $|\mathcal{L}| = n$. Consider the projection $\pi_i : \Gamma^* \to \Gamma_i^*$ defined through

$$\pi_i(w_k) = \begin{cases} w_k, & \text{if } \mathrm{alph}(w_k) = \mathrm{bin}(i) \\ 1, & \text{otherwise.} \end{cases}$$

This projection can obviously be computed by an AC^0 circuit. A necessary condition for w to equal the identity is $\pi_i(w) =_{G_i} 1$ for all $i \in \mathcal{L}$. Since G is the direct product of the G_α it holds

$$w =_G \prod_{\alpha\in\mathcal{L}} \pi_\alpha(w).$$

Hence, $w =_G 1$ if and only if $\pi_i(w) =_{G_i} 1$ for all $i \in \mathcal{L}$. Finally, in *case 1.* the word problem of G reduces to the circuit

$$w \in \mathrm{WP}(G) \equiv \bigwedge_{i\in\mathcal{L}} \Big(\pi_i(w) \in \mathrm{WP}(G_\alpha)\Big).$$

by using the oracle gates for $\mathrm{WP}(G_i)$. The uniform solution in *case 2.* reduces to the circuit

$$(w, G) \in \mathrm{WP}(\Pi\mathcal{C}) \equiv \bigwedge_{i=1}^{n} \Big((\pi_i(w), G_i) \in \mathrm{WP}(\mathcal{C})\Big).$$

by using the oracle gates for $\mathrm{WP}(\mathcal{C})$. □

Example 5.3.2. *Let $\mathcal{C} = \{\langle a \mid a^n \rangle \mid n \in \mathbb{N} \cup \{0\}\}$ be the set of all cyclic groups. Then the uniform word problem* $\mathrm{WP}(\mathbf{\Pi}\mathcal{C})$ *is* uTC^0*-complete.*

Proof. By Theorem 5.3.1 case 2, the uniform word problem of $\Pi\mathcal{C}$ is contained in $\mathsf{uAC}^0(\mathrm{WP}(\mathcal{C}))$. Moreover, by Example 5.1.3 we have that the word problem of $\mathcal{C}$ is uTC^0-complete. Hardness follows immediately for direct products, since for any group G we have $G = \prod_{i=1}^{1} G$. The result follows, since $\mathsf{uAC}^0 \subseteq \mathsf{uTC}^0$. □

5.4 in free products

We start with a result of S. Waack dating back to 1990. It states that the word problem of the free product of two groups G_1, G_2 is NC^1-reducible to the word problem of G_1, G_2 and the free group F_2.

Theorem 5.4.1 (Waack [Waa90])**.** *Let G_α, G_β be f. g. groups and $G = G_\alpha * G_\beta$ with $|G_\alpha| \cdot |G_\beta| \geq 6$. Then there is an NC^1 Turing reduction of* $\mathrm{WP}(G)$ *to* $\{\mathrm{WP}(G_\alpha), \mathrm{WP}(G_\beta), \mathrm{WP}(F_2)\}$.

Corollary 5.4.2. *Let G_α, G_β be f. g. groups and $G = G_\alpha * G_\beta$ with $|G_\alpha| \cdot |G_\beta| \geq 6$. If the word problem of G_α, G_β is in $\mathsf{C}_=\mathsf{NC}^1$, then the word problem of G is in $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$.*

The theorem of Waack shows that the complexity of the word problem of a fixed free product does not increase in most cases. The proof of Waack uses that the commutator subgroup of G is a free group, when $|G_\alpha| \cdot |G_\beta| \geq 6$. However, as remarked by Waack if $|G|_\alpha \cdot |G|_\beta < 6$, then $G_\alpha \simeq G_\beta \simeq \mathbb{Z}/2\mathbb{Z}$. But the free product $\mathbb{Z}/2\mathbb{Z} * \mathbb{Z}/2\mathbb{Z}$ has a uTC^0-complete word problem. Therefore, it is not a real restriction since the word problem of the free group F_2 is NC^1-hard .

Later in this section, we will see that the restriction of $|G_\alpha| \cdot |G_\beta| \geq 6$ is not necessary for our proof. Furthermore, we show that the reduction can already be achieved in AC^0.

The following lemma will be useful in some reductions. Its consequence is that any TC^0 reduction on the word problem of some free product can be transformed into an AC^0 reduction.

Lemma 5.4.3. *Let G_α, G_β be non-trivial f. g. groups. Then the word problem of the free product $G = G_\alpha * G_\beta$ is uTC^0-hard.*

Proof. The word problem of $\mathbb{Z} = \langle a \rangle$ is uTC^0-hard, since the problem MAJORITY reduces uniformly to the word problem in $\mathbb{Z}$. If either G_α or G_β contain some torsion-free group element, then their word problem is already hard for uniform TC^0. Otherwise, suppose $x \in G_\alpha$ and $y \in G_\beta$ have torsion. The embedding $a \mapsto xy$ maps $\mathbb{Z}$ into G. Note that even if x or y are torsion-free, the mapping defines an embedding of $\mathbb{Z}$ into G. Thus, the word problem of G is hard for uniform TC^0. □

Solving the word problem in AC^0 with oracle gates

The main tool for the word problem in free products is the following lemma. It is stated for an n-fold free product over the same base group. Later we will see that this restricted free product is already sufficient for arbitrary free products.

Lemma 5.4.4. *Let B be a f. g. group, $\mathcal{R}$ be some (possibly infinite) set and let $K = *_{\nu\in\mathcal{R}} B^{(\nu)}$ be an $|\mathcal{R}|$-fold free product with $B^{(\nu)} \simeq B$ and $1 \in \mathcal{R}$ a distinguished element. Then it holds $K \simeq B \rtimes F(X)$, where $F(X)$ is a free group with basis*

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

Proof. Consider the homomorphism $\varphi : K \to B$ with $\varphi(b_1^{(\nu_1)} \cdots b_n^{(\nu_n)}) = b_1 \cdots b_n$. We obtain the splitting short exact sequence $(1) \to \ker\varphi \to K \xrightarrow{\varphi} B \to (1)$ by choosing any embedding of B into $*_{\nu\in\mathcal{R}} B^{(\nu)}$. Therefore we have $K \simeq B \rtimes \ker\varphi$. We show that $\ker\varphi \simeq F(X)$ for some basis X. Let $w = b_1^{(\nu_1)} \cdots b_n^{(\nu_n)}$ be any word in $\ker\varphi$ and let $g_i = b_1 \cdots b_i \in B$ and $g_i^{(\nu)} \in B^{(\nu)}$. Observe that $g_n = 1$ in B since $w \in \ker\varphi$. A short calculation shows

$$\begin{aligned} w &= g_1^{(\nu_1)} \cdot \left(\overline{g_1}^{(\nu_2)} \cdot g_2^{(\nu_2)}\right) \cdots \left(\overline{g_{n-1}}^{(\nu_n)} \cdot g_n^{(\nu_n)}\right) \\ &= \left(g_1^{(\nu_1)} \cdot \overline{g_1}^{(\nu_2)}\right) \cdots \left(g_{n-1}^{(\nu_{n-1})} \cdot \overline{g_{n-1}}^{(\nu_n)}\right) \\ &= \prod_{i=1}^{n-1} g_i^{(\nu_i)} \cdot \overline{g_i}^{(\nu_{i+1})} \end{aligned}$$

One might pick those factors as a basis. But, the set $\{g^{(\nu)}\overline{g}^{(\mu)} \mid g \in B, \mu, \nu \in \mathcal{R}\}$ is not a free basis yet since $\left(g^{(\nu)}\overline{g}^{(\mu)}\right) \cdot \left(g^{(\mu)}\overline{g}^{(\lambda)}\right) = g^{(\nu)}\overline{g}^{(\lambda)}$ for $\lambda, \mu, \nu \in \mathcal{R}$. In the following we identify (ν, g, μ) with $g^{(\nu)}\overline{g}^{(\mu)}$. We define

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

The inverse of $(\nu, g, 1) \in X$ is $(1, g, \nu)$. Let $\psi : F(X) \to K$ be the homomorphism defined by $\psi((\nu, g, \mu)) = g^{(\nu)}\overline{g}^{(\mu)}$. As we will see, ψ is indeed an isomorphism.

To see that ψ is surjective, let $w = \prod_{i=1}^{n-1} g_i^{(\nu_i)} \cdot \overline{g_i}^{(\nu_{i+1})}$ be as above. We identify $(1, g, 1)$ and $(\nu, 1, \mu)$ with $1 \in F(X)$ for any $g \in B$, $\nu, \mu \in \mathcal{R}$ to simplify notation. It follows directly that $\psi(\prod_{i=1}^{n-1}(\nu_i, g_i, 1)\overline{(\nu_{i+1}, g_i, 1)}) = w$ and hence ψ is surjective.

Now, for ψ being injective, let $w \in (X \uplus \overline{X})^*$ be a reduced sequence. We show the following property of ψ with induction:

> *If $w = v \cdot (\nu, g, \mu)$, then the last factor in a reduced sequence of its image $\psi(w) = \psi(v \cdot (\nu, g, \mu))$ is $\overline{g}^{(\mu)}$. Moreover, if $\mu = 1$, then the last two factors of its image are $h^{(\nu)}\overline{g}^{(1)}$ for some $h \in B$.*

This claim shows that the kernel of ψ is trivial, and hence, ψ is injective. For its proof, assume $w = (\nu, g, \mu)$, then $\psi((\nu, g, \mu)) = g^{(\nu)}\overline{g}^{(\mu)}$ as desired. Now, let $|w| > 1$, then we have $w = v' \cdot (\alpha, f, \beta) \cdot (\nu, g, \mu)$ with $v = v'(\alpha, f, \beta)$. By induction, the last factor of $\psi(v)$ is $\overline{f}^{(\beta)}$. If $\beta \neq \nu$, no further reduction occurs in $\overline{f}^{(\beta)} g^{(\nu)} \overline{g}^{(\mu)}$ and we are done. Hence, we may assume $\beta = \nu$. If $f \neq g$, then the last two factors are $(\overline{f}g)^{(\nu)}\overline{g}^{(\mu)}$. Moreover, it satisfies the second condition if $\mu = 1$. Finally, assume $f = g$, then we must have $\alpha \neq \mu$, because otherwise w would not have been reduced. But then we must have $\beta = \nu = 1$ and by induction the last two factors of $\psi(v)$ are $h^{(\alpha)}\overline{f}^{(1)}$. Hence, the last two factors in $\psi(w)$ are $h^{(\alpha)}\overline{g}^{(\mu)}$. □

We are now able to construct a circuit to solve the word problem in a free product over the same base group. The correctness of the circuit follows directly by the previous lemma.

Lemma 5.4.5. *Let B be some f. g. group and $\mathcal{R}$ some finite list. Moreover, let $G = *_{\nu \in \mathcal{R}} B^{(\nu)}$. Then*

$$\mathrm{WP}(G) \in \mathsf{uAC}^0(\{\mathrm{WP}(B), \mathrm{WP}(F_2)\}),$$

that is the word problem of G is AC^0 Turing reducible to the word problem of B and F_2. Moreover, the reduction is a $\mathsf{uAC}(\mathrm{WP}(B))$ many-one reduction of $\mathrm{WP}(G)$ onto the word problem of the free group F_2.

Proof. Let $w = w_1 \cdots w_n \in \Gamma*$ be the factorization of the input word. Every letter is given as $w_i = \nu_i b_i$, which means w_i equals $b_i \in B^{(\nu_i)}$. By Lemma 5.4.4 we have $G \simeq B \rtimes F(X)$ with

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

Let $g_i = b_1 \cdots b_i$. These words can be constructed in AC^0. Moreover, if we have $b_1 \cdots b_n =_B 1$, then

$$w = \prod_{i=1}^{n-1} (\nu_i, g_i, 1)\overline{(\nu_{i+1}, g_i, 1)}.$$

Let $Y = \{(\nu_i, g_i, 1), (\nu_{i+1}, g_i, 1) \mid 1 \leq i < n-1\} \subseteq X$ be a finite subset of X. Then we have $w \in F(Y) \leq F(X)$. Note that $(\nu, g, 1) = (\mu, h, 1)$ if and only if $\nu = \mu$ and $g =_B h$. Hence, for the cardinality of Y we have $|Y| \leq 2(n-1)$. The following circuit Eq is to decide, if $(\nu, g_i, 1) = (\mu, g_j, 1)$.

$$Eq(\nu, i, \mu, j) \equiv \Big(\nu = \mu \wedge \Big(g_i \overline{g_j} \in \mathrm{WP}(B)\Big)\Big)$$

In a final step we simplify the basis and make the tuples $(\nu, g, 1)$ unique. For this, let $\widehat{Y} = \{x_1, \ldots, x_{2(n-1)}\}$. Consider the map $\psi : Y \to \widehat{Y}$ defined by

$$(\nu, g_i, 1) \mapsto \begin{cases} x_k, & \text{if } \nu = \nu_i \text{ and} \\ & k = \min\{j \leq i \mid (\nu_i, g_i, 1) = (\nu_j, g_j, 1)\}, \\ x_k, & \text{if } \nu = \nu_{i+1} \text{ and} \\ & k = \min\{1 \leq j \leq n \mid (\nu_{i+1}, g_i, 1) = (\nu_j, g_j, 1)\} \\ & \text{exists}, \\ x_{k+(n-1)}, & \text{if } \nu = \nu_{i+1} \text{ and} \\ & k = \min\{j \leq i \mid (\nu_{i+1}, g_i, 1) = (\nu_{j+1}, g_j, 1)\} \\ & \text{otherwise.} \end{cases}$$

The map ψ extends naturally to an isomorphism $\psi : F(Y) \to F(\widehat{Y})$. The minimum of a set is AC^0-computable and hence, ψ is AC^0-computable. Let $\widehat{w}$ be the resulting word over $\left(\widehat{Y}^{\pm}\right)^*$. By Theorem 5.2.5 there is an uAC^0 many-one reduction of $\left(F(\widehat{Y}), \widehat{w}\right)$ onto the word problem of F_2. Summing up, we have an $\mathsf{uAC}^0(\mathrm{WP}(B))$ many-one reduction onto the word problem of F_2. □

Remark 5.4.6. In the previous lemma, the only requirement for $\mathcal{R}$ is to be finite. In the AC^0 reduction a test for $\nu =_{\mathcal{R}} \mu$ is required. If the encoding of any $\nu \in \mathcal{R}$ is unique, this check can be realized by a binary comparison in AC^0. But then, $\mathcal{R}$ does not have to be fixed for the reduction. Moreover, for any word w the set $\mathcal{R}$ is finite, since only a finite subset does occur in w. We will exploit this fact in the reductions for the word problem of (some restricted) amalgamated products, see Section 5.5.

Theorem 5.4.7. *Let $G = *_{\alpha \in \mathcal{L}} G_\alpha$ be a free product of f. g. groups. Then*

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

that is there is a uniform AC^0 Turing reduction of the word problem $\mathrm{WP}(G)$ onto the word problems $\mathrm{WP}(G_\alpha)$ and the word problem of the free group F_2. Moreover, $\mathrm{WP}(G)$ is $\mathsf{uAC}(\{\mathrm{WP}(G_\alpha) \mid \alpha \in \mathcal{L}\})$ many-one reducible onto the word problem of the free group F_2.

Proof. The input is transformed such that Lemma 5.4.5 can be applied. We set $B = \prod_{\alpha \in \mathcal{L}} G_\alpha$ and $G' = *_{\alpha \in \mathcal{L}} B^{(\alpha)}$. Then G embeds into G' via $\varphi : G \to G'$ with

$$a \mapsto a^{(\alpha)} \quad \text{if } a \in G_\alpha \text{ for some } \alpha \in \mathcal{L}$$

By case 1 of Theorem 5.3.1 we have $\mathrm{WP}(B) \in \mathsf{AC}^0(\{\mathrm{WP}(G_\alpha) \mid \alpha \in \mathcal{L}\})$. Finally, by applying Lemma 5.4.5 we obtain a $\mathsf{uAC}^0(\{\mathrm{WP}(G_\alpha) \mid \alpha \in \mathcal{L}\})$ many-one reduction of $WP(G)$ onto the word problem $\mathrm{WP}(F_2)$ of the free group. □

Corollary 5.4.8. *Let $G = *_{\alpha\in\mathcal{L}} G_\alpha$ be a free product over f. g. groups G_α with $\mathrm{WP}(G_\alpha) \in \mathsf{uNC}^1$ for $\alpha \in \mathcal{L}$. Then*

$$\mathrm{WP}(G) \in \mathsf{uAC}^0(\mathrm{WP}(F_2)) \text{ and } \mathrm{WP}(G) \in \mathsf{C}_=\mathsf{NC}^1.$$

Proof. By Theorem 5.4.7 the word problem of G is $\mathsf{uAC}^0(\{\mathrm{WP}(G_\alpha) \mid \alpha \in \mathcal{L}\})$ many-one reducible to the word problem of the free group F_2. This reduction is at least a uNC^1 many-one reduction, since $\mathrm{WP}(G_\alpha) \in \mathsf{NC}^1$ for all $\alpha \in \mathcal{L}$. The word problem of the free group is NC^1 hard by Theorem 5.2.2. Hence, the reduction can be transformed into an uAC^0 Turing reduction.

For the second claim, the complexity class $\mathsf{C}_=\mathsf{NC}^1$ is closed under NC^1 many-one reductions. Hence, the result follows. □

Example 5.4.9. *The word problem of a free product over finite groups is contained in $\mathsf{C}_=\mathsf{NC}^1$.*

Proof. There are two ways to proof the statement. First, the word problem of a fixed finite group is in NC^1. Together with the previous corollary we have that the word problem of a free product over finite groups is in $\mathsf{C}_=\mathsf{NC}^1$. For the second way, a different approach is used. By Corollary 3.3.7 a free product over finite groups is a linear group. Hence, its word problem is solvable in $\mathsf{C}_=\mathsf{NC}^1$. □

The previous examples and corollaries considered the word problem of a fixed free product. For the uniform variant, basically the same proofs can be applied. Furthermore, the result for the reduction is the same.

Theorem 5.4.10. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then*

$$\mathrm{WP}(\mathbf{F}\mathcal{C}) \in \mathsf{uAC}^0(\{\mathrm{WP}(\mathcal{C}), \mathrm{WP}(F_2)\}),$$

that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$ and a word $w \in \Gamma^*$ the word problem $w = 1$ in $G = *_{i=1}^n G_i$ can be solved in uniform AC^0 with oracle gates for the word problem of $\mathcal{C}$ and oracle gates for the word problem of the free group F_2 on two generators. Moreover, the reduction is a $\mathsf{uAC}(\mathrm{WP}(\mathcal{C}))$ many-one reduction of $\mathrm{WP}(G)$ onto the word problem of the free group F_2.

Proof. The input is transformed such that Lemma 5.4.5 can be applied. We set $B = \prod_{\alpha\in\mathcal{L}} G_\alpha$ and $G' = *_{\alpha\in\mathcal{L}} B^{(\alpha)}$. Then G embeds into G' via $\varphi : G \to G'$ defined by

$$a \mapsto a^{(\alpha)} \quad \text{if } a \in G_\alpha \text{ for some } \alpha \in \mathcal{L}$$

By case 2 of Theorem 5.3.1 we have $\mathrm{WP}(B, \mathbf{\Pi}\mathcal{C}) \in \mathsf{AC}^0(\mathrm{WP}(\mathcal{C}))$. Finally, by applying Lemma 5.4.5 we obtain a $\mathsf{uAC}^0(\mathrm{WP}(\mathcal{C}))$ many-one reduction of $WP(G)$ onto the word problem $\mathrm{WP}(F_2)$ of the free group. □

Example 5.4.11. *Let $\mathcal{C} = \{\langle a \mid a^n \rangle \mid n \in \mathbb{N} \cup \{0\}\}$ be the set of all cyclic groups. Then the uniform word problem* $\mathrm{WP}(\mathbf{F}\mathcal{C})$ *is contained in* $\mathsf{C}_=\mathsf{NC}^1$.

Proof. By Theorem 5.4.10, the uniform word problem of $\mathbf{F}\mathcal{C}$ is $\mathsf{uAC}^0(\mathrm{WP}(\mathcal{C}))$ many-one reducible onto the word problem of F_2. Moreover, by Example 5.1.3 we have that the word problem of $\mathcal{C}$ is uTC^0-complete. Therefore, the reduction is a uTC^0 many-one reduction. The result follows, since $\mathrm{WP}(F_2) \in \mathsf{C}_=\mathsf{NC}^1$ and $\mathsf{C}_=\mathsf{NC}^1$ is closed under uTC^0 many-one reductions. □

Example 5.4.12. *The uniform word problem of free products over finite groups is* LOGSPACE*-complete, when the finite groups are given by their multiplication table.*

Proof. By Cook and McKenzie the uniform word problem of finite groups given by their multiplication table is LOGSPACE-complete [CM87]. Hardness follows immediately for free products, since for any group G we have $G = *_{i=1}^{1} G$. Completeness follows, since LOGSPACE is closed under AC^0 Turing reductions and the word problem of F_2 is contained in LOGSPACE by Theorem 5.1.1. □

5.5 in certain amalgamated products

This section serves as a preparation for the word problem in graph products. Any graph product can inductively be written as a free product of two smaller graph products amalgamated over some base group. For the graph product, we only need a solution for the word problem of groups of the form $G = P *_A (B \times A)$. However, under some additional assumptions, the achieved results hold for groups of the form $G = P *_A (B \rtimes A)$. In this section, we first show the results for groups of the form $G = P *_A (B \times A)$. These results are then transfered to groups of the form $G = P *_A (B \rtimes A)$ under some minor changes. For the direct product the generalized word problem of A in P must be solved. But for the semi-direct product one must additionally be able to rewrite the word in generators of A.

In this section, we will denote the set $\{b^{(\nu)} \mid b \in B\}$ by $B^{(\nu)}$ for any $\nu \in P/A$. Moreover, let $P = \langle \Sigma_P \mid \mathcal{R}_P \rangle$, $A = \langle \Sigma_A \mid \mathcal{R}_A \rangle$ and $B = \langle \Sigma_B \mid \mathcal{R}_B \rangle$. The union of all alphabets is denoted by $\Sigma = \Sigma_P \uplus \Sigma_A \uplus \Sigma_B$. For the theoretical part, the main idea is to consider a retract $\pi : G \to P$. Then the short exact sequence $(1) \to K \to G \xrightarrow{\pi} P \to (1)$ splits since $P \leq G$ and π is a retract. Thus, G is the semi-direct product $G \simeq K \rtimes P$, where $K = \ker \pi$. The goal is therefore to analyze the kernel and to rewrite the input word to solve the word problem of G via the word problem of K and P.

For the correctness proof of the following algorithms we do need the normal form theorem for amalgamated products. Let $G = F *_{A=B} H$ be an amalgamated

product. A factorization of a word $w = w_1 \cdots w_n \in \Gamma^*$ is said to be reduced if w_i, w_{i+1} come from different factors, $w_i \notin A, B$ if $n > 1$ and $w_1 \neq 1$ if $n = 1$.

Theorem 5.5.1 ([LS01, Theorem 2.6])**.** *If $w = w_1 \cdots w_n \in \Gamma^*$ is reduced, $n \geq 1$, then the product $w_1 \cdots w_n \neq 1$ in G. In particular, F and H are embedded in G by the maps $f \mapsto f$ and $h \mapsto h$.*

Word problem in $G = P *_A (B \times A)$

The algorithm for the word problem is based on the following lemma. It is the crucial part for the correctness of the algorithm.

Lemma 5.5.2. *Let $G = P *_A (B \times A)$ and let $\pi : G \to P$ be a retract. Then we have $G \simeq \ker \pi \rtimes P$ and, more precisely,*

$$\ker \pi \simeq \mathop{\ast}_{\nu \in P/A} B^{(\nu)}.$$

Proof. Since π is a retract, we have $G \simeq \ker \pi \rtimes P$. To see that $\ker \pi \simeq *_{\nu \in P/A} B^{(\nu)}$, we fix a set $\mathcal{R}$ of representatives of the cosets in P/A. Let $\varphi : *_{\nu \in \mathcal{R}} B^{(\nu)} \to \ker \pi$ be defined through $\varphi(b_1^{(\nu_1)} \cdots b_n^{(\nu_n)}) = \nu_1 b_1 \overline{\nu_1} \cdots \nu_n b_n \overline{\nu_n}$.

We first show that φ is injective. Let $w = b_1^{(\nu_1)} \cdots b_n^{(\nu_n)} \neq 1$ be a reduced sequence in $*_{\nu \in \mathcal{R}} B^{(\nu)}$, that is $\nu_i A \neq \nu_{i+1} A$ for $1 \leq i < n$. Thus, we have $\nu_i \overline{\nu_{i+1}} \notin A$ and by the normal form theorem for amalgamated products (Theorem 5.5.1) it follows $\varphi(w) \neq 1$. Hence, the kernel of φ is trivial and φ must be injective.

To see that φ is surjective, let $w = p_1 b_1 \cdots p_n b_n \in \ker \pi$ and choose the representative ν_i of $p_1 \cdots p_i$ in $\mathcal{R}$, that is $\nu_i A = p_1 \cdots p_i A$. We obtain $p_1 \cdots p_i b \overline{p_1 \cdots p_i} = \nu_i b \overline{\nu_i}$ for all $b \in B$ since there is some $a \in A$ with $\nu_i a = p_1 \cdots p_i$ and $ab = ba$. It is easy to see that $w = b_1^{\nu_1} \cdots b_n^{\nu_n}$ in G and therefore $\varphi(b_1^{(\nu_1)} \cdots b_n^{(\nu_n)}) = w$. □

One might remark that the free product in Lemma 5.5.2 is possibly an infinite free product since the index $[P : A]$ is, in most cases, infinite.

Theorem 5.5.3. *Let P, A and B be f. g. groups. Moreover, let $G = P *_A (B \times A)$. Then*

$$\mathrm{WP}(G) \in \mathsf{uAC}^0(\{\mathrm{WP}(P), \mathrm{WP}(B), \mathrm{GWP}(A, P), \mathrm{WP}(F_2)\}),$$

that is the word problem of G is AC^0 Turing reducible to the word problems $\mathrm{WP}(P)$, $\mathrm{WP}(B)$, $\mathrm{WP}(F_2)$ and the generalized word problem of A in P.

Proof. Let $w = w_1 \cdots w_n \in (\Sigma^{\pm})^*$ be the input word with $w_i \in \Sigma^{\pm}$. For $a \in \Sigma_A$ we denote by $a_P \in \left(\Sigma_P^{\pm}\right)^*$ a fixed word with $a =_P a_P$. The projections $\pi_P : (\Sigma^{\pm})^* \to$

$\left(\Sigma_P^{\pm}\right)^*$ and $\pi_B : (\Sigma^{\pm})^* \to \left(\Sigma_B^{\pm}\right)^*$ are induced by

$$\pi_P(a) = \begin{cases} a, & a \in \Sigma_P^{\pm} \\ a_P, & a \in \Sigma_A^{\pm} \\ 1, & a \in \Sigma_B^{\pm} \end{cases} \qquad \text{and} \qquad \pi_B(a) = \begin{cases} 1, & a \in \Sigma_P^{\pm} \uplus \Sigma_A^{\pm} \\ a, & a \in \Sigma_B^{\pm} \end{cases}$$

The projection $\pi_P : G \to P$ is a retract and hence, by Lemma 5.5.2 we have $G \simeq *_{\nu \in P/A} B^{(\nu)} \rtimes P$. Let $p_i = \pi_P(w_i)$ and $b_i = \pi_B(w_i)$. Therefore, we have $p_i \in \left(\Sigma_P^{\pm} \uplus \{1\}\right)^*$ and $b_i \in \left(\Sigma_B^{\pm} \uplus \{1\}\right)$. We may assume $\pi_B(w) =_B 1$ for the rest of the construction. Next, the input word has to be rewritten as a part of the kernel. For this the words $g_i = p_1 \cdots p_i$ can be constructed in AC^0. By the same reasoning as in the previous lemma we have $w = g_1 b_1 \overline{g_1} \cdots g_n b_n \overline{g_n}$. Let $\mathcal{R} = \{1, \ldots, n\}$. We set

$$\begin{aligned} \nu_i &= \min \{1 \leq j \leq n \mid g_i A = g_j A\} \\ &= \min \{1 \leq j \leq n \mid g_i \overline{g_j} \in A\} \end{aligned}$$

The input word w equals the identity, if and only if $\pi_B(w) =_B 1$ and $b_1^{(\nu_1)} \cdots b_n^{(\nu_n)}$ is the identity in $*_{\nu \in \mathcal{R}} B^{(\nu)}$. By using the oracle gates of $\mathrm{GWP}(A, P)$ for $g_i \overline{g_j} \in A$ the ν_i can be computed in AC^0. Finally, by Lemma 5.4.5 the result follows. □

The reduction process in the previous theorem is depicted in Figure 5.1 on page 60.

Word problem in $G = P *_A (B \rtimes_\varphi A)$

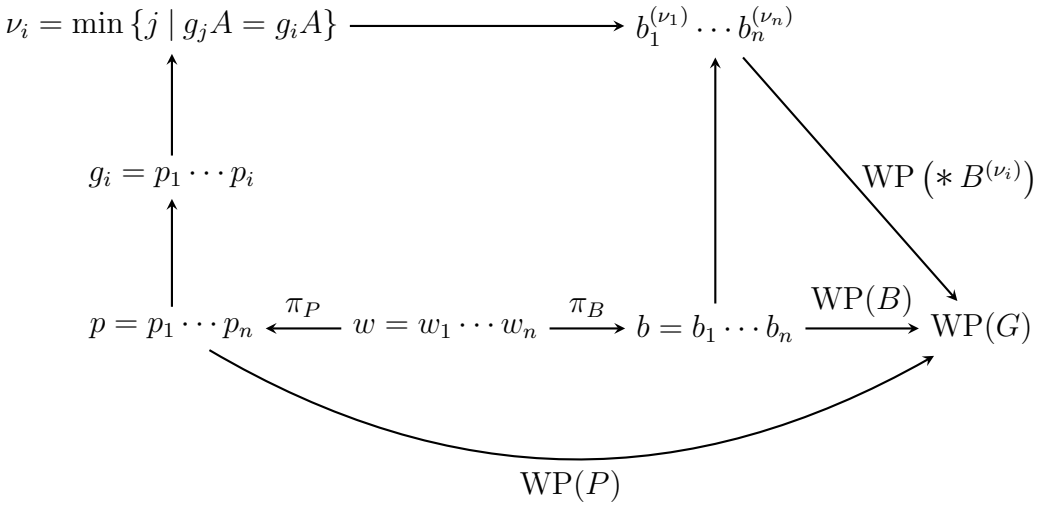
For the switch of the direct product $B \times A$ to the semi-direct product $B \rtimes_\varphi A$ we change the representation of B to

$$B^A = \{(1, a)(b, \overline{a}) \mid a \in A, b \in B\} \leq B \rtimes_\varphi A$$

Since B is normal in $B \rtimes_\varphi A$, we have $B \simeq B^A$. However, one might not be able to compute the isomorphism between B^A and B in AC^0. For this reason, we will only use the above representation of B^A. Note that B^A is not finitely generated if A is not finite. If $B \rtimes_\varphi A$ is a direct product, then we have $(1, a)(b, \overline{a}) = (b, 1) = b$ and thus, we would obtain the results and proofs of the previous section.

Lemma 5.5.4. *Let $G = P *_A (B \rtimes_\varphi A)$ and let $\pi : G \to P$ be a retract. Then we have $G \simeq \ker \pi \rtimes P$ and, more precisely,*

$$\ker \pi \simeq \mathop{*}_{\nu \in P/A} \left(B^A\right)^{(\nu)}.$$



The circuit for $\mathrm{WP}(G)$ *on input* w *returns true, if the gates for* $\mathrm{WP}(P), \mathrm{WP}(B)$ *and* $\mathrm{WP}(* B^{(\nu_i)})$ *return true.*

Figure 5.1: The steps in the reduction process for the word problem of $P *_A (B \times A)$ (see Theorem 5.5.3).

Proof. Since π is a retract, we have $G \simeq \ker \pi \rtimes P$. To see that $\ker \pi$ is isomorphic to $*_{\nu \in P/A} \left(B^A\right)^{(\nu)}$, we fix a set $\mathcal{R}$ of representatives of the cosets in P/A. Let $\varphi : *_{\nu \in \mathcal{R}} \left(B^A\right)^{(\nu)} \to \ker \pi$ be the homomorphism induced through

$$\varphi(((1,a)(b,\bar{a}))^{(\nu)}) = ((1,a)(b,\bar{a}))^{\nu}$$

We first show that φ is injective. Let $w = b_1^{(\nu_1)} \cdots b_n^{(\nu_n)} \neq 1$ be a reduced sequence in $*_{\nu \in \mathcal{R}} \left(B^A\right)^{(\nu)}$, that is $\nu_i A \neq \nu_{i+1} A$ for $1 \leq i < n$. Thus, we have $\nu_i \overline{\nu_{i+1}} \notin A$ and by the normal form theorem for amalgamated products (Theorem 5.5.1) it follows $\varphi(w) \neq 1$. Hence, the kernel of φ is trivial and φ must be injective.

To see that φ is surjective, let $w = p_1 b_1 \cdots p_n b_n \in \ker \pi$ and choose the representative ν_i of $p_1 \cdots p_i$ in $\mathcal{R}$, that is $\nu_i A = p_1 \cdots p_i A$. Therefore, for each i with $1 \leq i \leq n$ there exists some $a_i \in A$ such that $\nu_i a_i = p_1 \cdots p_i$. We obtain

$$p_1 \cdots p_i b \overline{p_1 \cdots p_i} = \nu_i a_i b \overline{a_i} \overline{\nu_i} = \nu_i (1,a_i)(b,\overline{a_i})\overline{\nu_i} = ((1,a_i)(b,\overline{a_i}))^{\nu_i}$$

for all $b \in B$. This leads to $w = ((1,a_1)(b_1,\overline{a_1}))^{\nu_1} \cdots ((1,a_n)(b_n,\overline{a_n}))^{\nu_n}$ in G and we finally have $\varphi(((1,a_1)(b_1,\overline{a_1}))^{(\nu_1)} \cdots ((1,a_n)(b_n,\overline{a_n}))^{(\nu_n)}) = w$. □

As can be seen in the proof, for $((1,a_i)(b,\overline{a_i}))$ with a_i given in the generators of P one must be able to effectively compute any representation of it in the generators

of B^A to obtain an algorithmic version of the above result. For this, let

$$\tau : \left(\Sigma_P^{\pm}\right)^* \times \left(\Sigma_B^{\pm}\right)^* \to \left(\Sigma_A^{\pm}\right)^* \times \left(\Sigma_B^{\pm}\right)^*$$

denote this transformation for the rest of this section. It may only be defined for valid inputs, that is for words $(u, v) \in \left(\Sigma_P^{\pm}\right)^* \times \left(\Sigma_B^{\pm}\right)^*$ we demand $u \in A$.

Theorem 5.5.5. *Let P, A and B be f. g. groups. Moreover, let $G = P *_A (B \rtimes_\varphi A)$. Then*

$$\mathrm{WP}(G) \in \mathsf{uAC}^0(\{\mathrm{WP}(P), \mathrm{WP}(B \rtimes_\varphi A), \mathrm{GWP}(A, P), \tau, \mathrm{WP}(F_2)\}),$$

that is the word problem of G is AC^0 Turing reducible to the word problems $\mathrm{WP}(P)$, $\mathrm{WP}(B \rtimes_\varphi A)$, $\mathrm{WP}(F_2)$, the function τ and the generalized word problem of A in P.

Proof. Let $w = w_1 \cdots w_n \in (\Sigma^{\pm})^*$ be the input word with $w_i \in \Sigma^{\pm}$. For $a \in \Sigma_A$ we denote by $a_P \in \left(\Sigma_P^{\pm}\right)^*$ a fixed word with $a =_P a_P$. The projections $\pi_P : (\Sigma^{\pm})^* \to \left(\Sigma_P^{\pm}\right)^*$ and $\pi_B : (\Sigma^{\pm})^* \to \left(\Sigma_B^{\pm}\right)^*$ are induced by

$$\pi_P(a) = \begin{cases} a, & a \in \Sigma_P^{\pm} \\ a_P, & a \in \Sigma_A^{\pm} \\ 1, & a \in \Sigma_B^{\pm} \end{cases} \qquad \text{and} \qquad \pi_B(a) = \begin{cases} 1, & a \in \Sigma_P^{\pm} \uplus \Sigma_A^{\pm} \\ a, & a \in \Sigma_B^{\pm} \end{cases}$$

The projection $\pi_P : G \to P$ is a retract and hence, by Lemma 5.5.2 we have $G \simeq *_{\nu \in P/A} B^{(\nu)} \rtimes P$. Let $p_i = \pi_P(w_i)$ and $b_i = \pi_B(w_i)$. Therefore, we have $p_i \in \left(\Sigma_P^{\pm} \uplus \{1\}\right)^*$ and $b_i \in \left(\Sigma_B^{\pm} \uplus \{1\}\right)$. We may assume $\pi_B(w) =_B 1$ for the rest of the construction. Next, the input word has to be rewritten as a part of the kernel. For this the words $g_i = p_1 \cdots p_i$ can be constructed in AC^0. By the same reasoning as in the previous lemma we have $w = g_1 b_1 \overline{g_1} \cdots g_n b_n \overline{g_n}$. Let $\mathcal{R} = \{1, \ldots, n\}$. We set

$$\begin{aligned} \nu_i &= \min\{1 \le j \le n \mid g_i A = g_j A\} \\ &= \min\{1 \le j \le n \mid g_i \overline{g_j} \in A\} \end{aligned}$$

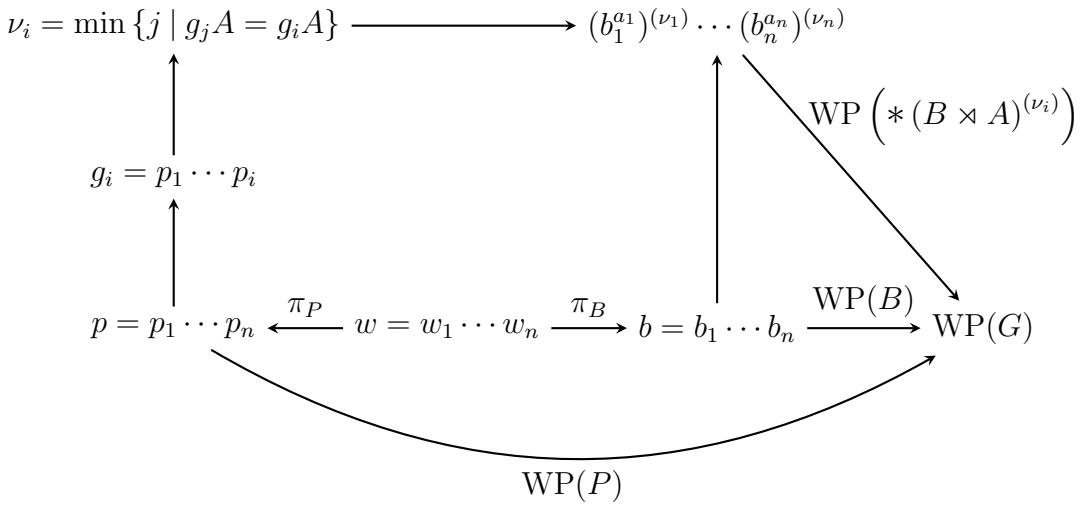
By using the oracle gates of $\mathrm{GWP}(A, P)$ for $g_i \overline{g_j} \in A$ the ν_i can be computed in AC^0. This is the point, where the semi-direct product has to be treated differently than the direct product in Theorem 5.5.3. Let $a_i = \overline{p_1 \cdots p_{\nu_i}} p_1 \cdots p_i \in \left(\Sigma_P^{\pm}\right)^*$. By construction we have $a_i \in A$, but it is given in the generators of P. We use the oracle gate for the function τ to compute $(\widetilde{a_i}, \widetilde{b_i}) = \tau(a_i, b_i)$ with $\widetilde{a_i} \in \left(\Sigma_A^{\pm}\right)^*$. With these we have $a_i b_i \overline{a_i} = \widetilde{a_i} \widetilde{b_i} \overline{\widetilde{a_i}}$.

Now, the input word w equals the identity, if and only if $\pi_B(w) =_B 1$ and

$$\left((1,\widetilde{a}_1)(\widetilde{b_1},\widetilde{\overline{a_1}})\right)^{(\nu_1)} \cdots \left((1,\widetilde{a}_1)(\widetilde{b_1},\widetilde{\overline{a_1}})\right)^{(\nu_n)}$$

is the identity in $*_{\nu\in\mathcal{R}}(B^A)^{(\nu)}$. Since B^A is a subgroup of $B \rtimes_\varphi A$ the oracle gate for $\mathrm{WP}(B \rtimes_\varphi A)$ can be used for the word problem of B^A. Finally, by Lemma 5.4.5 the result follows. □

The reduction process in the previous theorem is depicted in Figure 5.2 on page 62.



The circuit for $\mathrm{WP}(G)$ *on input* w *returns true, if the gates for* $\mathrm{WP}(P), \mathrm{WP}(B)$ *and* $\mathrm{WP}(*\,(B \rtimes_\varphi A)^{(\nu_i)})$ *return true.*

Figure 5.2: The steps in the reduction process for the word problem of $P *_A (B \rtimes_\varphi A)$ (see Theorem 5.5.5).

Example 5.5.6. *If* $B \rtimes_\varphi A = B \times A$, *then* $((1,a)(b,\bar{a}))$ *simply equals* $(1,1)(b,1)$ *in* B^A. *Hence, the transformation* τ *can be realized as* $\tau(a,b) = (1,b)$. *The word problem of* $B \rtimes A = B \times A$ *reduces to to the word problem of* A *and* B *(see Theorem 5.3.1 case 1). Since* A *is a subgroup of* P, *its word problem can be solved with oracle gates for the word problem of* P. *In conclusion, for direct products the statement in Theorem 5.5.5 is the same as in Theorem 5.5.3.*

Example 5.5.7. *Let* $G = \langle t, a_1, \ldots, a_n \mid ta_i\bar{t} = a_i^2 \rangle$. *We use Theorem 5.5.5 to show that its word problem can be solved in* $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$. *For the group* G *we have*

$$\begin{aligned} G &\simeq \langle t, a_1, \ldots, a_{n-1} \mid ta_i\bar{t} = a_i^2 \rangle *_{\langle t\rangle = \langle s\rangle} \langle s, a \mid sa\bar{s} = a^2 \rangle \\ &\simeq \langle t, a_1, \ldots, a_{n-1} \mid ta_i\bar{t} = a_i^2 \rangle *_{\langle t\rangle = \langle s\rangle} \mathbb{Z}[\tfrac{1}{2}] \rtimes_\varphi \langle s \rangle, \end{aligned}$$

where $\varphi(s)(a) = 2a$. *The group* $\mathbb{Z}[\frac{1}{2}] \rtimes_\varphi \langle s\rangle$ *is the Baumslag-Solitar group* $\mathrm{BS}(1,2)$. *The Baumsalg-Solitar group* $\mathrm{BS}(1,2)$ *is a linear group. Therefore, its word problem is solvable in* $\mathsf{C}_=\mathsf{NC}^1$. *Additionally, the word problem of the free group is solvable in* $\mathsf{C}_=\mathsf{NC}^1$. *Hence, the group* G *has a word problem in* $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$ *by using induction.*

5.6 in graph products

We start this section by stating two immediate results on special graph products. These two results depend on the linearity of these specific graph products. The general result for graph products is more involved. Using the linearity is not an approach to go for since not all graph products are linear and even for graph products over linear groups it is an open question if it is linear again. Fortunately, for arbitrary graph products the results are almost as good as for graph products that are linear groups.

Theorem 5.6.1. *Let* G *be a fixed graph group or a right angled Coxeter group. Then the word problem of* G *is in* $\mathsf{C}_=\mathsf{NC}^1$.

Proof. Right angled Coxeter groups are linear and graph groups are subgroups of right angled Coxeter groups. The corresponding matrices do only contain integer values. Since the problem V-ItMatProd$_\mathrm{N}$ of multiplying a sequence of $n \times n$ integer matrices (with n fixed) is complete for $\mathsf{C}_=\mathsf{NC}^1$, the result follows. □

Theorem 5.6.2. *Let* G *be a graph product over finite groups and the infinite cyclic group. Then the word problem of* G *is in* $\mathsf{C}_=\mathsf{NC}^1$.

Proof. Graph products of finite groups are linear by Corollary 3.3.7. The group G embeds into a graph product of finite groups since the infinite cyclic group embeds into $\mathbb{Z}/2\mathbb{Z} * \mathbb{Z}/2\mathbb{Z}$. Again, by using the problem V-ItMatProd$_\mathrm{N}$ the result follows. □

Lemma 5.6.3. *Let* $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ *be a graph product of f. g. groups. Moreover, let* $G_\mathcal{S} = \mathrm{GP}(\mathcal{S}, I_\mathcal{S}; (G_\alpha)_{\alpha\in\mathcal{S}})$ *be a subgroup with* $\mathcal{S} \subseteq \mathcal{L}$ *and* $I_\mathcal{S} = I \cap \mathcal{S} \times \mathcal{S}$. *Then*

$$\mathrm{GWP}(G_\mathcal{S}, G) \in \mathsf{uAC}^0(\mathrm{WP}(G)),$$

that is there is an AC^0 *Turing reduction of the generalized word problem onto the word problem of* G *for subgroups induced by a subset of* $\mathcal{L}$. *Moreover, the reduction is an* AC^0 *many-one reduction.*

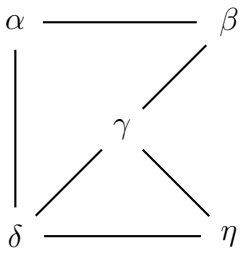
Proof. Consider the projection $\pi : \Gamma^* \to \Gamma_{\mathcal{S}}^*$ induced by

$$\pi(a) = \begin{cases} a & \text{if } \operatorname{alph}(a) \in \mathcal{S} \\ 1 & \text{otherwise.} \end{cases}$$

It holds $w = \pi(w)$ in G if and only if $w \in G_{\mathcal{S}}$. Therefore, we have $w \in \mathrm{GWP}(G_{\mathcal{S}}, G)$ if and only if $w^{-1}\pi(w) \in \mathrm{WP}(G)$. The projection can be computed in AC^0, hence the result follows. □

Example 5.6.4. *For the proof of the following theorem, any graph product can inductively be written as an amalgamated product. For the idea, consider the graph product G as depicted as follows.*

$G =$ *graph product over*



The link of β is $\{\alpha, \gamma\}$ and let A be the free product $A = G_\alpha * G_\gamma$. Removing the node β we obtain a smaller graph and the link of α becomes the singleton $\{\delta\}$. Removing α leaves us with a triangle with nodes γ, δ, η which yields the direct product $G_\gamma \times G_\delta \times G_\eta$. Going backwards we see that P is the amalgamated product

$$P = (G_\alpha \times G_\delta) *_{G_\delta} (G_\gamma \times G_\delta \times G_\eta)$$

which contains A. Finally, $G = P *_A (G_\beta \times A)$.

Theorem 5.6.5. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f. g. groups. Then*

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

that is there is a uniform AC^0 Turing reduction of the word problem of G to the word problem of each G_α and the word problem of F_2.

Proof. We proof the theorem by induction on the cardinality of $\mathcal{L}$. Let G_β be any group with $\beta \in \mathcal{L}$, $\mathcal{L}' = \mathcal{L} \setminus \{\beta\}$ and $I' = I \cap (\mathcal{L}' \times \mathcal{L}')$. Furthermore, let $P = \mathrm{GP}(\mathcal{L}', I'; (G_\alpha)_{\alpha\in\mathcal{L}'})$ and

$$\mathrm{link}(G_\beta) = \mathrm{GP}\left(\mathrm{link}(\beta), I \cap (\mathrm{link}(\beta) \times \mathrm{link}(\beta)); (G_\alpha)_{\alpha\in\mathrm{link}(\beta)}\right)$$

where $\mathrm{link}(\beta)$ is the link of β in the independence graph. For the graph product we obtain $G = P *_{\mathrm{link}(G_\beta)} (\mathrm{link}(G_\beta) \times G_\beta)$. By induction, there is an AC^0 Turing reduction of the word problem of P and of $\mathrm{link}(G_\beta)$ to the word problem of F_2 and the word problem of each G_α. The generalized word problem of G_β in P is AC^0 Turing reducible to the word problem of each G_α by Lemma 5.6.3. Now, the theorem follows by Theorem 5.5.3. □

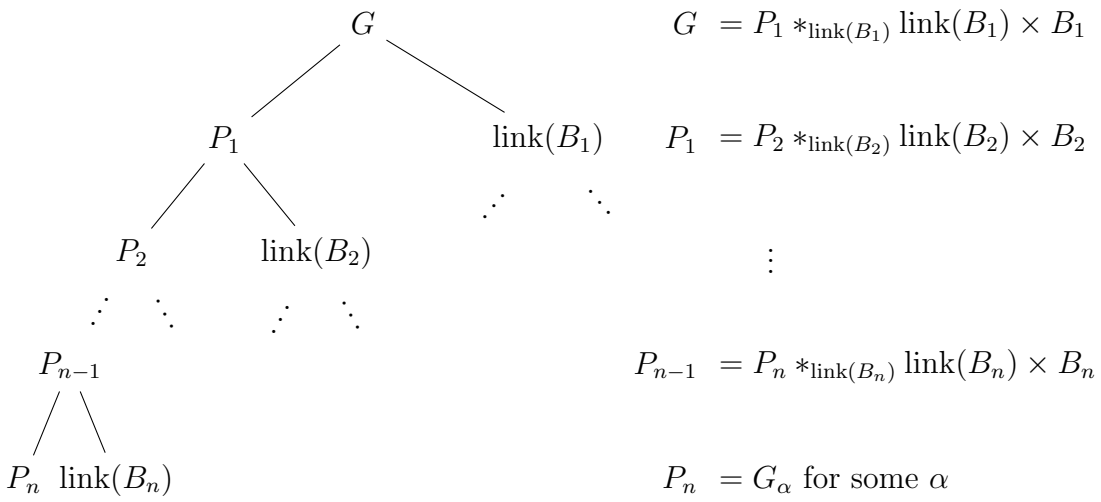


Figure 5.3: The graph depicts the induction process in the proof for the word problem of graph products (see Theorem 5.6.5).

Remark 5.6.6. The reduction in Theorem 5.6.5 uses the word problem of the free group on two generators. In almost all cases, the word problem of the free group on two generators is required. The only case, where it is not required is for direct products and the group $\mathbb{Z}/2\mathbb{Z} * \mathbb{Z}/2\mathbb{Z}$. All other graph products do contain the free group on two generators as a subgroup. Hence, it is inevitable to include it.

The following corollary is an immediate consequence of the previous theorem and remark. It implies that any hardness (respectively completeness) result of graph groups directly transfers to an hardness (respectively completeness) result of free groups and vice versa.

Corollary 5.6.7. *Let G be a fixed graph group, which is not free abelian. Then the word problem of G is AC^0 Turing equivalent to the word problem of the free group F_2 on two generators.*

The reduction used in Theorem 5.6.5 is a Turing reduction. Is it possible to transform this reduction into an AC^0 many-one reduction? As an intermediate step, one could try to only use oracle gates as output gates. A nice result would be to have the following: If the base groups have a word problem in $\mathsf{C}_=\mathsf{NC}^1$ (or weaker in NC^1), then the graph product has a word problem in $\mathsf{C}_=\mathsf{NC}^1$. If the reduction would get along with only using oracle gates as output gates, then one would obtain the desired behavior for $\mathsf{C}_=\mathsf{NC}^1$. The reason is that AC^0 can be simulated in $\mathsf{C}_=\mathsf{NC}^1$. It might also be possible that the Turing reduction is sufficient for this goal. If $\mathsf{C}_=\mathsf{NC}^1$ is closed under complement, then $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1) \stackrel{?}{=} \mathsf{C}_=\mathsf{NC}^1$ holds. However, closure under complement is still an open question for $\mathsf{C}_=\mathsf{NC}^1$.

The background for this question is the complexity of the word problem of right angled Coxeter groups and graph groups. By Theorem 5.6.1 their word problem is contained in $\mathsf{C}_=\mathsf{NC}^1$. However, the proof of Theorem 5.6.1 is deeply linked to the linearity of right angled Coxeter groups. Alternatively, with Theorem 5.6.5 and without using the linearity, the complexity is $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$. For this, let $\langle a \rangle$ and $\langle a \mid a^2 \rangle$ be the representation of $\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z}$. With these representations we have $\mathrm{WP}(\mathbb{Z}) \in \mathsf{uTC}^0$ and $\mathrm{WP}(\mathbb{Z}/2\mathbb{Z}) \in \mathsf{uTC}^0$. By Theorem 5.2.3 the word problem $\mathrm{WP}(F_2)$ of the free group is in $\mathsf{C}_=\mathsf{NC}^1$. Hence, with Theorem 5.6.5, one obtains only the slightly weaker result of $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$ for the complexity of the word problem of right angled Coxeter groups and graph groups.

5.6.1 Uniform solution for graph groups

The induction process in the proof of Theorem 5.6.5 is depicted in Figure 5.3 on page 65. As we remove in each induction step only one group G_β, the leftmost path has length $n = |\mathcal{L}|$. Observe that in the worst case 2^n different groups are constructed in the induction process. Moreover, we have exponentially many calls to the word problem of any of these (smaller) graph products. Thus, for this proof, it is of high importance that the group G is fixed for the algorithm and not part of the input. Hence, we can't say anything practical on the uniform word problem of graph products using this algorithm. Fortunately, by using a different approach we can construct an algorithm for the uniform word problem of graph groups in $\mathsf{C}_=\mathsf{L}$. The main point of this result is the linearity of right angled Coxeter groups. However, it remains an open question, whether their uniform word problem is complete for $\mathsf{C}_=\mathsf{L}$. But we can show that the uniform word problem of graph groups, and hence of right angled Coxeter groups is NL-hard. Thus, the complexity of the uniform word problem of graph groups is sandwiched between NL and $\mathsf{C}_=\mathsf{L}$. In the next section, we will extend this result to arbitrary graph products.

Theorem 5.6.8. *The uniform word problem of right angled Coxeter groups and graph groups is contained in* $\mathsf{C}_=\mathsf{L}$.

Proof. The problem V-ItMatProd is $\mathsf{C}_=\mathsf{L}$-complete. Moreover, the complexity class $\mathsf{C}_=\mathsf{L}$ is closed under LOGSPACE many-one reductions. Hence, we reduce the uniform word problem to the problem V-ItMatProd in LOGSPACE.

By [HW99] every graph group embeds into a right angled Coxeter group as follows. Let $G = \mathrm{GP}(\Sigma, I; (G_\alpha)_{\alpha\in\Sigma})$ with $G_\alpha \simeq \mathbb{Z}$. Each generator $a \in \Sigma$ is replaced by two new generators a_1, a_2. Set $\Sigma' = \{a_1, a_2 \mid a \in \Sigma\}$ and

$$I' = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2))) \mid (a, b) \in I\}.$$

The group $G' = \mathrm{GP}(\Sigma', I'; (G_\alpha)_{\alpha\in\Sigma'})$ with $G_\alpha \simeq \mathbb{Z}/2\mathbb{Z}$ is a right angled Coxeter group. The homomorphism $\varphi : G \to G'$ defines an embedding via $\varphi(a) = a_1a_2$. This embedding is LOGSPACE-computable. Hence, it suffices to consider only right angled Coxeter groups.

Right angled Coxeter groups are linear over $\mathbb{Z}$, see Section 3.2. An entry of the matrix corresponding to σ_a does only depend on the dependency relation. Hence, the sequence of matrices representing the input word can be computed in LOGSPACE. The result follows since the input word represents the identity if and only if the product of the matrices equals the identity matrix.

□

Theorem 5.6.9. *The uniform word problem in graph groups is* NL*-hard.*

Proof. We reduce the NL-complete problem $\overline{\text{Gap}}$ to the word problem in a graph group. First, we construct a new graph with an easy topological ordering of its nodes. Let the graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ be given. We define the new graph $G' = (V', E')$ with $V' = (V \times [n]) \cup \{s, t\}$, where $[n] = \{1, \ldots, n\}$. Its set of edges is given by

$$((u, i), (v, i + 1)) \in E' \iff u = v \text{ or } (u, v) \in E$$

for all $1 \leq i < n$. Moreover, we add the edges $(s, (s, 1))$ and $((t, n), t)$. There is an $s \xrightarrow{*} t$ path in G if and only if there is an $s \xrightarrow{*} t$ path in G'. A topological ordering of the nodes in G' is

$$s, (v_1, 1), (v_2, 1), \ldots, (v_n, 1), \ldots, (v_1, n), (v_2, n), \ldots, (v_n, n), t.$$

For the graph group, let $\Sigma = V'$ and the independence relation I be given by

$$(u, v) \in I \iff (u, v) \notin E' \text{ and } (v, u) \notin E'.$$

For the word $w_G = (v_1, 1)(v_2, 1) \cdots (v_n, 1) \cdots (v_1, n)(v_2, n) \cdots (v_n, n)$ we obtain

$$w = \bar{t}\overline{w_G} s w_G t \overline{w_G} \bar{s} w_G = 1 \iff (G, s, t) \in \overline{\text{GAP}}.$$

For the proof of the claim, we first show that $w \neq 1$ in the graph group if there is some $s \xrightarrow{*} t$ path in G. To see this, let P be the set of nodes on any $s \xrightarrow{*} t$ in G'. Let $\pi_P : \Sigma^* \to \Sigma^*$ be the projection defined by

$$v \mapsto \begin{cases} v, & \text{if } v \in P \\ 1, & \text{otherwise.} \end{cases}$$

$\pi_P(w)$ is a word, whose dependence graph is a single path. Hence, no reduction does occur in the dependence graph and therefore $\pi_P(w) \neq 1$. But then we also must have $w \neq 1$ in the graph group.

For the other direction, let there be no $s \xrightarrow{*} t$ path in G. Further, let T be the set of nodes from which t is reachable, that is

$$T = \{v \in V' \mid \exists v \xrightarrow{*} t \text{ path}\}$$

Furthermore, let $S = V' \setminus T$. Then - by construction - there is no path from a node in S to a node in T. Note that there might be a path from a node in T to a node in S.

We will now split the word w_G into $w_t w_s$ such that $w_G = w_t w_s$ in the graph group. The graph G' is the dependence graph of w_G. Hence, we can consider subgraphs of G' as dependence graphs. Let w_s be any word representing the dependence graph induced by $S \setminus \{s\}$ and w_t be any word representing the dependence graph induced by $T \setminus \{t\}$. We obtain $w_G = w_t w_s$ in the graph group since there is no path from a node in S to a node in T. Moreover, we have $s \in S$ and $t \in T$. Since nodes in T are not reachable from nodes in S, we obtain $w_t s = s w_t$ and $w_s t = t w_s$ in the graph group. By construction, we have $st = ts$. Taking all this together yields

$$\begin{aligned} w &= \bar{t}\overline{w_G} s w_G t \overline{w_G} \bar{s} w_G \\ &= \bar{t}\overline{w_t w_s} s w_t w_s t \overline{w_t w_s} \bar{s} w_t w_s \\ &= \bar{t}\overline{w_s}\overline{w_t} s w_t w_s t \overline{w_s}\overline{w_t} \bar{s} w_t w_s \\ &= \bar{t}\overline{w_s} s w_s t \overline{w_s}\overline{w_t} \bar{s} w_t w_s \\ &= \bar{t}\overline{w_s} s t \overline{w_t} \bar{s} w_t w_s \\ &= \bar{t}\overline{w_s} t s \overline{w_t} \bar{s} w_t w_s \\ &= \overline{w_s} s \bar{s} w_s = 1 \end{aligned}$$

in the constructed graph group. □

5.6.2 Uniform solution for graph products

We first rephrase the hardness result for an arbitrary class of groups.

Corollary 5.6.10. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then we have that* $\mathrm{WP}(\mathbf{GP}\mathcal{C})$ *is* NL*-hard.*

Proof. Let $H \in \mathcal{C}$ and $1 \neq x \in H$ be a fixed element. Then $\mathbb{Z} = \langle a \rangle$ embeds into $H^{(1)} * H^{(2)}$ via $a \mapsto x^{(1)}x^{(2)}$. Hence, any graph group embeds into a graph product over $\mathcal{C}$. Therefore, the statement follows by Theorem 5.6.9. □

Now, for graph products the solution of the uniform word problem is more involved. We will utilize the linearity of graph products as it has been introduced in Section 3.3. Since the dimension of the underlying vector space is in general infinite, we cannot reduce the word problem to multiplying a sequence of integer matrices.

The idea for the solution is as follows. Let $G = \mathrm{GP}\left(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}}\right)$ be a graph product. An extra free group $\langle x \rangle$ which is dependent to all other groups is added to the graph product. Let σ_w be the linear map as defined in Section 3.3. On input of a word w the mapping σ_w is applied to x. As we will see, only a finite number of coefficients of $\sigma_w(x)$ is non-zero. Moreover, $\sigma_w(x) = x$ if and only if $w =_G 1$, where G is the graph product. Any coefficient, which might be non-zero is a subword of $\pi_\alpha(w)$ for some $\alpha \in \mathcal{L}$. Hence, it suffices to check the coefficients of all those subwords and use that $\mathsf{C}_=\mathsf{L}$ is closed under polynomially many conjunctions.

Lemma 5.6.11. *Let the group $G = \mathrm{GP}\left(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}}\right)$ be a graph product and let $w = w_1 \cdots w_n \in \Gamma^*$ be some word with $w_i \in \Gamma$. Moreover, let*

$$\sigma_w(x) = \sum_{a\in\Gamma} \lambda_a \cdot a.$$

Then we have

1. *$\sigma_w(x) = x$ if and only if $w =_G 1$,*
2. *$\lambda_a \neq 0$ implies $a =_G \pi_\alpha(w_k \cdots w_l)$ for some $1 \leq k \leq l \leq n$.*

Proof. Let $\widehat{w} = \widehat{w}_1 \cdots \widehat{w}_n \in \Gamma^*$ be reduced with $w =_G \widehat{w}$. Statement *1.* follows immediately by Lemma 3.3.4. For Statement *2.*, any factor $\pi_\alpha(\widehat{w}_i \cdots \widehat{w}_j)$ equals in G a factor $\pi_\alpha(w_k \cdots w_l)$ since in the reduction process only neighbored nodes in the dependence graph are merged. Hence, Statement *2.* follows by Lemma 3.3.4 either. □

Algorithm 5.1 GapL algorithm for computing the coefficient of $a \in \Gamma_\alpha$ of $\sigma_w(x)$

function COEFF$(a, w_1 \cdots w_n)$
  ▷ Two paths for $\sigma_{w_n}(x) = 2w_n + x$
  Guess 'Branch 1' or 'Branch 2'
  if 'Branch 1' **or** 'Branch 2' **then** $(k, \ell, i, s) \leftarrow (n, n, n-1, 1)$

  while true **do**
    if $i = 0$ **then**
      if $a =_{G_\alpha} \pi_\alpha(w_k \cdots w_\ell)$ **then**
        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
    end if
    ▷ First, two branches for $\sigma_{w_i}(x) = 2w_i + x$
    Guess 'Branch 1' or 'Branch 2'
    if 'Branch 1' **or** 'Branch 2' **then** $(k, \ell, i, s) \leftarrow (i, i, i-1, s)$

    ▷ Second, branches for $\sigma_{w_i}(\pi_\alpha(w_k \cdots w_\ell))$
    if $\mathrm{alph}(w_i) = \mathrm{alph}(w_k)$ **then**
      Guess 'Branch 1' or 'Branch 2'
      if 'Branch 1' **then** $(k, \ell, i, s) \leftarrow (i, \ell, i-1, s)$
      if 'Branch 2' **then** $(k, \ell, i, s) \leftarrow (i, i, i-1, -s)$
    end if
    if $(\mathrm{alph}(w_i), \mathrm{alph}(w_k)) \notin I$ **then**
      Guess 'Branch 1', 'Branch 2' or 'Branch 3'
      if 'Branch 1' **or** 'Branch 2' **then** $(k, \ell, i, s) \leftarrow (i, i, i-1, s)$
      if 'Branch 3' **then** $(k, \ell, i, s) \leftarrow (k, \ell, i-1, s)$
    end if
    if $(\mathrm{alph}(w_i), \mathrm{alph}(w_k)) \in I$ **then** $(k, \ell, i, s) \leftarrow (k, \ell, i-1, s)$
  end while
end function

Computing any coefficient of $\sigma_w(x)$ on input of a graph group and a word can be done in GapL with access to an oracle for the word problem of each base group. This statement is crucial for the uniform word problem of graph products and phrased in detail in the following lemma.

Lemma 5.6.12. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Let the input be groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I, a word $w \in \Gamma^*$ and an element $a \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$. Then the coefficient of a in $\sigma_w(x)$ with $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ for $\mathcal{L} = \{1, \ldots, n\}$ can be computed in $\mathsf{GapL}^{\mathrm{WP}(\mathcal{C})}$.*

Proof. Let $w = u_0 a_1 u_1 \cdots a_n u_n$ be the α-factorization of w. By the construction of σ_w the only non-zero coefficients of $\sigma_w(x)$ occurring during the computation are of the form $a_k \cdots a_l$ for $1 \leq k \leq l \leq n$.

On the work tape k and l, the current index i in the input word in binary and a sign $s = \pm 1$ are stored. We denote such a configuration by the quadruple (k, ℓ, i, s). Hence, logarithmic space does suffice. Let $w = w_1 \cdots w_n$. The algorithm Algorithm 5.1 describes the computation. At any stage, the computation can be considered as a tree. The leafs denote the coefficients of $\sigma_{w_i \cdots w_n}(x)$ as follows. Let $S_x = \{(k, \ell, s) \mid \pi_\alpha(w_k \cdots w_\ell) =_G x \text{ and } (k, \ell, *, s) \text{ is a leaf}\}$. Then the coefficient of any $x \in \Gamma$ is

$$\sum_{(k,\ell,s)\in S_x} s.$$

When the whole input word is read only the leafs of a do survive. Hence, the coefficient of a equals the number of accepting paths minus the number of rejecting paths. □

Corollary 5.6.13. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Let the input be groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I, a word $w \in \Gamma^*$ and an element $a \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$. Then verifying the coefficient of a in $\sigma_w(x)$ with $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ for $\mathcal{L} = \{1, \ldots, n\}$ is in $\mathsf{C}_=\mathsf{L}^{\mathrm{WP}(\mathcal{C})}$.*

Theorem 5.6.14. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then*

$$\mathrm{WP}(\mathbf{GP}\mathcal{C}) \in \mathsf{C}_=\mathsf{L}^{\mathrm{WP}(\mathcal{C})},$$

that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I, a word $w \in \Gamma^*$ the word problem $w = 1$ in $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ for $\mathcal{L} = \{1, \ldots, n\}$ can be solved in $\mathsf{C}_=\mathsf{L}$ with oracles for the word problem in $\mathcal{C}$.

Proof. By Lemma 5.6.11 the coefficients of $\sigma_w(x)$ are all zero, if w equals the identity. Moreover, if w is not the identity, then there exists $\alpha \in \mathcal{L}$ and $1 \leq k \leq \ell \leq n$ such that the coefficient of $\pi_\alpha(w_k \cdots w_\ell)$ is non-zero. By Corollary 5.6.13 verifying that any of these coefficients is zero is in $\mathsf{C}_=\mathsf{L}^{\mathrm{WP}(\mathcal{C})}$. Since $\mathsf{C}_=\mathsf{L}$ is closed under conjunctive truth table reductions, the statement follows. □

Example 5.6.15. *The uniform word problem of graph products over finite groups is contained in* $\mathsf{C}_=\mathsf{L}$*, when the finite groups are given by their multiplication table.*

Proof. By Example 5.1.4 the uniform word problem of finite groups given by their multiplication table is $\mathsf{LOGSPACE}$-complete. Together with Theorem 5.6.14 we obtain that the uniform word problem of graph products over finite groups is in $\mathsf{C}_=\mathsf{L}^{\mathsf{LOGSPACE}}$. Finally, with $\mathsf{C}_=\mathsf{L}^{\mathsf{LOGSPACE}} = \mathsf{C}_=\mathsf{L}$ the result follows. □

We close this section with the following statement. Solving the word problem by using the linear embedding approach is complete for $\mathsf{C}_=\mathsf{L}$. However, it might be possible to solve the word problem with another approach in a complexity class below $\mathsf{C}_=\mathsf{L}$. To show the hardness result we will use the following completeness result.

Theorem 5.6.16 ([ST98])**.** *The problem* V-Paths *defined below is* $\mathsf{C}_=\mathsf{L}$*-complete.*

Problem:	V-Paths
Input:	*An acyclic directed graph* $G = (V, E)$*, two nodes* $s, t \in V$ *and a natural number* $m \in \mathbb{N}$*.*
Question:	*Is the number of* $s \xrightarrow{*} t$ *paths equal to* m*?*

Theorem 5.6.17. *The verification problem of the mapping* $\sigma_w(x)$ *on input of a word* $w \in \Sigma^*$ *and a graph group is complete for* $\mathsf{C}_=\mathsf{L}$*.*

Proof. For the hardness, we reduce the problem V-Paths to the verification of the coefficients of the mapping σ_w applied to some vector. By [ST98] the problem V-Paths is complete for $\mathsf{C}_=\mathsf{L}$. First, we construct a new graph with an easy topological ordering of its nodes with the additional property that all $s - t$ paths have length $n + 1$. Let the graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ be given. We define the new graph $G' = (V', E')$ with $V' = (V \times [n]) \cup \{s, t\}$, where $[n] = \{1, \ldots, n\}$. Its set of edges is given by

$$((u, i), (v, i + 1)) \in E' \iff u = v = t \textbf{ or } (u, v) \in E$$

for all $1 \leq i < n$. Moreover, we add the edges $(s, (s, 1))$ and $((t, n), t)$. There is a one-to-one correspondence between $s \xrightarrow{*} t$ paths in G and $s \xrightarrow{*} t$ paths in G'. Moreover, any $s \xrightarrow{*} t$ path in G' has length $n + 1$. A topological ordering of the nodes in G' is

$$s, (v_1, 1), (v_2, 1), \ldots, (v_n, 1), \ldots, (v_1, n), (v_2, n), \ldots, (v_n, n), t.$$

For the graph group, let $\Sigma = V'$ and the independence relation I be given by

$$(u, v) \in I \iff (u, v) \notin E' \textbf{ and } (v, u) \notin E'.$$

Let $w = s \cdot (v_1, 1) \cdot (v_2, 1) \cdots (v_n, 1) \cdots (v_1, n) \cdot (v_2, n) \cdots (v_n, n)$. Now, all $s \xrightarrow{*} t$ paths have length $n + 1$. More precisely, let $v = (v_k, i) \in E'$ be any node. The length of any $v \xrightarrow{*} t$ path is $(n - i + 1)$. We will show the following claim:

> *The number of $v \xrightarrow{*} t$ paths is equal to m if and only if the coefficient of v in $\sigma_w(t)$ equals $(2^{n-i+1} \cdot m)$.*

In particular, the number of $s \xrightarrow{*} t$ paths equals m if and only if the coefficient of s in $\sigma_w(t)$ equals $(2^{n+1} \cdot m)$. For the number of paths from v to t the equation

$$\#\textsc{Paths}(v, t) = \sum_{(v,u) \in E'} \#\textsc{Paths}(u, t)$$

holds. Note that $\#\textsc{Paths}(t, t) = 1$. Let vu be any suffix of w with $u = u_1 \cdots u_l$. Denote by $level(v) = i$ the level of the node $v = (v_k, i)$. For the node s we set $level(s) = 0$ and for the node t we set $level(t) = n + 1$. Moreover let

$$\sigma_u(t) = \sum_{k=1}^{l} 2^{n-level(i)+1} \cdot \#\textsc{Paths}(u_k, t) \cdot v,$$

then we have

$$\begin{aligned}
\sigma_{vu}(t) &= \sigma_v(\sigma_u(t)) \\
&= \sum_{(v,u_k) \in E'} 2^{n-level(u_k)+1} \cdot \#\textsc{Paths}(u_k, t) \cdot (2v + u_k) + \\
&\quad \sum_{(v,u_k) \notin E'} 2^{n-level(u_k)+1} \cdot \#\textsc{Paths}(u_k, t) \cdot u_k \\
&= \sum_{k=1}^{l} 2^{n-level(i)+1} \cdot \#\textsc{Paths}(u_k, t) \cdot v + 2^{n-level(v)+1} \cdot \#\textsc{Paths}(v, t) \cdot v
\end{aligned}$$

The last equation follows since there are only edges from level i to level $i + 1$. Hence, the claim follows by induction. □

Chapter 6

Geodesics and normal forms

Often, one is not only interested in solving the word problem, but also in finding a *geodesic* or a *normal form*. In particular, we do need a solution of the geodesic problem to solve the conjugacy problem in free products and graph products in Chapter 7. A geodesic representative of some group element is a shortest word representing this group element. Thus, geodesics do depend on the particular alphabet of the group. In our case, for some group product G over the groups G_α this is the alphabet $\Gamma = \biguplus_{\alpha \in \mathcal{L}} \Gamma_\alpha$. The set of geodesics representing the group element of some word $w \in \Gamma^*$ is

$$\{u = u_1 \cdots u_n \in \Gamma^* \mid u_1 \cdots u_n \text{ is a factorization, } n = \|w\| \text{ and } u =_G w\}.$$

For some group G we denote any function computing a geodesic on input of some word by geo_G and for some class $\mathcal{C}$ of groups by $\mathrm{geo}_{\mathcal{C}}$. The (uniform) *geodesic problem* is the computation of any geodesic function on input of a word. It is important to note that a geodesic function is not unique in general. However, for free products the geodesic function is unique. Thus, in this case a geodesic word directly leads to a normal form. For direct products and graph products we equip the list $\mathcal{L}$ of groups with a linear order $<_{\mathcal{L}}$. The length lexicographic normal form of some word $w \in \Gamma^*$ is the lexicographically first word in the set of geodesics. A function $\mathrm{nf} : \Gamma^* \to \Gamma^*$ is a normal form function of some group G if $\mathrm{nf}(u) = \mathrm{nf}(v)$ whenever $u =_G v$. We denote a normal form function of some group G by nf_G and of some class $\mathcal{C}$ of groups by $\mathrm{nf}_{\mathcal{C}}$. The (uniform) *normal form problem* is the computation of such a normal form function on input of a word. For all normal form functions nf_G we may assume that the identity is mapped to the identity, that is $\mathrm{nf}_G(1) = 1$. This can easily be ensured in AC^0.

6.1 in direct products

For direct products, there are two options for representing elements of the direct product. Either as a tuple or as a sequence of factors. When representing the elements as a tuple the geodesic and normal form problem can be solved in AC^0 by

a simple rearrangement. However, when representing the elements as a sequence of factors, that is $w = w_1 \cdots w_n \in \Gamma^*$, then for a geodesic one would like to have $w_i \neq 1$ in G for all $1 \leq i \leq n$. The consequence of this is that the uniform geodesic problem and thus also the uniform normal form problem requires the full power of TC^0. For a fixed direct product the power of AC^0 is sufficient. We stick to the representation as a sequence of factors since it is in line with all other group products considered in this thesis.

Theorem 6.1.1. *Let $G = \prod_{\alpha \in \mathcal{L}} G_\alpha$ be a direct product of f. g. groups. Then*

$$\mathrm{geo}_G \in \mathsf{uAC}^0(\{\mathrm{WP}(G_\alpha) \mid \alpha \in \mathcal{L}\}),$$

that is there is an AC^0 Turing reduction of the geodesic problem of G to the word problem of each G_α.

Proof. Let $w = w_1 \cdots w_n \in \Gamma^*$ be the factorization of the input word. We fix an enumeration of the $\alpha \in \mathcal{L}$. Hence, we may assume $\mathcal{L} = \{1, \ldots, |\mathcal{L}|\}$. For $\alpha \in \mathcal{L}$ let π_α be the projection of G onto G_α.

$$\pi_\alpha(w_k) = \begin{cases} w_k, & \text{if } \mathrm{alph}(w_k) = \mathrm{bin}(\alpha) \\ 1, & \text{otherwise.} \end{cases}$$

Now, for direct products we have

$$w =_G \pi_1(w) \cdots \pi_{|\mathcal{L}|}(w).$$

However, it might be the case that $\pi_i(w) =_G 1$ for some $1 \leq i \leq |\mathcal{L}|$. These factors $\pi_i(w)$ must be omitted in the output. Let $pos(i,k)$ be the predicate, which is true if the position of $\pi_i(w)$ in the output is k. Since $|\mathcal{L}|$ is finite, only finitely many positions have to be checked. Let $W(i)$ be a shorthand for the complement of the word problem of each factor, that is

$$W(i) \equiv \pi_i(w) \notin \mathrm{WP}(G_i)$$

then the predicate $pos(i,k)$ is equivalent to

$$pos(i,k) \equiv \bigvee_{P \in \binom{\{1,\ldots,i-1\}}{k-1}} \left(\bigwedge_{t \in P} W(t) \wedge \bigwedge_{f \in \{1,\ldots,i-1\} \setminus P} \neg W(f) \right) \wedge W(i)$$

The formula is true, if $\pi_i(w) \neq_G 1$ and if exactly $k-1$ factors before the factor $\pi_i(w)$ do not vanish. Moreover, there are only polynomially many subsets P. Hence, the formula can be realized in AC^0. □

The next theorem states the reduction for the uniform geodesic problem. Its approach is primarily the same as for the non-uniform variant. However, the power of TC^0 is required for the reduction, since the number of groups G_i is not bounded by a constant in the uniform setting.

Theorem 6.1.2. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then*

$$\mathrm{geo}_{\mathbf{\Pi}\mathcal{C}} \in \mathsf{uTC}^0(\mathrm{WP}(\mathcal{C})),$$

that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$ and a word $w \in \Gamma^*$ computing a geodesic of w in $G = \prod_{i=1}^{n} G_i$ is in uniform TC^0 with oracle gates for the uniform word problem of $\mathcal{C}$.

Proof. The proof is essentially the same as for the previous theorem. The only difference is the positioning of the factors in the output. Let $w = w_1 \cdots w_n \in \Gamma^*$ be the factorization of the input words. Let G_i be the i-th group of the input and π_i be the projection of G onto G_i, that is

$$\pi_i(w_k) = \begin{cases} w_k, & \text{if } \mathrm{alph}(w_k) = \mathrm{bin}(i) \\ 1, & \text{otherwise.} \end{cases}$$

Now, for direct products we have

$$w =_G \pi_1(w) \cdots \pi_n(w).$$

As in the previous theorem, the $\pi_i(w)$ might equal the identity for some $1 \leq i \leq n$. Let $pos(i,k)$ be true if the position of $\pi_i(w)$ is k in the output. Let again $W(i)$ be a shorthand for the complement of the word problem of each factor, that is

$$W(i) \equiv (\pi_i(w), G_i) \notin \mathrm{WP}(\mathcal{C})$$

Now, the predicate $pos(i,k)$ can be realized through the TC^0-circuit

$$pos(i,k) \equiv W(i) \wedge ((k-1) = (\#j < i : W(j)))$$

The circuit evaluates to true if $\pi_i(w)$ is not the identity and if exactly $k-1$ factors before $\pi_i(w)$ do not vanish. □

Computing a normal form in direct products is obtained by first computing a geodesic and then applying the normal form function on each factor. Hence, for direct products computing a normal form is in uniform AC^0 with oracle gates for the geodesic and normal form problem. The next theorem states the complexity of the normal form problem when only oracle gates for the normal form problem of the base groups are used.

Theorem 6.1.3. *For direct products the complexity of the normal form problem is as follows.*

1. *Let $G = \prod_{\alpha\in\mathcal{L}} G_\alpha$ be a direct product of f. g. groups. Then*

$$\mathrm{nf}_G \in \mathsf{uAC}^0(\{\mathrm{nf}_\alpha \,|\alpha \in \mathcal{L}\}),$$

   that is there is an AC^0 Turing reduction of the normal form problem of G to the normal form problem of each G_α.

2. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then*

$$\mathrm{nf}_{\mathbf{\Pi}\mathcal{C}} \in \mathsf{uTC}^0(\mathrm{nf}_\mathcal{C}),$$

   that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$ and a word $w \in \Gamma^*$ computing a normal form of w in $G = \prod_{i=1}^n G_i$ is in uniform TC^0 with oracle gates for the uniform normal form problem of $\mathcal{C}$.

Proof. Let $w = w_1 \cdots w_n \in \Gamma^*$ be the factorization of the input word. First, for *case 1*, let nf_α be the normal form function of G_α. The normal form problem of G_α can be used to solve the word problem of G_α. Let $u \in \Gamma_\alpha^*$ for some $\alpha \in \mathcal{L}$, then $u =_{G_\alpha} 1$ if and only if $\mathrm{nf}_\alpha(u) = \mathrm{nf}_\alpha(1)$. Hence, by Theorem 6.1.1 a geodesic of w can be computed in AC^0 with oracle gates for nf_α. Let the geodesic of w be $\widehat{w} = \widehat{w}_1 \cdots \widehat{w}_{|\mathcal{L}|}$. We may assume $\mathcal{L} = \{1, \ldots, |\mathcal{L}|\}$. Then we have

$$\mathrm{nf}_G(\widehat{w}_i)_j = \bigvee_{\alpha\in\mathcal{L}} (\mathrm{alph}(\widehat{w}_i) = \alpha) \wedge \mathrm{nf}_\alpha(\widehat{w}_i)_j$$

where $\mathrm{nf}(w)_j$ denotes the j-th bit of $\mathrm{nf}(w)$. The normal form $\mathrm{nf}_G(w)$ is then

$$\mathrm{nf}_G(w) = \mathrm{nf}_G(\widehat{w}_1) \cdots \mathrm{nf}_G(\widehat{w}_{|\mathcal{L}|}),$$

This finishes *case 1*.

Now, *case 2.* is very similar. By the same reasoning as above, Theorem 6.1.2 can be used to compute a geodesic of w in TC^0 with oracle gates for the uniform normal form problem of $\mathcal{C}$. Let $\widehat{w} = \widehat{w}_1 \cdots \widehat{w}_n$ be the geodesic of w. Moreover, let $\mathrm{nf}_\mathcal{C}$ be the normal form function of $\mathcal{C}$. Then the normal form of a factor $\widehat{w}_i$ is given by

$$\mathrm{nf}_\mathcal{C}(\widehat{w}_i)_j = \bigvee_{k=1}^{n} (\mathrm{alph}(\widehat{w}_i) = \mathrm{bin}(i)) \wedge \mathrm{nf}_\mathcal{C}(G_k, \widehat{w}_i)_j$$

where again $\mathrm{nf}(w)_j$ denotes the j-th bit of $\mathrm{nf}(w)$. Now, the normal form function $\mathrm{nf}_{\mathbf{\Pi}\mathcal{C}}$ for the normal form problem of $\mathbf{\Pi}\mathcal{C}$ is given by

$$\mathrm{nf}_{\mathbf{\Pi}\mathcal{C}}(G_1, \ldots, G_n, w) = \mathrm{nf}_\mathcal{C}(\widehat{w}_1) \cdots \mathrm{nf}_\mathcal{C}(\widehat{w}_n).$$

This finishes *case 2.* □

6.2 in free products

For free groups the geodesic problem was recently solved by Armin Weiß in AC^0 with oracle gates for the word problem of the free group on two generators [Wei16]. However, his approach is for free groups only. We show that for any free product $G = *_{\alpha\in\mathcal{L}} G_\alpha$, the geodesic problem is AC^0 Turing reducible to the word problem of G. Interestingly, this is true for the uniform and non-uniform variant. As we will see in the next section, for graph products (in particular graph groups) the complexity of the uniform and non-uniform variant differs (unless $\mathsf{TC}^0 = \mathsf{NL}$).

Geodesics in free groups are unique, which already makes them a normal form. The same holds for geodesics in free products, except that additionally the normal form of each factor has to be computed.

For the geodesic problem we introduce the relation $\approx_w$ on a word $w \in \Gamma^*$ with factorization $w = w_1 \cdots w_n \in \Gamma^*$. Recall that $w_{ij} = w_i \cdots w_j$.

$$i \approx_w j \iff \begin{cases} w_{ij} \in G_\alpha, & \text{if } i \leq j \text{ and } \exists \alpha \in \mathcal{L} : w_i, w_j \in G_\alpha \setminus \{1\} \\ w_{ji} \in G_\alpha, & \text{if } i > j \text{ and } \exists \alpha \in \mathcal{L} : w_i, w_j \in G_\alpha \setminus \{1\} \\ i = j, & \text{if } w_i =_G 1 \end{cases}$$

Lemma 6.2.1. *The relation $\approx_w$ is an equivalence relation.*

Proof. By the very definition of $\approx_w$, it is reflexive and symmetric. Thus it only remains to show transitiveness of $\approx_w$. For this, let $i \approx_w j$ and $j \approx_w k$ with $w_i, w_j, w_k \in G_\alpha$. Suppose $i < j < k$. Then we do have

$$w_{ij} \in G_\alpha \qquad \textbf{and} \qquad w_{jk} \in G_\alpha.$$

Since $w_{ik} = w_i \cdots w_k = w_i \cdots w_j \cdots w_k = w_{ij} \cdot \overline{w}_j \cdot w_{jk} \in G_\alpha$, it follows $i \approx_w k$. Now, suppose $i < k < j$. Then we do have

$$w_{ij} \in G_\alpha \qquad \textbf{and} \qquad w_{kj} \in G_\alpha.$$

It is $w_{ik} = w_i \cdots w_k = w_{ij} \cdot \overline{w}_{kj} \cdot w_k$. Hence, $w_{ik} \in G_\alpha$ and thus $i \approx_w k$. All other cases for i, j and k follow by symmetry. □

Theorem 6.2.2. *Let $G = *_{\alpha\in\mathcal{L}} G_\alpha$ be a free product and $w = w_1 \cdots w_n \in \Gamma^*$ be any word. Moreover, let $\widehat{w} = \widehat{w}_1 \cdots \widehat{w}_n$ with*

$$\widehat{w}_i = \begin{cases} w_{k_1} \cdots w_{k_l}, & \text{if } i = k_1 < \ldots < k_l \text{ and } [i]_w = \{k_1, \ldots, k_l\} \\ 1, & \text{otherwise} \end{cases}.$$

Then we have $w =_G \widehat{w}$. Furthermore, the number of $\widehat{w}_j \neq_G 1$ is minimal. In this sense, $\widehat{w}$ is reduced.

Proof. We use the reduction process on Γ^* where uv with $u, v \in \Gamma_\alpha$ gets replaced by $[uv] \in \Gamma_\alpha$. This reduction process is confluent. Suppose w is reduced, then $w_i = \widehat{w}_i$ and we are done. Otherwise a reduction in w occurs. Let a reduction happen for $w_k w_{k+1}$. Since a reduction occurs only between words of the same group we must have $k \approx_w k{+}1$. We set $w'_i = w_i$ for $k \neq i \neq k{+}1$ and $w'_k = [w_k w_{k+1}]$ and $w'_{k+1} = 1$. We claim $\widehat{w}_i = \widehat{w}'_i$ for all $1 \leq i \leq n$. By construction we have $w =_G w'$ and hence, by induction $\widehat{w} = \widehat{w}'$ is reduced in the above sense. For the proof of the claim, let $i < k + 1 < j$. Then $w_i \cdots w_j = w'_i \cdots w'_j$. Hence we have $i \approx_w j$ if and only if $i \approx_{w'} j$. Only the equivalence class of $[k+1]_w$ changes to $[k+1]_{w'} = \{k+1\}$ and for the equivalence class of k it holds $[k]_{w'} = [k]_w \setminus \{k+1\}$. With $w'_k = w_k w_{k+1}$, $w'_{k+1} = 1$ and $\min[k]_w < k + 1$ we have $\widehat{w}_k = \widehat{w}'_k$ and $\widehat{w}_{k+1} = \widehat{w}'_{k+1}$. Thus $\widehat{w}_i = \widehat{w}'_i$ for all $1 \leq i \leq n$. □

For free products, determining a geodesic is AC^0 Turing reducible to the word problem in both the uniform and the non-uniform variant.

Theorem 6.2.3. *For free products computing a geodesic is in uniform* AC^0 *with oracle gates for the word problem.*

1. *Let* $G = *_{\alpha\in\mathcal{L}} G_\alpha$ *be a free product of f. g. groups. Then*

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

   that is there is an AC^0 *Turing reduction of the geodesic problem of* G *to the word problem of* G*.*

2. *Let* $\mathcal{C}$ *be a non-trivial class of f. g. groups. Then*

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

   that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$ *and a word* $w \in \Gamma^*$ *computing a geodesic of* w *in* $G = *_{i=1}^n G_i$ *is in uniform* AC^0 *with oracle gates for the uniform word problem of* $\mathbf{F}\mathcal{C}$*.*

Proof. We may assume that the free product has at least two factors. Otherwise, nothing has to be done. The crucial part is to determine the equivalence class $[i]_w$. For this, the containment of w_{ij} in G_α has to be decided. A word w_{ij} is contained in G_α if and only if $\pi_\alpha(w_{ij})\overline{w}_{ij} =_G 1$, where π_α is the projection of G onto G_α. Hence, we do need a circuit for this projection.

$$\pi_\alpha(w_k) = \begin{cases} w_k, & \text{if } \mathrm{alph}(w_k) = \alpha \\ 1, & \text{otherwise} \end{cases}$$

The α is not fixed, instead it depends on the factor w_i if $i \approx_w j$ is to be decided. Hence, the mapping we are interested in is

$$\pi_i(w_k) = \begin{cases} w_k, & \text{if } \operatorname{alph}(w_k) = \operatorname{alph}(w_i) \\ 1, & \text{otherwise} \end{cases}$$

This mapping can obviously be computed in AC^0 since $\operatorname{alph}(w_k)$ is part of the input. Now, for $i < j$ we set

$$Eq(i,j) = \begin{cases} 1, & \text{if } \overline{w_{ij}}\pi_i(w_{ij}) =_G 1 \textbf{ and } \operatorname{alph}(w_i) = \operatorname{alph}(w_j) \\ 0, & \text{otherwise.} \end{cases}$$

Next, to construct the output, let $\widehat{w}_i$ be defined as

$$\widehat{w}_i = \begin{cases} w_{k_1} \cdots w_{k_l}, & \text{if } i = k_1 < \ldots < k_l \text{ and } [i]_w = \{k_1, \ldots, k_n\} \\ 1, & \text{otherwise.} \end{cases}$$

For this assignment it has to be decided, if i is minimal in $[i]_w$. This check can be realized by the formula

$$Min(i) \equiv \forall j : (j \geq i) \vee \neg Eq(i,j)$$

Since, we demand the output to be geodesic, the $\widehat{w}_i$ with $\widehat{w}_i =_G 1$ have to be removed. Determining if the final position of $\widehat{w}_i$ is k can be determined in TC^0 via

$$pos(i,k) \equiv (k = \#j < i : \widehat{w}_j \neq_G 1) \wedge \widehat{w}_i \neq_G 1$$

Since the word problem of free products is uTC^0-hard, $pos(i,k)$ can be evaluated in uniform AC^0 with oracle gates for the word problem. Rearranging the $\widehat{w}_i$ and shifting the factors equaling the identity to the back can be done in AC^0 by using the predicate $pos(i,k)$.

Hence, for the rest of the proof we may assume that the $\widehat{w}_i$ are in the wanted order. The output gates for $\widehat{w}_i$ are realized as n blocks each having a size of n-bit. Since in the input every w_k is encoded by a block of n-bits this can be realized by

$$W_{i,k} = \begin{cases} w_k & \text{if } Min(i) \wedge Eq(i,k) = true \\ 0 & \text{otherwise} \end{cases}$$

Recall that the identity element is encoded by a block consisting only of zeros. Finally, the output of the circuit is

$$\operatorname{bin}(\alpha_1)W_{1,1} \cdots W_{1,n} \cdots \operatorname{bin}(\alpha_n)W_{n,1} \cdots W_{n,n},$$

where G_{α_i} is the group w_i belongs to. □

The normal form of some word is obtained from its geodesic, if each factor in some G_α gets replaced by its normal form in G_α. The proof how to obtain the normal form of each factor in AC^0 is similar to that of direct products and thus omitted. Therefore, as a direct consequence of the previous theorem we obtain

Theorem 6.2.4. *For free products computing a normal form is in uniform* AC^0 *with oracle gates for the normal form problem of each* G_α *and oracle gates for the word problem.*

1. *Let* $G = *_{\alpha\in\mathcal{L}} G_\alpha$ *be a free product of f. g. groups. Then*

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

   that is there is an AC^0 *Turing reduction of the normal form problem of* G *to the word problem of* G *and the normal form problem of each* G_α*.*

2. *Let* $\mathcal{C}$ *be a non-trivial class of f. g. groups. Then*

$$\mathrm{nf}_{\mathbf{F}\mathcal{C}} \in \mathsf{uAC}^0(\{\mathrm{nf}_\mathcal{C}, \mathrm{WP}(\mathbf{F}\mathcal{C})\}),$$

   that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$ *and a word* $w \in \Gamma^*$ *computing a normal form of* w *in* $G = *_{i=1}^n G_i$ *is in uniform* AC^0 *with oracle gates for the uniform word problem of* $\mathbf{F}\mathcal{C}$ *and the uniform normal form problem of* $\mathcal{C}$*.*

6.3 in graph products

Computing the length-lexicographic normal form in a graph product consists of two stages. First, a geodesic is computed and second, this geodesic is transformed into the length-lexicographic normal form. This section is divided into three parts. First, a parallel algorithm for finding a geodesic is described. This algorithm can be realized in AC^0 with oracle gates for the word problem for the uniform and non-uniform variant. The next two parts describe how this geodesic can be transformed into a lexicographic normal form. It turns out that the computations can be done very efficiently in parallel in the non-uniform case. The word problem in trace monoids is complete for uniform TC^0 [ÀG91]. This holds for both, the uniform and the non-uniform variant of the word problem. Any solution to a normal form yields a solution to the word problem. Hence, TC^0 is the best we can hope for in general. For the uniform case, the normal form problem turns out to be FNL-complete when the input word is geodesic. The hardness follows because the lexicographical first topological ordering problem of an (distinctively) labeled acyclic directed graph is NL-complete. Since NL is contained in NC^2, even in the uniform case, the geodesic and normal form problem can be solved efficiently in parallel.

Geodesic in AC^0

Finding a geodesic is in the uniform and non-uniform variant possible in AC^0 with access to oracle gates for the word problem. For this section, let the group $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and $w \in \Gamma^*$ be any word. The main obstacle is to determine, which factors in the word w need to be merged to obtain a geodesic word equivalent to w. For this, we define the relation $\approx_w$ on a word $w \in \Gamma^*$ with factorization $w = w_1 \cdots w_n \in \Gamma^*$. Recall that $w_{ij} = w_i \cdots w_j$.

$$i \approx_w j \iff \begin{cases} w_{ij} \in G_{I(\alpha)} \times G_\alpha, & \text{if } i \leq j \text{ and } \exists \alpha \in \mathcal{L} : w_i, w_j \in G_\alpha \setminus \{1\} \\ w_{ji} \in G_{I(\alpha)} \times G_\alpha, & \text{if } i > j \text{ and } \exists \alpha \in \mathcal{L} : w_i, w_j \in G_\alpha \setminus \{1\} \\ i = j, & \text{if } w_i =_G 1 \end{cases}$$

In case of the graph product being actually a free product, the relation $\approx_w$ is identical to the relation defined for free products. The relation $\approx_w$ shares the property of being an equivalence relation and the proof is essentially the same as for free products with some small adjustments.

Lemma 6.3.1. *The relation $\approx_w$ is an equivalence relation.*

Proof. By the very definition of $\approx_w$, it is reflexive and symmetric. Thus it only remains to show transitiveness of $\approx_w$. For this, let $i \approx_w j$ and $j \approx_w k$ with $w_i, w_j, w_k \in G_\alpha$. Suppose $i < j < k$. Then we do have

$$w_{ij} \in G_{I(\alpha)} \times G_\alpha \qquad \textbf{and} \qquad w_{jk} \in G_{I(\alpha)} \times G_\alpha.$$

Since $w_{ik} = w_i \cdots w_k = w_i \cdots w_j \cdots w_k = w_{ij} \cdot \overline{w}_j \cdot w_{jk} \in G_{I(\alpha)} \times G_\alpha$, it follows $i \approx_w k$. Now, suppose $i < k < j$. Then we do have

$$w_{ij} \in G_{I(\alpha)} \times G_\alpha \qquad \textbf{and} \qquad w_{kj} \in G_{I(\alpha)} \times G_\alpha.$$

It is $w_{ik} = w_i \cdots w_k = w_{ij} \cdot \overline{w}_{kj} \cdot w_k$. Hence, $w_{ik} \in G_{I(\alpha)} \times G_\alpha$ and thus $i \approx_w k$. All other cases for i, j and k follow by symmetry. □

Theorem 6.3.2. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and $w \in \Gamma^*$ be any word with factorization $w = w_1 \cdots w_n$. Moreover, let $\widehat{w} = \widehat{w}_1 \cdots \widehat{w}_n$ with*

$$\widehat{w}_i = \begin{cases} w_{k_1} \cdots w_{k_l}, & \text{if } i = k_1 < \ldots < k_l \text{ and } [i]_w = \{k_1, \ldots, k_l\} \\ 1, & \text{otherwise} \end{cases}.$$

Then we have $w =_G \widehat{w}$. Furthermore, the number of $\widehat{w}_j \neq_G 1$ is minimal. In this sense, $\widehat{w}$ is reduced.

Proof. For the proof, we proceed by induction on the number of nodes in the dependence graph $D(w)$ of w. Let the nodes of $D(w)$ be v_i with $\lambda(v_i) = w_i$. For simplification, we may assume $w_i \neq_G 1$ for all i. Otherwise, if $w_i =_G 1$ we set $[i] = \{i\}$ and do not include it in the dependence graph.

First, let the dependence graph $D(w)$ be reduced. Suppose, there are indices i, j with $i \approx_w j$. We may assume $|i - j|$ to be minimal. Let $w_i, w_j \in G_\alpha$, then we must have $w_{ij} \in G_{I(\alpha)} \times G_\alpha$. Since $|i - j|$ is minimal we further must have $w_{i+1,j-1} \in G_{I(\alpha)}$. But then the edge (v_i, v_j) does exist in the Hasse diagram of $D(w)$ and therefore $D(w)$ is not reduced. A contradiction. Thus, we have $[i] = \{i\}$ for all i and we are done in case of the dependence being reduced.

Now, suppose a reduction is possible in the dependence graph. The reduction process on the dependence graph is confluent, hence we can pick any two nodes that allow a reduction. Let i, j be the maximal indices for which a reduction is possible. Let v_i, v_j be nodes in $D(w)$ with $a = \lambda(v_i), a' = \lambda(v_j) \in G_\alpha$ and there is an edge from v_i to v_j in the Hasse diagram of $D(w)$. Then we remove the node v_j and replace the label of v_i by $[a \cdot a']$. If $[a \cdot a'] =_G 1$, then we remove the node v_i either. This gives us a new dependency graph D' of the word $w' = w'_1 \cdots w'_n$. Thereby we have $w'_t = w_t$ for $t \neq i, j$. Moreover $w'_i = [a \cdot a']$ and $w'_j = 1$. We claim $\widehat{w}'_i = \widehat{w}_i$. Observe that $w'_i \cdots w'_j =_G w_i \cdots w_j$. Since $w =_G w'$ the result follows by induction on $w' = \widehat{w}'_1 \cdots \widehat{w}'_n$. To proof the claim. Let $i \neq k < l \neq j$ and $k \approx_w l$ with $w_k, w_l \in G_\beta$. If $w_k \cdots w_l =_G w'_k \cdots w'_l$, then $k \approx_{w'} l$. Since we did reduce node i with node j, all nodes v_t with $i < t < j$ in between must be independent from α. Thus the two remaining cases $k < i < l < j$ and $i < k < j < l$ follow. Contrary, by the same arguments if $k \approx_{w'} l$, then we have $k \approx_w l$ for $i \neq k < l \neq j$. It follows $[k]_w = [k]_{w'}$ and therefore $\widehat{w}_k = \widehat{w}'_k$ for $k \not\approx_w i, j$.

Now, let $k \approx_w i$ with $k \neq j$. Since we did reduce the node v_i with the node v_j there is no node v_t in between with $w_t \in G_\alpha$. Thus, we have $k < i$ or $k > j$. For $k < i$ we have $w_k \cdots w_i =_G w'_k \cdots w'_i \cdot a'^{-1}$. In the case $i < k$, we have $w_i \cdots w_k =_G w'_i \cdots w'_k$. Thus for $w'_i \neq_G 1$ we have $[i]_{w'} = [i]_w \setminus \{j\}$ and $[j] = \{j\}$. Since $i < j$ it follows that $\widehat{w}_i = \widehat{w}'_i$ and $\widehat{w}_j = \widehat{w}'_j$. If $w'_i =_G 1$, we have $[i]_{w'} = \{i\}$. As shown above, we have $i \neq k \approx_w l \neq j$ if, and only if $i \neq k \approx_{w'} l \neq j$. Hence, for $i \approx_w k$ we have $[k]_{w'} = [k]_w \setminus \{i, j\}$. Suppose $k > j$. Since j was chosen maximal the word $w_j \cdots w_k$ must be reduced, but then $j \not\approx_w k$. A contradiction. Therefore, we must have $k < i$. Hence, since $k < i < j$ we have $\widehat{w}_i = 1$ and $\widehat{w}_j = 1$ either way. Moreover, $\widehat{w}'_k = \widehat{w}_k$. □

Theorem 6.3.3. *For graph products computing a geodesic is in uniform* AC^0 *with oracle gates for the word problem.*

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

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

that is there is an AC^0 *Turing reduction of the geodesic problem of* G *to the word problem of* G*.*

2. *Let* $\mathcal{C}$ *be a non-trivial class of f. g. groups. Then*

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

that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$*, an independence relation* I *and a word* $w \in \Gamma^*$ *computing a geodesic of* w *in* $G = \mathrm{GP}\,(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ *for* $\mathcal{L} = \{1, \ldots, n\}$ *is in uniform* AC^0 *with oracle gates for the uniform word problem of* $\mathbf{GP}\mathcal{C}$*.*

Proof. For the proof, if G is a fixed graph product, we may assume it contains a free product. Otherwise, G is a direct product and the result follows by Theorem 6.1.1.

Let $w = w_1 \cdots w_n \in \Gamma^*$ be a factorization of the input word. In a first step, the equivalence relation $\approx_w$ is computed. For this, let π_α be the projection of Γ^* to $\left(\Gamma_{I(\alpha)} \cup \Gamma_\alpha\right)^*$. For any word $u \in \Gamma^*$ the question $u \in G_{I(\alpha)} \times G_\alpha$ is equivalent to $\overline{u}\pi_\alpha(u) =_G 1$ by Lemma 5.6.3. Thus, the oracle gates for the word problem of G can be used to decide the equivalence relation $\approx_w$. The α corresponding to w_i is $\alpha = \mathrm{alph}(w_i)$. Therefore, for π_α we use

$$\begin{aligned}\pi_i(w) &= \begin{cases} w, & \text{if } \mathrm{alph}(w) \in I(\mathrm{alph}(w_i)) \\ w, & \text{if } \mathrm{alph}(w) = \mathrm{alph}(w_i) \\ 1, & \text{otherwise} \end{cases} \\ &= \begin{cases} w, & \text{if } (\mathrm{alph}(w_i), \mathrm{alph}(w)) \in I \\ w, & \text{if } \mathrm{alph}(w) = \mathrm{alph}(w_i) \\ 1, & \text{otherwise} \end{cases}\end{aligned}$$

The projection π_i can be realized in AC^0, even for the uniform variant. The independence relation I is given as an $n \times n$ matrix M_I. We denote the (i, j)-th entry of M_I by $M_I(i, j)$. Then we have

$$(\alpha, \beta) \in D \equiv \bigvee_{i=1}^{n} \bigvee_{j=1}^{n} \alpha = \mathrm{bin}(i) \wedge \beta = \mathrm{bin}(j) \wedge M_I(i, j)$$

Now, for $i < j$ we set

$$Eq(i, j) = \begin{cases} 1 & \text{if } \overline{w_{ij}}\pi_i(w_{ij}) =_G 1 \textbf{ and } \mathrm{alph}(w_i) = \mathrm{alph}(w_j) \\ 0 & \text{otherwise} \end{cases}$$

Next, the words $\widehat{w}_i$ are constructed with

$$\widehat{w}_i = \begin{cases} w_{k_1} \cdots w_{k_l}, & \text{if } i = k_1 < \ldots < k_l \text{ and } [i]_w = \{k_1, \ldots, k_l\} \\ 1, & \text{otherwise} \end{cases}.$$

For this construction, i must equal the minimum of $[i]_w$. This is realized by the formula

$$Min(i) \equiv \forall j : (j \geq i \vee \neg Eq(i,j)).$$

Since, we demand the output to be geodesic, the $\widehat{w}_i$ with $\widehat{w}_i =_G 1$ have to be removed. Determining if the final position of $\widehat{w}_i$ is k can be determined in TC^0 via

$$pos(i,k) \equiv (k = \#j < i : \widehat{w}_j \neq_G 1) \wedge \widehat{w}_i \neq_G 1$$

Now, we have two cases. If G is a fixed graph product, then G contains a free product by the above assumption and hence, its word problem is uTC^0-hard. Otherwise, we are in the uniform case. But then, the word problem is NL-hard and therefore also uTC^0-hard. Either way, $pos(i,k)$ can be evaluated in uniform AC^0 with oracle gates for the word problem. Rearranging the $\widehat{w}_i$ and shifting the factors equaling the identity to the back can be done in AC^0 by using the predicate $pos(i,k)$.

Hence, for the rest of the proof we may assume that the $\widehat{w}_i$ are in the wanted order. The output gates for $\widehat{w}_i$ are realized as n blocks each having a size of n-bit. Since in the input every w_k is encoded by a block of n-bits this can be realized by

$$W_{i,k} = \begin{cases} w_k & \text{if } Min(i) \wedge Eq(i,k) = true \\ 0 & \text{otherwise} \end{cases}$$

Recall that the identity element is encoded by a block consisting only of zeros. Finally, the output of the circuit is

$$\mathrm{bin}(\alpha_1) W_{1,1} \cdots W_{1,n} \cdots \mathrm{bin}(\alpha_n) W_{n,1} \cdots W_{n,n},$$

where G_{α_i} is the group w_i belongs to. □

Length-lexicographic normal form

The complexity for finding a lexicographic normal form in graph products is different for the uniform and non-uniform variant (unless $\mathsf{TC}^0 = \mathsf{NL}$). However, the underlying technique remains the same. The main reason for this difference in complexity is due to finding paths in the dependence graph. The length of some

path in the dependence graph is bounded by the size of the graph product. This bound is a constant, when the graph product is fixed. However, as soon as the graph product is part of the input, it resembles to finding any path in some graph.

We first give a recursive formula for the lexicographical topological ordering $<_{lf}$ on the dependence graph. It does hold in general for any directed acyclic graph, for details see [Sho89]. However, we restrict the lemma to dependence graphs as they occur in graph products to make the proof more easy to read. This proof follows essentially the ideas in [Ebi95], where Ebinger constructed such a formula for trace monoids. By slightly adjusting this formula, it can be used to compute the length-lexicographic normal form in graph products.

The following lemma is useful for the proof of the recursive formula and was shown for trace monoids by [Die90, AK79]. The lemma for trace monoids transfers directly to geodesics in graph products.

Lemma 6.3.4. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and $<_\mathcal{L}$ be a linear order on $\mathcal{L}$. Then, a geodesic word $w \in \Gamma^*$ is in lexicographic normal form if and only if for each factor aub of w with $a \in G_\alpha, b \in G_\beta, u \in \Gamma^*$ and $(au, b) \in I$ it holds $\alpha <_\mathcal{L} \beta$.*

Proof. Let $w \in \Gamma^*$ be in lexicographic normal form and aub be a factor of w with $a \in G_\alpha, b \in G_\beta, u \in \Gamma^*$ and $(au, b) \in I$. Then $aub =_G bau$ and thus, we must have $\alpha <_\mathcal{L} \beta$.

For the contrary, we show that for any geodesic $w' \in \Gamma^*$ with $w =_G w'$ we have $w <_{lf} w'$. Then, we have that w is in lexicographic normal form. Let $w = w_1 a w_2$ and $w' = w_1 b w_2'$ with $w_1, w_2, w_2' \in \Gamma^*$, $a \in G_\alpha$ and $b \in G_\beta$. Since $w =_G w'$ we must have $aw_2 = bw_2'$ and thus $w_2' = \bar{b}aw_2$. The words w_2 and w_2' are geodesic, but $\bar{b}aw_2$ is not. Hence a reduction must occur and thus, there exists $u, v \in \Gamma^*$ and $b' \in G_\beta$ with $\bar{b}aw_2 = \bar{b}aub'v$. For a reduction to happen, we must have $(au, b) \in I$. The word aub' is a factor of w and therefore $\alpha <_\mathcal{L} \beta$. Finally, this yields $w <_{lf} w'$. □

Lemma 6.3.5. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and let $w \in \Gamma^*$ be a reduced word. Then we have $u <_{lf} v$ in the length-lexicographic topological ordering if and only if either*

1. *there is some path $u \xrightarrow{*} v$ in the dependence graph $D(w)$, or*
2. *it exists some $x \in V$ with $\mathrm{alph}(u) <_\mathcal{L} \mathrm{alph}(x)$, there is some path $x \xrightarrow{*} v$ in $D(w)$ and $u <_{lf} x$.*

Proof. For the if-part. Suppose there is some path from u to v in $D(w)$, then $u < v$ in any topological ordering. Otherwise case 2. in the lemma holds. Since there is some path from x to v in $D(w)$, we have $x <_{lf} v$. The order $<_{lf}$ is a total order and together with $u <_{lf} x$ the desired $u <_{lf} v$ follows.

The proof for the only-if part of the lemma is more involved. Suppose $u <_{lf} v$ and there is no path $u \xrightarrow{*} v$. Let $x_0 \cdots x_n$ be the corresponding factor between u and v in the lexicographic normal form of w with $u = x_0$ and $v = x_n$. Let $x := x_i$ with $1 \leq i \leq n$ being minimal such that there is some path $x \xrightarrow{*} v$ in $D(w)$. Then we have $(x_0 \cdots x_{i-1}, x_i) \in I$ and thus, by Lemma 6.3.4 $\operatorname{alph}(u) <_{\mathcal{L}} \operatorname{alph}(x)$, since $x_0 \cdots x_n$ is in lexicographic normal form. Moreover, by the choice of x we have $u <_{lf} x$. □

One has to be careful with the lemma. If $\operatorname{alph}(u) < \operatorname{alph}(v)$, then *case 2.* of the lemma is trivially fulfilled and is useless to determine the lexicographical ordering of the dependence graph. However, the following corollary is immediate and does overcome this problem.

Corollary 6.3.6. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ be a graph product and let $w \in \Gamma^*$ be a reduced word. Then we have $u <_{lf} v$ in the length-lexicographic topological ordering if and only if either*

1. *there is some path $u \xrightarrow{*} v$ in the dependence graph $D(w)$, or*
2. *it exists some $v \neq x \in V$ with $\operatorname{alph}(u) <_{\mathcal{L}} \operatorname{alph}(x)$, there is some path $x \xrightarrow{*} v$ in $D(w)$ and $u <_{lf} x$, or*
3. *we have $\operatorname{alph}(u) <_{\mathcal{L}} \operatorname{alph}(v)$ and $\neg(v <_{lf} u)$.*

Now, *case 3.* of the corollary ensures that we do not end in an endless recursion in order to determine the lexicographical topological ordering.

6.3.1 The non-uniform case

The complexity of the non-uniform variant of the length-lexicographic normal form problem is mainly determined by the complexity of the word problem. All the reductions in the following can be transformed into uniform AC^0 Turing reductions. How to determine the lexicographic ordering was - in some sense - known before due to [Ebi95]. His first order formula translates under minor additions into an AC^0-circuit.

The recursive application of Corollary 6.3.6 does stop after at most $2|\mathcal{L}|$ steps, since in case 2. of the lemma the order of the involved nodes according to $<_{\mathcal{L}}$ does increase and case 3. cannot bounce. See also Figure 6.1 on page 89. Since the size of the formula for $lex(u, v)$ is bounded, it can be realized in AC^0. However, for the computation of the length-lexicographic normal form, the power of AC^0 does not suffice, as we will see. Nonetheless, given a reduced word $w \in \Gamma^*$ its lexicographic normal form can be computed in TC^0.

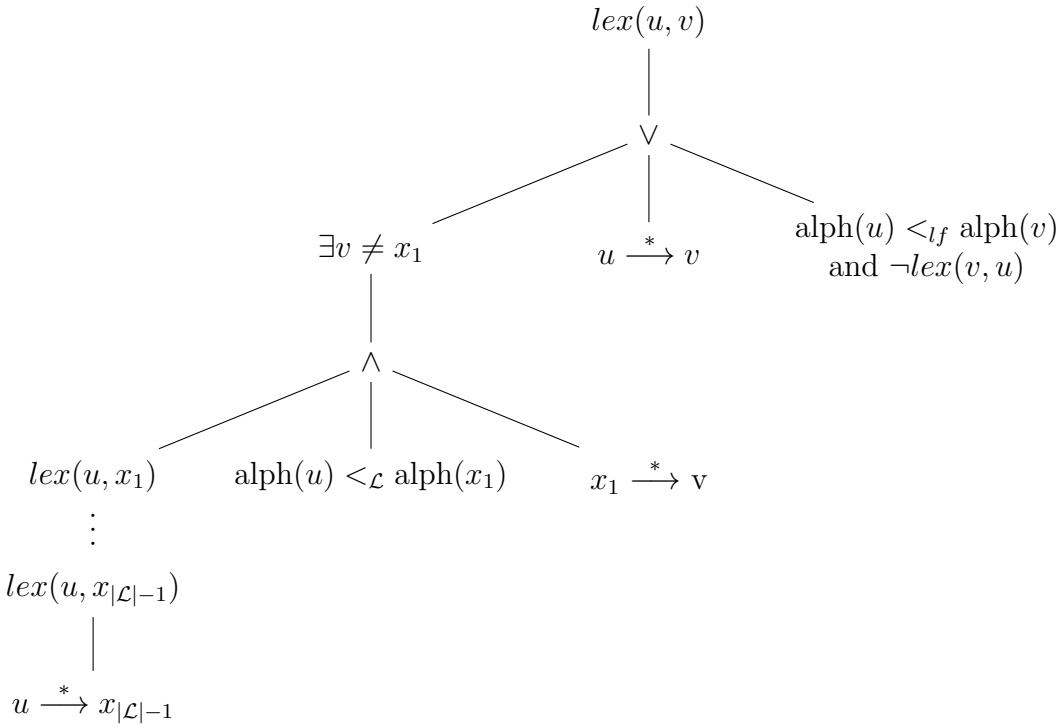


Figure 6.1: The recursive formula for $lex(u, v)$. The height of the recursion tree is bounded by a multiple of $|\mathcal{L}|$.

Theorem 6.3.7. *Let $G = \text{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f. g. groups and $w \in \Gamma^*$ be a reduced word with factorization $w = w_1 \cdots w_n \in \Gamma^*$. Then a lexicographic ordering of w can be computed in uniform TC^0.*

Proof. In the above remarks using Corollary 6.3.6, it was shown that $lex(u, v)$ can be decided in AC^0 by only using gates for the order $<_{\mathcal{L}}$ and testing on paths. First, $\text{alph}(u) <_{\mathcal{L}} \text{alph}(v)$ is realized by

$$\text{alph}(u) <_{\mathcal{L}} \text{alph}(v) \equiv \bigvee_{\alpha<_{\mathcal{L}}\beta} \text{alph}(u) = \alpha \wedge \text{alph}(v) = \beta$$

The gate for testing on a path between two nodes is more involved. Since we do have dependence graphs, the length of any path is bounded by $|\mathcal{L}|$. Let $k = |\mathcal{L}|$, then $u \xrightarrow{*} v$ can be expressed by

$$\exists\, i_1 \leq \ldots \leq i_k : u = w_{i_1} \wedge v = w_{i_k} \wedge (w_{i_1}, w_{i_2}) \notin I \wedge \ldots \wedge (w_{i_{k-1}}, w_{i_k}) \notin I.$$

Since $<_{lf}$ is a total order, the position of some w_i in this order equals exactly the number of w_j which are lexicographically smaller. This number can be determined

in TC^0, hence for the position of w_i we have

$$pos(i) \equiv \ \#j : lex(w_j, w_i)$$

Finally, the lexicographic ordering representing w is given by

$$(\widehat{w}_i)_j \equiv \bigvee_{1 \leq k \leq n} pos(k) = bin(i) \wedge (w_k)_j,$$

where $(w)_j$ denotes the j-th bit of the binary representation of w. □

The upper complexity bound of TC^0 in Theorem 6.3.7 is tight, as there is some graph product for which the normal form problem is uTC^0-complete.

Theorem 6.3.8. *There is some graph product G, for which on input of a reduced word computing the lexicographic ordering is uTC^0-complete. More precisely, computing the lexicographic ordering is uTC^0-complete if the graph product G contains at least two free products.*

Proof. Consider the graph product $\mathrm{G} = (\langle a \rangle * \langle b \rangle) \times (\langle c \rangle * \langle d \rangle)$ with $a < b < c < d$. We reduce the uTC^0-complete problem MAJORITY to the normal form problem. Let the sequence of input bits be $x = x_1 \cdots x_n$. The input is mapped to an element of G via the homomorphism

$$\varphi : \mathbb{B} \to \{a, b, c, d\}^* : x_i \mapsto \begin{cases} ab & \text{if } x_i = 1 \\ cd & \text{if } x_i = 0. \end{cases}$$

The normal form of $\varphi(x) = \varphi(x_1 \cdots x_n)$ is $(ab)^k (cd)^{n-k}$, where k is the number of ones in the input sequence x. The computation of φ can be done in AC^0. Therefore, we obtain

$$\textsc{Majority}(x) \equiv \bigvee_{n/2 \leq k \leq n} \Big(\mathrm{nf}_G(\varphi(x)) = (ab)^k (cd)^{n-k} \Big)$$

□

The missing case is the computation of the length-lexicographic normal form of an arbitrary (non-reduced) input word. The next theorem might seem contradicting to the above theorem on TC^0-completeness. The length-lexicographic normal form is AC^0-computable with oracle gates for the word problem. As we will see, all the necessary TC^0 computations can be encapsulated in the oracle gate for the word problem.

Theorem 6.3.9. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f. g. groups. Then*

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

that is on input of a word its normal form computation can be reduced to the normal form problem of each G_α and the word problem of G by an AC^0 Turing reduction.

Proof. We distinguish two cases. If G is a direct product, then the result follows by case 1 of Theorem 6.1.3.

Otherwise the graph product contains a free product. Therefore, its word problem is uTC^0-hard and hence, any TC^0-computation can be reduced to the word problem of G by a uniform AC^0-circuit. By Theorem 6.3.2 computing a geodesic is AC^0 Turing reducible to the word problem of G. Computing the lexicographic ordering of the geodesic can then be done in TC^0 by Theorem 6.3.7. The only step remaining is to compute the normal form of each factor, which can be achieved by

$$\mathrm{nf}_G(w_i)_j \equiv \bigvee_{\alpha\in\mathcal{L}} (\mathrm{alph}(w_i) = \alpha) \wedge \mathrm{nf}_\alpha(w_i)_j,$$

where $\mathrm{nf}(w)_j$ denotes the j-th bit of $\mathrm{nf}(w)$. □

6.3.2 The uniform case

The complexity of computing the length lexicographic normal form is more involved in the uniform case. We will see that computing the lexicographic normal form of some geodesic input word is complete in FNL. The following theorem by Shoudai on directed graphs serves as the basis for computing the lexicographic normal form in FNL. For showing the completeness some more work has to be done.

Theorem 6.3.10 (Shoudai [Sho89])**.** *The lexicographically first topological order problem* (LexTop) *is NL-complete. This problem is as follows. Given a graph $G = (V, E)$ with a linear order on V and two nodes u, v. Question: Is $u <_{lf} v$? Hereby $<_{lf}$ denotes the lexicographical topological ordering.*

The lexicographic normal form is a lexicographic topological ordering of the dependence graph $D(w)$. Computing the lexicographic topological ordering of a graph with distinct labels is NL-complete. However, we do have the additional information that the word w itself is a topological ordering of the dependence graph. It turns out that this additional information does not help to compute the lexicographically first topological ordering. The proof of the NL-hardness of computing the lexicographic normal form proceeds by first reducing Gap to TopGap and

second reducing TopGap to LexOrder. The mentioned problems are defined formally in the following. First, the problem Gap is as follows.

Problem:	Gap
Input:	A directed graph $G = (V, E)$ and nodes $s, t \in V$.
Question:	Is there a path from s to t in G?

The problem TopGap is an extension of Gap with a given topological order.

Problem:	TopGap
Input:	A directed graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and $(v_i, v_j) \in E \implies i < j$ and nodes $s, t \in V$.
Question:	Is there a path from s to t in G?

And finally, the problem LexOrder is to determine a lexicographic ordering of some word in a trace Monoid M.

Problem:	LexOrder
Input:	A trace monoid $M = M(\Sigma, I)$ with an ordering on Σ, a word $w = a_1 \cdots a_n$ with $a_i \in \Sigma$ for all $1 \leq i \leq n$ and integers i, j with $1 \leq i, j \leq n$.
Question:	Does the i-th letter a_i occur before the j-th letter a_j in the lexicographic normal form of w.

Theorem 6.3.11. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Let the input be groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I and a geodesic $w = a_1 \cdots a_n$. Then deciding if a_i has to be placed before a_j in the lexicographic normal form of $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ for $\mathcal{L} = \{1, \ldots, n\}$ is* NL*-hard.*

Proof. We show the following sequence of reductions.

$$\textsc{Gap} \leq_{\mathrm{cd}} \textsc{TopGap} \leq_{\mathrm{cd}} \textsc{LexOrder}$$

Since Gap is NL-complete the result follows. Moreover, since the input word is geodesic, it suffices to show the hardness for trace monoids. The original proof for $(a) \implies (b)$ goes back to [Jon75]. Let $G = (V, E)$ be a directed graph with $|V| = n$. We construct a new graph $G' = (V', E')$ with $V' = V \times [n]$, where $[n]$ denotes the set $\{1, \ldots, n\}$. The new nodes are ordered in levels 1 to n and edges exist only between neighbored levels. More precisely, for the set of edges E' we have for all $1 \leq i < n$:

$$((u, i), (v, i+1)) \in E' \iff u = v \textbf{ or } (u, v) \in E.$$

Ordering the nodes (v, i) by their level i gives a topological ordering of the graph G'. There is a path from s to t in G if and only if there is a path from $(s, 1)$ to (t, n) in G'.

$(b) \implies (c)$. Let $G = (V, E)$ be a graph with $V = \{v_1, \ldots, v_n\}$ and $v_1, \ldots, v_n$ being a topological order of the vertices of G. We construct the trace monoid M as follows. Let $\Sigma = V$ and $I = E \cup \overline{E}$, where $(u, v) \in E \iff (v, u) \in \overline{E}$. Let $w = v_1 \cdots v_n$. The dependence graph of w is G. For the ordering we make s maximal in $<_\Sigma$. If there is a path $s \xrightarrow{*} t$ in G, then s is placed before t in any topological order. Hence $s <_{lf} t$. Contrary, if $s <_{lf} t$, then Corollary 6.3.6 implies the existence of a path $s \xrightarrow{*} t$ since s is maximal according to $<_\Sigma$.

Finally, we can finish the proof as follows. There is some non-trivial group $H \in \mathcal{C}$ since $\mathcal{C}$ is non-trivial. Now, $\mathbb{Z}$ embeds into $H * H$. Thus, M embeds into some graph product over $\mathcal{C}$ and the result follows.

□

Remark 6.3.12. The proof idea of $(b) \implies (c)$ is similar to the proof that LexTop is NL-hard by Shoudai [Sho89].

Theorem 6.3.13. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Let the input be groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I and a geodesic $w = a_1 \cdots a_n$. Then deciding if a_i has to be placed before a_j in the lexicographic normal form of $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ for $\mathcal{L} = \{1, \ldots, n\}$ is NL-complete.*

Proof. Hardness is shown in Theorem 6.3.11. For containment, we apply Theorem 6.3.10 on the dependence graph $D(w)$. The linear order on the nodes $a_1, \ldots, a_n$ is given as follows.

$$a_i < a_j \iff \mathrm{alph}(a_i) < \mathrm{alph}(a_j) \textbf{ or } (\mathrm{alph}(a_i) = \mathrm{alph}(a_j) \text{ and } i < j)$$

□

Corollary 6.3.14. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Let the input be groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I and a geodesic $w = w_1 \cdots w_n$. Then computing the lexicographic normal form of w in $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ for $\mathcal{L} = \{1, \ldots, n\}$ is FNL-hard and contained in $\mathsf{FNL}^{\mathrm{nf}c}$.*

Proof. Let $w = w_1 \cdots w_n$ be a factorization of the input word. Moreover, let $<_{lf}$ be the lexicographic ordering of the w_i. Since $<_{lf}$ is a total order, the position of w_i in the lexicographic normal form is determined by the number of w_j with $w_j <_{lf} w_i$. Let $\pi(i)$ denote this number and $\widehat{w}_k = w_i$ if $k = \pi(i) + 1$. Then the lexicographic normal form of w is $\widehat{w} = \widehat{w}_1 \cdots \widehat{w}_n$. Since only counting is involved, this can be achieved in logarithmic space with an oracle for the decision of $w_i <_{lf} w_j$. To

output the normal form of w_i the oracle for the normal form function $\mathrm{nf}_{\mathcal{C}}$ can be applied. Since $\mathsf{FL}^{\mathsf{NL}}$ equals FNL, containment in $\mathsf{FNL}^{\mathrm{nf}_{\mathcal{C}}}$ follows.

For the hardness, any NL-complete problem is FNL-complete, since $\mathsf{FNL} = \mathsf{NC}^1(\mathsf{NL})$ and completeness in FNL is due to NC^1 Turing reductions. Thus, it suffices to reduce GAP to the lexicographic normal form function. Let $H = (V, E)$ be the input graph with vertices $s, t \in V$. Moreover, let M be the trace monoid and w be the word as constructed in the proof of Theorem 6.3.11. There is an $s \xrightarrow{*} t$ path in H if and only if $s <_{lf} t$ in the constructed word w. Hence, there is an $s \xrightarrow{*} t$ path in H if and only if s is placed to the left of t in the normal form of w. Finally, the result follows since M embeds into some graph product over $\mathcal{C}$. □

The remaining part for the normal form is to compute the normal form for an arbitrary (non-reduced) input word.

Theorem 6.3.15. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then*

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

that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I and a word $w \in \Gamma^*$ computing the length lexicographic normal form of w in the group $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ for $\mathcal{L} = \{1, \ldots, n\}$ is in uniform AC^0 with oracle gates for $\mathsf{C}_=\mathsf{L}^{\mathrm{nf}_{\mathcal{C}}}$.

Proof. By Theorem 5.6.14 the word problem is solvable in $\mathsf{C}_=\mathsf{L}^{\mathrm{WP}(\mathcal{C})}$. The solution of the normal form can be used for the word problem since $w =_G 1$ if and only if $\mathrm{nf}_{\mathcal{C}}(w, G) = \mathrm{nf}_{\mathcal{C}}(1, G)$. Hence, by Theorem 6.3.3 case 2 a geodesic can be computed in $\mathsf{AC}^0(\mathsf{C}_=\mathsf{L}^{\mathrm{nf}_{\mathcal{C}}})$. Finally, since $\mathsf{NL} \subseteq \mathsf{C}_=\mathsf{L}$ the length-lexicographic normal form can be computed in $\mathsf{AC}^0(\mathsf{C}_=\mathsf{L}^{\mathrm{nf}_{\mathcal{C}}})$. □

Corollary 6.3.16. *The uniform length-lexicographic normal form problem of right angled Coxeter groups and graph groups is contained in $\mathsf{uAC}^0(\mathsf{C}_=\mathsf{L})$.*

Proof. The result follows with Theorem 6.3.15 and $\mathsf{C}_=\mathsf{L}^{\mathsf{LOGSPACE}} = \mathsf{C}_=\mathsf{L}$. □

Example 6.3.17. *The uniform normal form problem of graph products over finite groups is contained in $\mathsf{uAC}^0(\mathsf{C}_=\mathsf{L})$, when the finite groups are given by their multiplication table.*

Proof. Let G be a finite group and $a_1, \ldots, a_n \in G$. Computing the group element $[a_1 \cdots a_n]$ is FL-complete if the group is given by its multiplication table [CM87]. We let $\mathrm{nf}_G(a_1 \cdots a_n) = [a_1 \cdots a_n]$ be the normal form function of G. Together with Theorem 6.3.15 we obtain that the uniform normal form problem of graph products over finite groups is in $\mathsf{uAC}^0(\mathsf{C}_=\mathsf{L}^{\mathsf{LOGSPACE}})$. Finally, with $\mathsf{C}_=\mathsf{L}^{\mathsf{LOGSPACE}} = \mathsf{C}_=\mathsf{L}$ the result follows. □

Chapter 7

The conjugacy problem

The conjugacy problem $u \sim_G v$ is the question, whether there is some solution for the equation $uz =_G zv$. As for the word problem, the conjugacy problem does not depend on the generating alphabet since any automorphism between groups preserves conjugacy. In this chapter we consider groups with a solution to the conjugacy problem and investigate the parallel complexity of the conjugacy problem in products of these groups. The conjugacy problem in direct products is rather simple. For free groups it is long known that the conjugacy problem can be solved in linear time on a RAM. A first solution of the conjugacy problem in graph groups was given by Servatius [Ser89]. The linear time solution of free groups was generalized by Liu, Wrathall and Zeger to trace monoids and graph groups [LWZ90]. Another approach for a linear time solution was given by Crisp, Godelle and Wiest [CGW09]. Unfortunately, both solutions contain highly sequential algorithms. For solving the conjugacy problem in graph products the basic observation is that for cyclically reduced words the conjugacy problem behaves very much the same as for traces. We will show this and how to obtain parallel solutions in the following sections. Parts of these results are from [DK14, DK16], where the conjugacy problem of a fixed graph product is considered in the complexity class LOGSPACE.

7.1 in direct products

For direct products the conjugacy problem directly translates to the conjugacy problem of the base groups. We state the result for completeness. Moreover, it will be used for the special case of a graph product actually being a direct product.

Theorem 7.1.1. *For direct products the conjugacy problem is in uniform* AC^0 *with oracle gates for the conjugacy problem of the base groups.*

1. *Let* $G = \prod_{\alpha \in \mathcal{L}} G_\alpha$ *be a direct product of f. g. groups. Then*

$$\mathrm{CP}(G) \in \mathsf{uAC}^0(\{\mathrm{CP}(G_\alpha) \mid \alpha \in \mathcal{L}\}),$$

that is there is an AC^0 *Turing reduction of the conjugacy problem of* G *to the conjugacy problem of each* G_α

2. *Let* $\mathcal{C}$ *be a non-trivial class of f. g. groups. Then*

$$\mathrm{CP}(\mathbf{\Pi}\mathcal{C}) \in \mathsf{uAC}^0(\mathrm{CP}(\mathcal{C})),$$

that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$ *and words* $u, v \in \Gamma^*$ *the conjugacy problem* $u \sim v$ *in* $G = \prod_{i=1}^n G_i$ *can be solved in uniform* AC^0 *with oracle gates for the conjugacy problem of* $\mathcal{C}$*.*

Proof. Without restriction, suppose $\mathcal{L} = \{1, \ldots, |\mathcal{L}|\}$ for *case 1* and for *case 2* let $|\mathcal{L}| = n$. Let $u, v \in \Gamma^*$ be the input words. Moreover, let $\pi_i : G \to G_i$ be the canonical projection defined by

$$w_k \mapsto \begin{cases} w_k, & \text{if } \mathrm{alph}(w_k) = i \\ 1, & \text{otherwise.} \end{cases}$$

For direct products we have $u \sim_G v$ if and only if $\pi_i(u) \sim_{G_i} \pi_i(v)$. Hence, the circuit

$$\bigwedge_{i=1}^{|\mathcal{L}|} \pi_i(u) \sim_{G_i} \pi_i(v))$$

solves the conjugacy problem of G. □

7.2 in free products

The following lemma is the conjugacy theorem for free products [LS01, Theorem 1.4] and crucial for the parallel solution of the conjugacy problem in free products. We give its proof for completeness here. A word $w \in \Gamma^*$ is called *cyclically reduced* if it is reduced and $w = aub$ with $a \in \Gamma_\alpha, b \in \Gamma_\beta$ implies $\alpha \neq \beta$.

Lemma 7.2.1. *Let* $G = *_{\alpha \in \mathcal{L}} G_\alpha$ *be a free product. Let* $u, v \in \Gamma^*$ *be cyclically reduced words with* $|u|, |v| > 1$*, then* $u \sim_G v$ *if and only if for some cyclic permutation* u^* *of* u *we have* $u^* =_G v$ *and* $|u| = |v|$*.*

Proof. Suppose there is some cyclic permutation u^* of u with $u =_G v$. Then obviously u is conjugate to v in G. For the other direction, suppose $u \sim_G v$. Thus, there is some $z \in G$ with $z^{-1}uz =_G v$. Let $u = u_1 \cdots u_m$, $v = v_1 \cdots v_m$ Moreover, let z be reduced. We proof the claim by induction on $|z|$. For $|z| = 0$ we have $u^* = u = v$. Now suppose $|z| > 0$ and $z = z_1 \cdots z_k$. We have

$$z_1 \cdots z_k u_1 \cdots u_n \overline{z_k} \cdots \overline{z_1} =_G v_1 \cdots v_m.$$

Since v is cyclically reduced, some reduction must occur on the left hand side of the above equation. This is only possible for $z_k u_1$ or $u_n \overline{z_k}$. Now, u is cyclically reduced, hence u_1 and u_n belong to different base groups. Without restriction, suppose the reduction happens for $z_k u_1$. If $z_k u_1 \neq_G 1$, then no further reduction can happen on the left hand side of the equation. This would be a contradiction to v being cyclically reduced. Therefore, we must have $z_k u_1 =_G 1$ and thus $\overline{z_k} = u_1$. This leads to

$$z_1 \cdots z_{k-1}(u_2 \cdots u_n \cdot u_1)\overline{z_{k-1}} \cdots \overline{z_1} =_G v_1 \cdots v_m.$$

By induction v is a cyclic permutation of $u_2 \cdots u_n \cdot u_1$, which in turn is a cyclic permutation of u. □

By the previous lemma, the conjugacy problem reduces to the transposition problem for cyclically reduced words. The next lemmata show, how to compute such a cyclically reduced word.

Lemma 7.2.2. *Let $u \in \Gamma^*$ be reduced. Then there exists $p \in \Gamma^*$ reduced and a unique $u' \in \Gamma^*$ of minimal length with $u = pu'\overline{p}$.*

Proof. If $p = 1$ is the only word with $u = pu'\overline{p}$, then we have $p = 1, u' = u$ and we are done. Otherwise we have some $a \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$ with $u = au'\overline{a}$. The lemma follows by induction on the length of u. □

Lemma 7.2.3. *Let $u \in \Gamma^*$ be reduced. Then there exist reduced $p, r, m, s \in \Gamma^*$ with*

1. *$u = prms\overline{p}$,*
2. *$|r| = |s| = |[sr]| \leq 1$,*
3. *$m[sr]$ is cyclically reduced and $u \sim_G m[sr]$.*

Proof. By Lemma 7.2.2 there exists the decomposition $u = pu'\overline{p}$ with u' of minimal length. If u' is cyclically reduced, we have $r = s = 1$. Otherwise we can write $u' = rms$ with $m \in \Gamma^*$ reduced and $r, s \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$. Since u' is of minimal length, we must have $sr \neq_G 1$. Therefore, $m[sr]$ is cyclically reduced. □

Lemma 7.2.4. *Let $G = *_{\alpha \in \mathcal{L}} G_\alpha$ be a free product of f. g. groups and $u \in \Gamma^*$ be reduced. Then there is an AC^0 Turing reduction onto the word problem of G to compute some cyclically reduced word u' of u with $u' \sim_G u$.*

Proof. Without restriction, let $|\mathcal{L}| > 1$. Otherwise $u' = u$ is cyclically reduced. We decompose u into $prms\overline{p}$ as in Lemma 7.2.3. Now, consider the word $w = uu$

and let $\varepsilon = |s| = |r| \leq 1$. The length of w is $|w| = 4|p|+2|m|+4\varepsilon$ and the geodesic length of w is $\|w\| = 2|p| + 2|m| + 3\varepsilon$. The difference of the two is

$$|w| - \|w\| = 2|p| + \varepsilon.$$

Since $|w| = 2|u|$ and $2|p|$ is even, this leads to

$$\varepsilon = \|w\| \mod 2.$$

Knowing the value of ε allows us to compute

$$|p| = \frac{1}{2}(|w| - \|w\| - \varepsilon).$$

The input word w is given as n blocks $w_i \in \Gamma^*$ with $w = w_1 \cdots w_n$. For the length of w, the problem is that some of the w_i equal the identity Hence, the TC^0-circuit

$$|w| \equiv \#i : w_i \neq_G 1$$

computes $|w|$. By Theorem 6.2.3 a geodesic of w is AC^0-computable with oracle gates for the word problem of G. Let $\widehat{w}$ be this geodesic. As for $|w|$ we have

$$\|w\| = |\widehat{w}| \equiv \#i : \widehat{w}_i \neq_G 1.$$

Since $|\mathcal{L}| > 1$ the word problem of G is uTC^0-hard and hence, any TC^0-computation can be transformed into a uniform AC^0-computation with oracle gates for $\mathrm{WP}(G)$. Now, selecting the subword $u' = u_{|p|+1} \cdots u_{|u|-|p|}$ if $\varepsilon = 0$ and the subword $u' = u_{|p|+2} \cdots [u_{|u|-|p|} \cdot u_{|p|+1}]$ if $\varepsilon = 1$ as output can be done in AC^0. □

Theorem 7.2.5. *The conjugacy problem of free products is AC^0 reducible to the conjugacy problem of the base groups and the word problem of the free group of rank 2.*

1. *Let $G = *_{\alpha\in\mathcal{L}} G_\alpha$ be a free product of f. g. groups. Then*

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

   that is there is an AC^0 Turing reduction of the conjugacy problem of G to the conjugacy problem of the G_α and the word problem of the free group F_2 of rank 2.

2. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then*

$$\mathrm{CP}(\mathbf{F}\mathcal{C}) \in \mathsf{uAC}^0(\{\mathrm{CP}(\mathcal{C}), \mathrm{WP}(F_2)\}),$$

   that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$ and words $u, v \in \Gamma^*$ the conjugacy problem $u \sim v$ in $G = *_{i=1}^n G_i$ can be solved in uniform AC^0 with oracle gates for the conjugacy problem of $\mathcal{C}$ and the word problem of the free group F_2 of rank 2.

Proof. Let $u, v \in \Gamma^*$ be the input words. For *case 1.* we assume without restriction $\mathcal{L} = \{1, \ldots, |\mathcal{L}|\}$ and for *case 2.* we let $|\mathcal{L}| = n$. The oracle gate for the conjugacy problem of G_i can be used for the word problem of G_i since for any $w \in \Gamma_i^*$ we have $w \sim_{G_i} 1$ if and only if $w =_{G_i} 1$. Hence the word problem of G can be solved by oracle gates for the conjugacy problem of the base groups and by oracle gates for $\mathrm{WP}(F_2)$. This follows for *case 1.* by Theorem 5.4.7 and for *case 2.* by Theorem 5.4.10. Now, by Theorem 6.2.3 a geodesic for u and v can be computed in AC^0 by using oracle gates for G and then by Lemma 7.2.4 cyclically reduced words of u and v can be computed. Let $\widehat{u} = \widehat{u}_1 \cdots \widehat{u}_n$ and $\widehat{v} = \widehat{v}_1 \cdots \widehat{v}_n$ be these cyclically reduced words. Now, by Lemma 7.2.1 we have for $|\widehat{u}|, |\widehat{v}| > 1$ that $u \sim_G v$ if and only if $\widehat{u}$ is a cyclic permutation of $\widehat{v}$. Thus, the circuit

$$1 \neq |\widehat{u}| \wedge |\widehat{u}| = |\widehat{v}| \wedge \left(\widehat{u} =_G \widehat{v} \vee \bigvee_{i=1}^{|\mathcal{L}|} \widehat{u}_i \cdots \widehat{u}_n \widehat{u}_1 \cdots \widehat{u}_{i-1} =_G \widehat{v} \right)$$

solves the conjugacy problem of G if $|\widehat{u}|, |\widehat{v}| > 1$. For $|\widehat{u}| = |\widehat{v}| = 1$ the oracle gates for the base groups have to be used.

$$1 = |\widehat{u}| \wedge |\widehat{u}| = |\widehat{v}| \wedge \mathrm{alph}(\widehat{u}_1) = \mathrm{alph}(\widehat{v}_1) \wedge \left(\bigvee_{i=1}^{|\mathcal{L}|} \mathrm{alph}(\widehat{u}_1) = \mathrm{bin}(i) \wedge \widehat{u}_1 \sim_{G_i} \widehat{v}_1 \right)$$

Finally, combining both circuits yields a solution to the conjugacy problem of G. □

Corollary 7.2.6. *For free groups we obtain the following results.*

1. *The conjugacy problem of a fixed free group is in* $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$.
2. *The uniform conjugacy problem of free groups is in* $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$.

7.3 in graph products

The basis of conjugacy in graph products is conjugacy in trace monoids. Conjugacy in trace monoids was first considered by Christine Duboc [Dub86]. Linear time solutions for the conjugacy problem were shown some years later by [LWZ90] and for the involved factor problem by [HY92]. Parallel solutions to problems in trace monoids have - to the best of my knowledge - so far been only known for the word problem.

We introduce the notion of transposition for the conjugacy problem in graph products. For its definition recall that for words $u, v \in \Gamma^*$ we have $u \equiv v$ if

and only if their dependence graphs $D(u)$ and $D(v)$ are isomorphic. We say u is transposed to v if there exists $r, s \in \Gamma^*$ such that $u \equiv rs$ and $v \equiv sr$. The transposition relation is reflexive and symmetric. By $\approx$ we denote the transitive closure of the transposition relation. We further say a reduced word $u \in \Gamma^*$ is *cyclically reduced* if $u \equiv au'b$ with $u' \in \Gamma^*$ and $a \in \Gamma_\alpha, b \in \Gamma_\beta$ implies $\alpha \neq \beta$. A word $u \in \Gamma^*$ is connected if its dependence graph $D(u)$ is connected.

Example 7.3.1. *Let G be the graph group with generating alphabet $\Sigma = \{a, b, c\}$ and independence relation $I = \{(a, b), (b, a)\}$. Figure 7.1 on page 100 shows two dependence graphs, both being connected. On the left, the dependence graph of the word $u = bc\bar{b}a$ is reduced but not cyclically reduced. The dependence graph of $v = ca$ on the right is cyclically reduced and conjugate to u.*

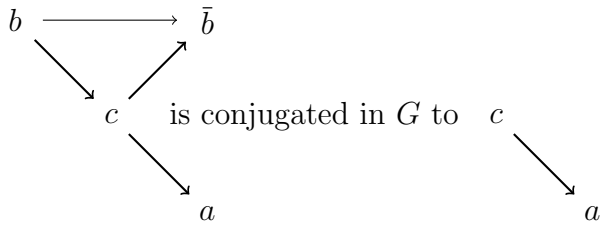


Figure 7.1: Dependence graph (Hasse diagram consists of the solid edges only) of $u = bc\bar{b}a$ on the left and $v = b = ca$ on the right (see Example 7.3.1).

The key observation to solve the conjugacy problem in graph products is that for cyclically reduced and connected words with length greater than one we have $u \sim v$ if and only if $u \approx v$ if and only if u is a factor of $v^{|\mathcal{L}|}$.

Lemma 7.3.2. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and let $u \in \Gamma^*$ be reduced. Then there exists $p \in \Gamma^*$ reduced and a unique $u' \in \Gamma^*$ of minimal length with $u \equiv pu'\bar{p}$.*

Proof. We present a length-reducing rewriting procedure on u and show that it is strongly confluent. Thus, we obtain a minimal and unique u'. If $p = \varepsilon$ is the only candidate for p, then $u' = u$ and we are done. Otherwise, there is some $\alpha \in \mathcal{L}$ and $a \in \Gamma_\alpha$ such that $u \equiv au_1\bar{a}$. We rewrite u into u_1. Assume we also have some $\beta \in \mathcal{L}$ with $\alpha \neq \beta$ and $b \in \Gamma_\beta$ with $u \equiv bu_2\bar{b}$. Then we must have $(\alpha, \beta) \in I$ and therefore, $u \equiv abu_3\bar{b}\bar{a}$. Thus, $u_1 \equiv bu_3\bar{b}$ and $u_2 \equiv au_3\bar{a}$. Hence, the rewriting procedure is strongly confluent. □

Lemma 7.3.3. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and let $u \in \Gamma^*$ be reduced. Then there exist reduced $p, r, m, s \in \Gamma^*$ such that*

1. *$u \equiv prms\bar{p}$,*
2. *$|r|_\alpha = |s|_\alpha = |[sr]|_\alpha \leq 1$ for all $\alpha \in \mathcal{L}$,*
3. *$m[sr]$ is cyclically reduced and $u \sim_G m[sr]$.*

Proof. By Lemma 7.3.2 there exists $p \in \Gamma^*$ and a minimal $u' \in \Gamma^*$ with $u \equiv pu'\bar{p}$. Since p is maximal in the above choice, there are unique maximal $r, s \in \Gamma^*$ and $m \in \Gamma^*$ with $u' \equiv rms$ such that $|[sr]|_\alpha = |r|_\alpha = |s|_\alpha \leq 1$ for all $\alpha \in \mathcal{L}$. Let $M = \mathrm{alph}(r) = \mathrm{alph}(s)$ be the alphabet of r and s. Then $M \subseteq \mathcal{L}$ consists of pairwise independent nodes since $|[sr]|_\alpha = |r|_\alpha = |s|_\alpha$. Therefore, we have

$$r \equiv \prod_{\alpha\in M} r_\alpha \text{ and } s \equiv \prod_{\alpha\in M} s_\alpha$$

with $r_\alpha, s_\alpha \in \Gamma_\alpha$. Now, the word

$$m[sr] \equiv m \prod_{\alpha\in M} [s_\alpha r_\alpha]$$

is cyclically reduced since u is reduced and r, s were chosen maximal. Finally, with $u \sim m[sr]$ the result follows. □

Lemma 7.3.4. *For graph products computing cyclically reduced words reduces to the word problem.*

1. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f. g. groups. Then we have that on input of a word $u \in \Gamma^*$ a cyclically reduced word u' of u with $u' \sim_G u$ can be computed in $\mathsf{uAC}^0(\mathrm{WP}(G))$.*
2. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then we have that on input of groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I and a word $u \in \Gamma^*$ a cyclically reduced word u' of u with $u' \sim u$ in $\mathrm{GP}\left(\{1, \ldots, n\}, I; (G_\alpha)_{\alpha\in\{1,\ldots,n\}}\right)$ can be computed in $\mathsf{uAC}^0(\mathrm{WP}(\mathbf{GP}\mathcal{C}))$.*

Proof. For *case 1* we assume without restriction that G contains a free product. Otherwise, G is a direct product. But then, a geodesic $\widehat{u}$ of u can be computed in AC^0 with oracle gates for the word problem of G by Theorem 6.1.1. The word $u' = \widehat{u}$ is cyclically reduced. Hence, we may assume that G contains a free product. Therefore, the word problem is uTC^0-hard. In *case 2* the word problem is NL-hard by Corollary 5.6.10. Thus, we may use TC^0-circuits in both cases. Moreover, by

Theorem 6.3.3 for both cases a geodesic of u can be computed in AC^0 with oracle gates for the word problem. Hence, we further assume u to be reduced.

By Lemma 7.3.3 there exist $p, r, m, s \in \Gamma^*$ with $u \equiv prms\bar{p}$. The goal is to compute the factor $m[sr]$. Consider the word $w = uu$ and let $\varepsilon_\alpha = |r|_\alpha = |s|_\alpha \leq 1$. By Theorem 6.3.3 a geodesic $\widehat{w}$ of w can be computed in AC^0 with oracle gates for the word problem of G. Since $m[sr]$ is cyclically reduced we have $\widehat{w} \equiv prm[sr]ms\bar{p}$. Now, the α-length of w is $|w|_\alpha = 4|p|_\alpha + 2|m|_\alpha + 4\varepsilon_\alpha$. Moreover, the α-length of $\widehat{w}$ is $|\widehat{w}|_\alpha = 2|p|_\alpha + 2|m|_\alpha + 3\varepsilon_\alpha$. The difference of the two is

$$|w|_\alpha - |\widehat{w}|_\alpha = 2|p|_\alpha - \varepsilon_\alpha.$$

Since $|w|_\alpha$ and $2|p|_\alpha$ are even we obtain

$$\varepsilon_\alpha = |\widehat{w}|_\alpha \mod 2.$$

This allows us to compute $|p|_\alpha$ and $|m|_\alpha$ via

$$|p|_\alpha = \frac{1}{2}(|w|_\alpha - |\widehat{w}|_\alpha - \varepsilon_\alpha)$$
$$|m|_\alpha = \frac{1}{2}(|w|_\alpha - 4|p|_\alpha - 4\varepsilon_\alpha)$$

Let $\widehat{w} = \widehat{w}_1 \cdots \widehat{w}_k$ be the factorization of $\widehat{w}$ with $k = |\widehat{w}|$. Now, the i-th factor $\widehat{w}_i$ belongs to the factor $m[sr]$ of $\widehat{w}$ if and only if

$$|p|_\alpha + \varepsilon_\alpha < i < k - |p|_\alpha - |m|_\alpha - \varepsilon_\alpha$$

for $\widehat{w}_i \in \Gamma_\alpha$. Thus, the circuit

$$F(i) \equiv \bigvee_{i=1}^{|\mathcal{L}|} \mathrm{alph}(\widehat{w}_i) = \mathrm{bin}(\alpha) \wedge (|p|_\alpha + \varepsilon_\alpha < i < k - |p|_\alpha - |m|_\alpha - \varepsilon_\alpha)$$

decides, whether $\widehat{w}_i$ belongs to the factor $m[sr]$. Determining the lengths can be done in TC^0 via

$$|w|_\alpha \equiv \#i : w_i \neq_G 1 \wedge \mathrm{alph}(w_i) = \mathrm{bin}(\alpha)$$
$$|\widehat{w}|_\alpha \equiv \#i : \widehat{w}_i \neq_G 1 \wedge \mathrm{alph}(\widehat{w}_i) = \mathrm{bin}(\alpha)$$

Now, let the output word be $u' = u'_1 \cdots u'_n$ where $n = |w|$. The predicate

$$pos(i, k) \equiv F(i) \wedge (k = \#j \leq i : F(j))$$

is true, if the position of $\widehat{w}_i$ in the output word u' is k. Hence, selecting the correct factor for u'_k can be done in AC^0 by using the predicate $pos(i, k)$.

All computations can be performed in uAC^0 with oracle gates for the word problem. By construction we have $u' = m[sr] \sim_G u$, hence the result follows. □

For the following lemmata it is important that the input words are connected. By the next lemma, we can reduce the conjugacy problem in graph products to connected words.

Lemma 7.3.5. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and $u, v \in \Gamma^*$ be reduced words. Moreover, let $\mathrm{alph}(u), \mathrm{alph}(v) \subseteq \mathcal{L}_1 \uplus \ldots \uplus \mathcal{L}_k$ with $\mathcal{L}_i \times \mathcal{L}_j \subseteq I$ for all $i \neq j$. Then we have*

$$u \sim_G v \text{ if and only if } \pi_i(u) \sim_{G_i} \pi_i(v) \text{ for all } 1 \leq i \leq k$$

where $G_i = \mathrm{GP}(\mathcal{L}_i, I \cap \mathcal{L}_i \times \mathcal{L}_i; (G_\alpha)_{\alpha\in\mathcal{L}_i})$ and $\pi_i : G \to G_i$ is the canonical projection of G onto G_i

Proof. Let $u \sim_G v$, then there exists some $z \in \Gamma^*$ with $uz =_G zv$. Since $\pi_i(u)\pi_i(z) = \pi_i(uz) = \pi_i(zv) = \pi_i(z)\pi_i(v)$ in G_i we have $\pi_i(u) \sim_{G_i} \pi_i(v)$ for all $1 \leq i \leq k$.

Contrary, let $\pi_i(u) \sim_{G_i} \pi_i(v)$ for all $1 \leq i \leq k$ and let $\Gamma_i = G_i \setminus \{1\}$. Then there exist $z_i \in \Gamma_i^*$ with $z_i\pi_i(u) = \pi_i(v)z_i$ for all i. Since $\mathcal{L}_i \times \mathcal{L}_j \subseteq I$ for $i \neq j$ we have

$$(z_1 \cdots z_n)u =_G \prod_{i=1}^{k} z_i\pi_i(u) =_G \prod_{i=1}^{k} \pi_i(v)z_i =_G v(z_1 \cdots z_n)$$

and thus, $u \sim_G v$. □

For connected words we must have that they share the same alphabet to be conjugate by the next lemma.

Lemma 7.3.6. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and $u, v \in \Gamma^*$ be cyclically reduced and connected words. Then $u \sim_G v$ implies $\mathrm{alph}(u) = \mathrm{alph}(v)$.*

Proof. Let $u \sim_G v$, then there is some reduced $z \in \Gamma^*$ with $zu =_G vz$. Suppose, there is some $\alpha \in \mathrm{alph}(u) \setminus \mathrm{alph}(v)$. Since $\alpha \notin \mathrm{alph}(v)$ we must have

$$|z|_\alpha = |vz|_\alpha = \|zu\|_\alpha < |u|_\alpha + |z|_\alpha.$$

Hence, a cancellation between u and z must occur. Let $u = u'a_1$ and $z = a_2z'$ for some $a_1, a_2 \in \Gamma_\alpha$. Since $uz =_G zv =_G a_2z'v$ we must have that a_2 is minimal in u. Assume $|u| > 1$, then $u \equiv a_2u''a_1$ is not cyclically reduced since u is connected. A contradiction. Hence, $|u| = 1$ and thus, by the above equality $a_1a_2 \neq 1$. Otherwise $|z|_\alpha = \|zu\|_\alpha < |z|_\alpha$ Moreover, since a_2 is minimal in u, we must have $a_1a_2 =_G a_2$ and therefore $a_1 =_G 1$. A contradiction to $\alpha \in \mathrm{alph}(u)$. By symmetry it follows $\mathrm{alph}(u) = \mathrm{alph}(v)$. □

Next, we show that the conjugacy problem in graph products is (with some minor restrictions) equivalent to the transitive closure of the transposition problem. For the proof of this equivalence we do need a Levi like lemma. Luckily, the Levi lemma for traces does hold in our setting since we only consider cyclically reduced words. A proof of the Levi lemma for traces can be found in [DR95, Theorem 3.2.2].

Lemma 7.3.7. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and let $x, y, z, t \in \Gamma^*$. Moreover, let xy and zt be reduced. If $xy \equiv zt$, then there exist $p, q, r, s \in \Gamma^*$ with $\mathrm{alph}(r) \times \mathrm{alph}(s) \subseteq I$ such that*

$$x \equiv pr, z \equiv ps, y \equiv sq \text{ and } t \equiv rq.$$

Lemma 7.3.8. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and let u, v be cyclically reduced and connected words. Moreover, let $\mathrm{alph}(u) = \mathrm{alph}(v)$ with $|\mathrm{alph}(u)| > 1$. Then we have $u \sim_G v$ if and only if $u \approx v$.*

Proof. Let $u \sim_G v$. Hence, there is some reduced $z \in \Gamma^*$ such that $uz =_G zv$. Assume uz is not reduced. Then there exist $a_1, a_2 \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$ and $u', z' \in \Gamma^*$ with $u \equiv u'a_1$ and $z \equiv a_2 z'$. Suppose $a_1 a_2 =_G 1$, then

$$au'z' =_G auz =_G azv =_G z'v.$$

Hence $au' \sim_G v$ and u is transposed to au'. Furthermore, by induction on $|z|$ we obtain

$$u \approx au' \approx v.$$

The same reasoning holds if zv is not reduced. Now, we may assume $a_1 a_2 \neq_G 1$ for all such decompositions of uz and zv. Let again $u = u'a_1$ and $z = a_2 z'$ for $a_1, a_2 \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$. Then a_2 is minimal in u since u is connected, $|\mathrm{alph}(u)| > 1$ and $uz =_G zv$. But then we have $u = a_2 u'' a_1$ and hence, u is not cyclically reduced. A contradiction.

Thus, for the rest of the proof we may assume that uz and zv is reduced. But then we must have $uz \equiv zv$ in G. By Lemma 7.3.7 there exist $p, q, r, s \in \Gamma^*$ such that $(r, s) \in I$ and

$$u \equiv pr, z \equiv sq \equiv ps \text{ and } v \equiv rq.$$

If $|z| = |s|$ then $p = q = 1$ and therefore $u \equiv v$. Hence, we may assume $|s| < |z|$. The above equations lead to $(rp)s \equiv rsq \equiv s(rq)$, since $(s, r) \in I$. Thus, $rp \sim_G rq$ and by induction $rp \approx rq$. Since $u \equiv pr$ and rp are transposed, it follows

$$u \approx rp \approx pr \approx qr \equiv v.$$

□

Now, to solve the conjugacy problem it mainly suffices to solve the transitive closure of the transposition problem. The $\approx$ relation was already characterised by [Dub86] for Mazurkiewicz traces back in 1986. The following lemma states her result adapted to graph products.

Lemma 7.3.9. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product and let $u, v \in \Gamma^*$ be cyclically reduced and connected words. Then we have $u \approx v$ if and only if*

1. *$|u|_\alpha = |v|_\alpha$ and*
2. *there exists $p, q \in \Gamma^*$ reduced with $puq \equiv v^{|\mathcal{L}|}$.*

Proof. Let $u \approx v$, then we must have $|u|_\alpha = |v|_\alpha$. For *part 2*, we show two things. First, there is some $k \in \mathbb{N}$ with $puq \equiv v^k$ and second, k can be bounded by $|\mathcal{L}|$. For the first part, we proceed by induction on the number of transpositions to transform u into v. If $u \equiv v$, then $p = \varepsilon, q = v$ and hence, $puq \equiv v^1$. Now, let $u \not\equiv v$. Then there exist $r, s \in \Gamma^*$ with $u \approx sr$ and $v \equiv rs$. By induction, there exist $p', q' \in \Gamma^*$ with $p'uq' \equiv (sr)^k$ for some $k \in \mathbb{N}$. Let $p = rp'$ and $q = q's$, then we have

$$puq \equiv rp'uq's \equiv r(sr)^k s \equiv (rs)^{k+1} \equiv v^{k+1}.$$

For the bound on k, let $puq \equiv v_1 \dots v_k$ with $v_\ell \equiv v$ for each ℓ. We consider $D(u)$ as a subgraph of $D(v_1 \cdots v_k)$. Some minimal vertex i_0 of $D(u)$ must be located in $D(v_1)$. Let j be a vertex of $D(u)$. Since $D(u)$ is connected, there is some (undirected) path from i_0 to j. By $d(i_0, j)$ we denote the length of a shortest path from i_0 to j. We claim that j is a vertex in $D(v_1 \cdots v_{d+1})$ where $d = d(i_0, j)$. Since any shortest path in $D(u)$ is bounded by $|\mathcal{L}| - 1$ the claim finishes the second part.

For the proof of the claim, we proceed by induction. For $d > 0$, let i be the preceding vertex of j on a shortest path from i_0 to j, then we have $d(i_0, i) = d-1 < d = d(i_0, j)$. Hence, by induction i is a vertex in $D(v_1 \cdots v_d)$. Let $\lambda(j) \in \Gamma_\alpha$ for some $\alpha \in \mathcal{L}$. Now, on any directed path from i to j in $D(v_1 \cdots v_k)$ there are at most $|u|_\alpha$ vertices with a label in Γ_α since u is a factor of v^k. Hence, j is a vertex in $D(v_1 \cdots v_{d+1})$ since $|u|_\alpha = |v|_\alpha$.

For the other direction, let $|u|_\alpha = |v|_\alpha$ for all $\alpha \in \mathcal{L}$ and $p, q \in \Gamma^*$ be reduced with $puq \equiv v^{|\mathcal{L}|}$. We proceed by induction on p. For $|p| = 0$ we have $u \equiv v$, since $|u|_\alpha = |v|_\alpha$ for all $\alpha \in \mathcal{L}$. Now, let $|p| > 0$. Then we have $p = ap'$ for some $a \in \Gamma$ and $p' \in \Gamma^*$. Therefore, we have $v^{|\mathcal{L}|} \equiv puq \equiv (ap')uq$ and hence, $v \equiv av'$ for some $v' \in \Gamma^*$. This leads to $p'u(qa) = (v'a)^{|\mathcal{L}|}$. Finally, by induction we get

$$u \approx v'a \approx av' \equiv v.$$

□

Hence, the transitive closure of the transposition problem for cyclically reduced words leads to the problem of verifying, whether some word u is a factor of some word t. This check differers in the non-uniform and uniform variant of the conjugacy problem. So far, the steps to solve the conjugacy problem of the graph product $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ are as follows.

Input: $u, v \in \Gamma^*$. **Question:** $u \sim_G v$?

1. Compute reduced u, v
2. Compute cyclically reduced u, v
3. Compute connected components of u, v
4. Check $\mathrm{alph}(u) = \mathrm{alph}(v)$. Otherwise $u \not\sim_G v$.
5. a) If $|\mathrm{alph}(u)| = 1$, then $u \sim_G v$ if and only if $u \sim_{G_\alpha} v$ for $\alpha = \mathrm{alph}(u)$.
   b) If $|\mathrm{alph}(u)| > 1$, then $u \sim_G v$ if and only if $u \approx v$ in G if and only if u is a factor of $v^{|\mathcal{L}|}$ and $|u|_\alpha = |v|_\alpha$.

7.3.1 The non-uniform case

For a fixed graph product the factor problem can be solved in AC^0 with oracle gates for the word problem of the graph product. The basic idea for this test is by personal communication with Markus Lohrey.

Lemma 7.3.10. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ be a graph product of f. g. groups. Then on input of reduced words $u, w \in \Gamma^*$ the question whether there exist reduced $p, q \in \Gamma^*$ such that $puq \equiv w$ can be decided in $\mathsf{uAC}^0(\mathrm{WP}(G)))$.*

Proof. Let $w = w_1 \cdots w_n$ be the factorization of w. Suppose u is a factor of w, that is there exist $p, q \in \Gamma^*$ reduced with $puq \equiv w$. We consider the dependence graph $D(u)$ as a subgraph of $D(w)$. Now, for any two vertices i, j in $D(u)$, we have that any $i \xrightarrow{*} j$ path in $D(w)$ is already a path in $D(u)$. Conversely, any subgraph D of $D(w)$ which is isomorphic to $D(u)$ and satisfies this property yields that u is a factor of w.

The idea to solve the problem is to guess the subgraph D. Let Min be the set of minimal vertices in D and Max be the set of maximal vertices in D. The sets Min and Max contain only independent vertices, hence $|Min|, |Max| \leq |\mathcal{L}|$. Two vertices i and j are independent, if there is no $i \xrightarrow{*} j$ path and no $j \xrightarrow{*} i$ path. Since we do have dependence graphs, there is some $i \xrightarrow{*} j$ path if there is some

$i \xrightarrow{*} j$ path with length less than $|\mathcal{L}|$. Let $k = |\mathcal{L}|$, then $i \xrightarrow{*} j$ can be expressed by

$$\exists\, i_1 \leq \ldots \leq i_k : i = i_1 \wedge j = i_k \wedge (w_{i_1}, w_{i_2}) \notin I \wedge \ldots \wedge (w_{i_{k-1}}, w_{i_k}) \notin I.$$

Now, a vertex i belongs to D, if there are vertices $x \in Min$ and $y \in Max$ with paths $x \xrightarrow{*} i$ and $i \xrightarrow{*} y$. Let $u' = u'_1 \cdots u'_n$ with

$$u'_i = \begin{cases} w_i, & \text{if } i \text{ is a vertex in } D \\ 1, & \text{otherwise.} \end{cases}$$

By the above, u' can be computed in AC^0 for fixed Min and Max and by using the oracle gate for $\mathrm{WP}(G)$ we can check if $u' =_G u$. We denote the constructed circuit by $D(Min, Max, u)$. Since the cardinality of Min and Max is bounded by $|\mathcal{L}|$ there are only polynomially many possibilities for Min and Max in $D(w)$. Hence,

$$\bigvee_{\substack{Min \subseteq D(w) \\ |Min| < |\mathcal{L}|}} \bigvee_{\substack{Max \subseteq D(w) \\ |Max| < |\mathcal{L}|}} D(Min, Max, u)$$

decides the factor problem in AC^0. □

Finally, we have all the ingredients to solve the conjugacy problem for a fixed graph product.

Theorem 7.3.11. *Let $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha \in \mathcal{L}})$ with G_α f. g. groups. Then*

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

that is there is an AC^0 Turing reduction of the conjugacy problem of G onto the word problem of G and the conjugacy problem of the G_α.

Proof. If G is actually a direct product, then the result follows by Theorem 7.1.1. Hence, we may assume that G contains a free product. Thus, the word problem of G is uTC^0-hard and therefore any TC^0-computation can be transformed into a uniform AC^0-computation with oracle gates for $\mathrm{WP}(G)$. We may use the oracle gates $\mathrm{CP}(G_\alpha)$ for the conjugacy problem as an oracle for $\mathrm{WP}(G_\alpha)$ since $w =_{G_\alpha} 1$ if and only if $w \sim_{G_\alpha} 1$. By Theorem 5.6.5 we obtain an oracle for $\mathrm{WP}(G)$ by using an AC^0 Turing reduction onto the oracles gates for the conjugacy problem of the base groups G_α and the oracle $\mathrm{WP}(F_2)$ for the word problem of the free group on two generators. Hence, we may use oracle gates for $\mathrm{WP}(G)$ in the following.

By Lemma 7.3.4 a cyclically reduced word of u and v can be computed in AC^0 with oracle gates for the word problem. Let in the following $\alpha \in \mathcal{L}$. The predicate

$con_w(i, \alpha)$ decides if a vertex i is connected to a vertex with label in Γ_α in the dependence graph $D(w)$. Any two vertices are connected, if there is an undirected path in the dependence graph of length at most $|\mathcal{L}| - 1$. Hence, with $k = |\mathcal{L}|$ we have that $con_w(i, \alpha)$ is equivalent to

$$\exists\, i_1, \ldots, i_k : i = i_1 \wedge \operatorname{alph}(w_j) = \alpha \wedge (w_{i_1}, w_{i_2}) \notin I \wedge \ldots \wedge (w_{i_{k-1}}, w_{i_k}) \notin I.$$

Consider the projections $\pi_{w,\alpha}$ defined by

$$w_i \mapsto \begin{cases} w_i, & \text{if } con_w(i, \alpha) \text{ is true} \\ 1, & \text{otherwise.} \end{cases}$$

Let u_α (respectively v_α) be the projection of $\pi_{u,\alpha}$ applied to u (respectively of $\pi_{v,\alpha}$ applied to v). Then we have by Lemma 7.3.5 that $u \sim_G v$ if and only if $u_\alpha \sim_G v_\alpha$ for all $\alpha \in \mathcal{L}$. By construction, the words u_α and v_α are connected. Thus, the conjugacy problem reduces to connected words via

$$\bigwedge_{\alpha \in \mathcal{L}} u_\alpha \sim_G v_\alpha.$$

Hence, for the rest of the circuit construction we can assume that u and v are connected. For $|\operatorname{alph}(u)| > 1$ we have by Lemma 7.3.8 that $u \sim_G v$ if and only if $u \approx v$. By Lemma 7.3.9 we have $u \approx v$ if and only if u is a factor of v^k and $|u|_\alpha = |v|_\alpha$. Let $factor(u, w)$ be the circuit from Lemma 7.3.10 to check, whether u is a factor of w. Then the circuit

$$\left(|u| = 1 \wedge |v| = 1 \wedge \left(\bigvee_{\alpha \in \mathcal{L}} \operatorname{alph}(u) = \operatorname{alph}(v) = \alpha \wedge u \sim_{G_\alpha} v \right) \right) \vee$$
$$\left(|u| > 1 \wedge |v| > 1 \wedge factor(u, v^k) \wedge \bigwedge_{\alpha \in \mathcal{L}} |u|_\alpha = |v|_\alpha \right)$$

decides the conjugacy problem $u \sim_G v$ with oracle gates for the conjugacy problem in G_α and oracle gates for the word problem of G inside of $factor(u, v^k)$. □

Corollary 7.3.12. *The conjugacy problem of a fixed graph group or right angled Coxeter group is in* $\mathsf{uAC}^0(\mathsf{C}_=\mathsf{NC}^1)$.

Proof. The base groups of a graph group are $H = \langle a \rangle$ (respectively $H = \langle a \mid a^2 \rangle$ for right angled Coxeter groups). We have $x \sim_H y$ if and only if $x =_H y$. Moreover, the word problem of H is in $\mathsf{TC}^0 \subseteq \mathsf{C}_=\mathsf{NC}^1$. Hence, with $\mathrm{WP}(F_2) \in \mathsf{C}_=\mathsf{NC}^1$ the result follows. □

7.3.2 The uniform case

In the uniform case we directly obtain the following hardness result as a consequence of the hardness result on the uniform word problem.

Corollary 7.3.13. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then we have that* $\mathrm{CP}(\mathbf{GP}\mathcal{C})$ *is* NL*-hard.*

Proof. The word problem $w =_G 1$ is the special case of the conjugacy problem $w \sim_G 1$. Hence, the corollary follows by Corollary 5.6.10. □

Unfortunately, the above approach on verifying if u is a factor of w does not work in the uniform setting. Since the list of groups is part of the input, guessing the minimal and maximal vertices is not possible as there are exponentially many possibilities.

Let $H(u)$ be the Hasse diagram of u and $H(w)$ be the Hasse diagram of w. Moreover, if $w \equiv puq$ then the Hasse diagram $H(u)$ is a subgraph of $H(w)$. Any directed path in the Hasse diagram is uniquely determined by a sequence of labels. Furthermore, any undirected path in the Hasse diagram is uniquely determined by a sequence of labels and the orientation of the edges to follow. Suppose that $D(u)$ and hence, $H(u)$ is connected. If D is a subgraph of $H(w)$ corresponding to the factor u, it suffices to know one vertex s of D to reconstruct the whole subgraph D in NL. Let t be the vertex in $D(u)$ to be reconstructed. Since $D(u)$ is connected, there is an undirected $s \xrightarrow{*} t$ path, which can be guessed in NL. By the above this path is uniquely determined by its labels and the orientation of its edges, hence it can be followed in $D(w)$ to find the corresponding vertex t in D.

Lemma 7.3.14. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then on input of groups $G_1, \ldots, G_n$, an independence relation I and reduced and connected words $u, w \in \Gamma^*$ it is decidable in* $\mathsf{uAC}^0(\mathsf{NL}^{\mathrm{WP}(\mathbf{GP}\mathcal{C})})$ *if there exist reduced $p, q \in \Gamma^*$ such that $puq \equiv w$ in* $\mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ *for $\mathcal{L} = \{1, \ldots, n\}$.*

Proof. Let s be a fixed vertex in $D(u)$ and let s' be a vertex in $D(w)$. Assume that s' induces the subgraph D of the factor u and s' corresponds to s. Let t be any vertex in $D(u)$. We denote by $D(t', t)$ the predicate, which is true if the vertex t' in $D(w)$ corresponds to t in $D(u)$. By the above description, this predicate can be decided in NL with the oracle of the word problem to compare the labels.

Now, any vertex i in $D(w)$ belongs to D if there are vertices c, d in $D(u)$ and vertices c', d' in $D(w)$ such that $D(c, c')$ and $D(d, d')$ holds and there exists a $c' \xrightarrow{*} i$ path and an $i \xrightarrow{*} d'$ path. Let the factorization of w be $w = w_1 \cdots w_n$ and $u' = u'_1 \cdots u'_n$ be the word with

$$u'_i = \begin{cases} w_i, & \text{if } i \text{ belongs to } D \\ 1, & \text{otherwise.} \end{cases}$$

This projection can be computed in AC^0 with an oracle for the predicate $D(t, t')$. Finally, the factor induced by D equals u, if and only if $u =_G u'$. We denote the constructed circuit above by $factor(s, s', u, w)$. Let the factorization of u be $u = u_1 \cdots u_k$. We fix $s = u_1$ to be the first vertex in $D(u)$. Thus, the circuit

$$\bigvee_{s'=1}^{n} \lambda(s) =_G \lambda(s') \wedge factor(s, s', u, w)$$

decides the factor problem. □

Theorem 7.3.15. *Let $\mathcal{C}$ be a non-trivial class of f. g. groups. Then*

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

that is on input of groups $G_1, \ldots, G_n \in \mathcal{C}$, an independence relation I and words $u, v \in \Gamma^*$ the conjugacy problem of $u \sim v$ in $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ for $\mathcal{L} = \{1, \ldots, n\}$ can be solved in uniform AC^0 with oracle gates for $\mathsf{C}_=\mathsf{L}^{\mathrm{CP}(\mathcal{C})}$.

Proof. The word problem of $\mathrm{WP}(\mathbf{GP}\mathcal{C})$ is NL-hard by Corollary 5.6.10. Therefore any TC^0-computation can be transformed into a uniform AC^0-computation with oracle gates for $\mathrm{WP}(\mathbf{GP}\mathcal{C})$. We may assume, to have an oracle for $\mathrm{WP}(\mathbf{GP}\mathcal{C})$ since $\mathrm{WP}(\mathbf{GP}\mathcal{C}) \in \mathsf{C}_=\mathsf{L}^{\mathrm{WP}(\mathcal{C})}$ by Theorem 5.6.14 and $\mathrm{WP}(\mathcal{C})$ can be decided by using the oracle for $\mathrm{CP}(\mathcal{C})$. By Lemma 7.3.4 cyclically reduced words of u and v can be computed in AC^0 with oracle gates for the word problem. Thus, we may assume u and v to be cyclically reduced in the following. Let in the following $\alpha \in \mathcal{L}$ and let the predicate $con_w(i, \alpha)$ decide if a vertex i is connected to a vertex with label in Γ_α in the dependence graph $D(w)$. Any two vertices are connected if there is an undirected path in the dependence graph of length at most $|\mathcal{L}| - 1$. Hence, i is in the connected component of α if there is some vertex j with label in Γ_α and there is some undirected $i \xrightarrow{*} j$ path. Undirected path connectivity is in $\mathsf{LOGSPACE}$ by Reingold [Rei08] and hence $con_w(i, \alpha)$ is decidable in $\mathsf{C}_=\mathsf{L}$. Consider the projections $\pi_{w,\alpha}$ defined by

$$w_i \mapsto \begin{cases} w_i, & \text{if } con_w(i, \alpha) \text{ is true} \\ 1, & \text{otherwise.} \end{cases}$$

Let u_α (respectively v_α) be the projection of $\pi_{u,\alpha}$ applied to u (respectively of $\pi_{v,\alpha}$ applied to v). Then we have by Lemma 7.3.5 that $u \sim_G v$ if and only if $u_\alpha \sim_G v_\alpha$ for all $\alpha \in \mathcal{L}$. By construction, the words u_α and v_α are connected. Thus, the conjugacy problem reduces to connected words via

$$\bigwedge_{\alpha\in\mathcal{L}} u_\alpha \sim_G v_\alpha.$$

Hence, for the rest of the circuit construction we may assume that u and v are connected. For $|\mathrm{alph}(u)| > 1$ we have by Lemma 7.3.8 that $u \sim_G v$ if and only if $u \approx v$. By lemma Lemma 7.3.9 we have $u \approx v$ if and only if u is a factor of v^k and $|u|_\alpha = |v|_\alpha$. Let $factor(u, w)$ be the circuit from Lemma 7.3.14 to check, whether u is a factor of w. Then the circuit

$$\left(|u| = 1 \wedge |v| = 1 \wedge \left(\bigvee_{\alpha \in \mathcal{L}} \mathrm{alph}(u) = \mathrm{alph}(v) = \alpha \wedge u \sim_{G_\alpha} v \right) \right) \vee$$
$$\left(|u| > 1 \wedge |v| > 1 \wedge factor(u, v^k) \wedge \bigwedge_{\alpha \in \mathcal{L}} |u|_\alpha = |v|_\alpha \right)$$

decides the conjugacy problem $u \sim_G v$ with oracle gates for the conjugacy problem $\mathrm{CP}(\mathcal{C})$ and oracle gates for the word problem $\mathrm{WP}(\mathbf{GP}\mathcal{C})$. □

Corollary 7.3.16. *The uniform conjugacy problem of right angled Coxeter groups and graph groups is contained in* $\mathsf{uAC}^0(\mathsf{C}_=\mathsf{L})$.

Proof. The base groups of a graph group are $H = \langle a \rangle$ (respectively $H = \langle a \mid a^2 \rangle$ for right angled Coxeter groups). We have $x \sim_H y$ if and only if $x =_H y$. Moreover, the word problem of H is in $\mathsf{TC}^0 \subseteq \mathsf{LOGSPACE}$. Hence, the result follows with Theorem 7.3.15 and $\mathsf{C}_=\mathsf{L}^{\mathsf{LOGSPACE}} = \mathsf{C}_=\mathsf{L}$. □

Example 7.3.17. *The uniform conjugacy problem of graph products over finite groups is contained in* $\mathsf{uAC}^0(\mathsf{C}_=\mathsf{L})$*, when the finite groups are given by their multiplication table.*

Proof. By Example 5.1.4 the uniform word problem of finite groups given by their multiplication table is $\mathsf{LOGSPACE}$-complete. Since the group G is finite and given via its multiplication table, all candidates $z \in G$ for $zu\bar{z} =_G v$ can be checked in $\mathsf{LOGSPACE}^{\mathsf{LOGSPACE}}$. Hence, with $\mathsf{LOGSPACE}^{\mathsf{LOGSPACE}} = \mathsf{LOGSPACE}$ it follows that the uniform conjugacy problem of finite groups is $\mathsf{LOGSPACE}$-complete. Together with Theorem 7.3.15 we obtain that the uniform conjugacy problem of graph products over finite groups is in $\mathsf{uAC}^0(\mathsf{C}_=\mathsf{L}^{\mathsf{LOGSPACE}})$. Finally, with $\mathsf{C}_=\mathsf{L}^{\mathsf{LOGSPACE}} = \mathsf{C}_=\mathsf{L}$ the result follows. □

Chapter 8

Conclusion and Open Questions

We have considered the parallel complexity of three important problems in algorithmic group theory. The considered problems are in particular the word problem, the conjugacy problem and the geodesic and normal form problem. Thereby, the focus was on the direct product, the free product and the graph product of groups.

The main obstacle to solve these problems in parallel is the word problem since any solution of the other problems does lead to a solution of the word problem. For the word problem, the complexity of free products and graph products is different in general (unless some of the complexity classes between NC^1 and $\mathsf{C}_=\mathsf{L}$ collapse). As is turned out, the complexity is already different in the special case of free groups and graph groups. Both, free groups and graph groups have a word problem in $\mathsf{C}_=\mathsf{NC}^1$ for a fixed group. The known lower bound of NC^1 due to Robinson holds for both (except for graph groups that are actually free abelian). Moreover, the word problem of a fixed free group and a graph group (which is not free abelian) is equivalent under AC^0 Turing reductions. While for a fixed group there is no difference in the complexity, there clearly is one in the uniform variant. Free groups have a uniform word problem in $\mathsf{C}_=\mathsf{NC}^1$, whereas the uniform word problem of graph groups is NL-hard. For now, the complexity of the uniform word problem of graph groups is sandwiched between NL and $\mathsf{C}_=\mathsf{L}$. Any clarification on the precise complexity of the uniform word problem might help in the future to resolve the precise complexity of the word problem of a fixed graph group and vice versa.

Another viewpoint on the complexity of free groups and graph groups is the problem of identity testing for the product of a sequence of integer matrices. For $\mathrm{SL}(3,\mathbb{Z})$ this problem is known to be complete for $\mathsf{C}_=\mathsf{NC}^1$. However, for $\mathrm{SL}(2,\mathbb{Z})$ the precise complexity is unknown. It clearly is contained in $\mathsf{C}_=\mathsf{NC}^1$, but it might also be contained and therefore complete in NC^1. An affirmative answer to this question would also resolve the complexity of the word problem of the free group (see [AW16] for recent developments).

For the solution of the word problem of a fixed graph product and the solution of the uniform word problem different approaches were required. The solution

for a fixed graph product is by induction and considering the graph product as an amalgamated product. For the uniform word problem of graph products an embedding into the automorphism group over some infinite dimensional vector space was constructed.

For the geodesic problem an equivalence relation on words was introduced for free products and graph products. This relation does depend on the particular group and determining its equivalence classes requires a solution of the word problem. In case of direct products and free products a solution of the normal form problem is directly obtained by applying the normal form functions of the base groups to a geodesic. For graph products the normal form is the lexicographically first word among the geodesics. Obtaining the lexicographically first word is in TC^0 for a fixed graph group and FNL complete for the uniform problem.

Finally, the solutions of the geodesic problem lead to the solutions of the conjugacy problem. For free products the conjugacy problem reduces to computing cyclically reduced words and solving the transposition problem. For graph products the conjugacy problem reduces to computing cyclically reduced words and solving the transitive closure of the transposition problem. Solving the transitive closure of the transposition problem was then reduced to the pattern matching problem in traces. This pattern matching problem can be solved in AC^0 for a fixed graph group with oracles for the word problem and in NL with oracles for the word problem in the uniform case.

The open questions mentioned above concerning the word problem problem and the open questions, whether $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1) \stackrel{?}{=} \mathsf{C}_=\mathsf{NC}^1$ and $\mathsf{AC}^0(\mathsf{C}_=\mathsf{L}) \stackrel{?}{=} \mathsf{C}_=\mathsf{L}$ yield the following questions:

(1) Is the word problem of graph groups $\mathsf{C}_=\mathsf{NC}^1$-hard?

(2) Is the conjugacy problem of graph groups $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$-hard?

(3) Is the uniform word problem of graph groups $\mathsf{C}_=\mathsf{L}$-hard?

(4) Is the uniform conjugacy problem of graph groups $\mathsf{AC}^0(\mathsf{C}_=\mathsf{L})$-hard?

If any of these questions is answered positively, then the results in this thesis would imply completeness. In the uniform setting we may also ask

(5) Is the uniform word (respectively conjugacy) problem of graph groups NL-complete?

The word problem of a fixed graph group (respectively a fixed graph product over finite groups) is contained in $\mathsf{C}_=\mathsf{NC}^1$. However, the general construction in this thesis yields the slightly weaker result of $\mathsf{AC}^0(\mathsf{C}_=\mathsf{NC}^1)$. If $\mathsf{C}_=\mathsf{NC}^1$ is closed under AC^0 Turing reductions, then the results would be the same. However, since this is an open problem, the question for now is

(6) Is the word problem of any graph product $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ with $\mathrm{WP}(G_\alpha) \in \mathsf{NC}^1$ contained in $\mathsf{C}_=\mathsf{NC}^1$?

And in more general, this question is

(7) Is the word problem of any graph product over linear groups in $\mathsf{C}_=\mathsf{NC}^1$?

(8) Is the word problem of any graph product $G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$ with $\mathrm{WP}(G_\alpha) \in \mathsf{C}_=\mathsf{NC}^1$ contained in $\mathsf{C}_=\mathsf{NC}^1$?

If graph products of linear groups are linear again, then the answer to the first question would be positive. However, this is a long-standing open question [HW99]. We can transfer these questions to the uniform word problem.

(9) Is the uniform word problem of graph products over linear groups in $\mathsf{C}_=\mathsf{L}$?

(10) Is - for some class $\mathcal{C}$ of f. g. groups - the uniform word problem of graph products over $\mathcal{C}$ in $\mathsf{C}_=\mathsf{L}$ if we have $\mathrm{WP}(G_\alpha) \in \mathsf{C}_=\mathsf{L}$?

Results

The following tables summarize the main results presented in this thesis.

word problem of	non-uniform	uniform over $\mathcal{C}$
$G = \prod_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the word problem of each G_α	AC^0-reducible to the word problem of the class $\mathcal{C}$
$G = \ast_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the word problem of each G_α and F_2	AC^0-reducible to the word problem of the class $\mathcal{C}$ and F_2
$G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$	AC^0-reducible to the word problem of each G_α and F_2	NL-hard, contained in $\mathsf{C}_=\mathsf{L}$ with oracle for the word problem of the class $\mathcal{C}$

Figure 8.1: The word problem of certain group products.

conjugacy problem of	non-uniform	uniform over $\mathcal{C}$
$G = \prod_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the conjugacy problem of each G_α	AC^0-reducible to the conjugacy problem of the class $\mathcal{C}$
$G = \ast_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the conjugacy problem of each G_α and the word problem of F_2	AC^0-reducible to the conjugacy problem of the class $\mathcal{C}$ and the word problem of F_2
$G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$	AC^0-reducible to conjugacy problem of each G_α and the word problem of F_2	NL-hard, AC^0-reducible to $\mathsf{C}_=\mathsf{L}$ with oracle for the conjugacy problem of the class $\mathcal{C}$

Figure 8.2: The conjugacy problem of certain group products.

geodesic problem of	non-uniform	uniform over $\mathcal{C}$
$G = \prod_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the word problem of each G_α	TC^0-reducible to the word problem of the class $\mathcal{C}$
$G = \ast_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the word problem of G	AC^0-reducible to the word problem of the class $\mathbf{F}\mathcal{C}$
$G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$	AC^0-reducible to the word problem of G	NL-hard, AC^0-reducible to the word problem of the class $\mathbf{GP}\mathcal{C}$

Figure 8.3: The geodesic problem of certain group products.

normal form problem with input geodesic of	non-uniform	uniform over $\mathcal{C}$
$G = \prod_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the normal form problem of each G_α	AC^0-reducible to the normal form problem of the class $\mathcal{C}$
$G = \ast_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the normal form problem of each G_α	AC^0-reducible to the normal form problem of the class $\mathcal{C}$
$G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$	in general TC^0-hard, TC^0-reducible to the normal form problem of each G_α	FNL-hard, contained in FNL with oracle for the normal form problem of the class $\mathcal{C}$

Figure 8.4: The normal form problem for geodesics of certain group products.

normal form problem of	non-uniform	uniform over $\mathcal{C}$
$G = \prod_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the normal form problem of each G_α	AC^0-reducible to the normal form problem of the class $\mathcal{C}$
$G = \ast_{\alpha\in\mathcal{L}} G_\alpha$	AC^0-reducible to the word problem of G and normal form problem of each G_α	AC^0-reducible to the word problem of the class $\mathbf{F}\mathcal{C}$ and normal form problem of the class $\mathcal{C}$
$G = \mathrm{GP}(\mathcal{L}, I; (G_\alpha)_{\alpha\in\mathcal{L}})$	AC^0-reducible to the word problem of G and normal form problem of each G_α	FNL-hard, AC^0-reducible to $\mathsf{C}_=\mathsf{L}$ with oracle for the normal form problem of the class $\mathcal{C}$

Figure 8.5: The normal form problem of certain group products.

Bibliography

[ABJ95] C. Àlvarez, J.L. Balcázar, and B. Jenner. Adaptive logspace reducibility and parallel time. *Mathematical systems theory*, 28(2):117–140, 1995.

[ABO99] E. Allender, R. Beals, and M. Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8(2):99–126, 1999.

[ÀG91] Carme Àlvarez and Joaquim Gabarró. The parallel complexity of two problems on concurrency. *Information Processing Letters*, 38:61–70, 1991.

[AK79] Anatolij V. Anisimov and Donald E. Knuth. Inhomogeneous sorting. *International Journal of Computer and Information Sciences*, 8:255–260, 1979.

[All04] Eric Allender. Arithmetic circuits and counting complexity classes. In *Complexity of Computations and Proofs, Quaderni di Matematica*, pages 33–72, 2004.

[AO94] E. Allender and M. Ogihara. Relationships among PL, #L, and the determinant. In *Structure in Complexity Theory Conference, 1994., Proceedings of the Ninth Annual*, pages 267–278, Jun 1994.

[AW16] Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *computational complexity*, 25(1):217–253, 2016.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[BB05] Anders Björner and Francesco Brenti. *Combinatorics of Coxeter groups*, volume 231 of *Graduate Texts in Mathematics*. Springer, New York, 2005.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274 – 306, 1990.

[CGW09] John Crisp, Eddy Godelle, and Bert Wiest. The conjugacy problem in right-angled Artin groups and their subgroups. *Journal of Topology*, 2(3):442–460, 2009.

[Cha07] Ruth Charney. An introduction to right-angled Artin groups. *Geometriae Dedicata*, 125(1):141–158, 2007.

[CM87] Stephen A. Cook and Pierre McKenzie. Problems complete for deterministic logarithmic space. *Journal of Algorithms*, 8(3):385 – 394, 1987.

[CMTV98] Hervé Caussinus, Pierre McKenzie, Denis Thérien, and Heribert Vollmer. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences*, 57(2):200 – 212, 1998.

[Coo85] Steven A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[Dam91] C. Damm. $DET = L^{\#L}$. Informatik-Preprint 8. *Fachbereich Informatik der Humboldt-Universitat zu, Berlin*, 1991.

[Die90] Volker Diekert. *Combinatorics on Traces*, volume 454 of *Lecture Notes in Computer Science*. Springer-Verlag, Heidelberg, 1990.

[DK14] Volker Diekert and Jonathan Kausch. Logspace computations in graph products. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *ISSAC*, pages 138–145. ACM, 2014.

[DK16] Volker Diekert and Jonathan Kausch. Logspace computations in graph products. *Journal of Symbolic Computation*, 75(C):94–109, July 2016.

[DKL12a] Volker Diekert, Jonathan Kausch, and Markus Lohrey. Logspace computations in Coxeter groups and graph groups. In *Computational and Combinatorial Group Theory and Cryptography*, volume 582 of *Contemporary Mathematics*, pages 77–94. Amer. Math. Soc., 2012. Journal version of LATIN 2012, 243–254, LNCS 7256, 2012.

[DKL12b] Volker Diekert, Jonathan Kausch, and Markus Lohrey. Logspace computations in graph groups and Coxeter groups. In D. Fernãndez-Baca, editor, *Proceedings of LATIN 2012*, volume 7256 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2012.

[DMR+12] Samir Datta, Meena Mahajan, B.V. Raghavendra Rao, Michael Thomas, and Heribert Vollmer. Counting classes and the fine structure between NC^1 and L. *Theoretical Computer Science*, 417:36 – 49, 2012.

[DR95] Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.

[Dro87] Carl Droms. Graph groups, coherence, and three-manifolds. *Journal of Algebra*, 106(2):484 – 489, 1987.

[Dub86] Christine Duboc. On some equations in free partially commutative monoids. *Theoretical Computer Science*, 46:159–174, 1986.

[Ebi95] Werner Ebinger. Logical definability of trace languages. In V. Diekert and G. Rozenberg, editors, *The Book of Traces*, chapter 10, pages 382–390. World Scientific, Singapore, 1995.

[EEO13] Murray Elder, Gillian Elston, and Gretchen Ostheimer. On groups that have normal forms computable in logspace. *Journal of Algebra*, 381:260 – 281, 2013.

[Gre90] Elisabeth R. Green. *Graph Products of Groups*. PhD thesis, The University of Leeds, 1990.

[Hes01] William Hesse. Division is in uniform TC^0. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *International Colloquium Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 104–114. Springer, 2001.

[HM95] Susan Hermiller and John Meier. Algorithms and geometry for graph products of groups. *Journal of Algebra*, 171(1):230–257, 1995.

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara. *The complexity theory companion*. Springer, Berlin; Heidelberg [u.a.], 2002.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[HW99] Tim Hsu and Daniel T. Wise. On linear and residual properties of graph products. *Michigan Mathematical Journal*, 46(2):251–259, 1999.

[HY92] Kosaburo Hashiguchi and Kazuya Yamada. String matching problems over free partially commutative monoids. *Information and Computation*, 101:131–149, 1992.

[Jon75] Neil D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68 – 85, 1975.

[Jun85] Hermann Jung. Depth efficient transformations of arithmetic into boolean circuits. In Lothar Budach, editor, *Fundamentals of Computation Theory*, volume 199 of *Lecture Notes in Computer Science*, pages 167–174. Springer Berlin Heidelberg, 1985.

[Kel73] Robert M. Keller. Parallel program schemata and maximal parallelism I. Fundamental results. *Journal of the ACM*, 20(3):514–537, 1973.

[KL06] Dietrich Kuske and Markus Lohrey. Logical Aspects of Cayley-Graphs: The Monoid Case. *International Journal of Algebra and Computation*, 16:307–340, 2006.

[LS01] Roger Lyndon and Paul Schupp. *Combinatorial Group Theory*. Classics in Mathematics. Springer, 2001. First edition 1977.

[LWZ90] Hai-Ning Liu, Celia Wrathall, and Kenneth Zeger. Efficient solution of some problems in free partially commutative monoids. *Information and Computation*, 89:180–198, 1990.

[LZ77] Richard J. Lipton and Yechezkel Zalcstein. Word problems solvable in logspace. *Journal of the ACM*, 24:522–526, 1977.

[Mar85] Zbigniew S. Marciniak. A note on free products of linear groups. *Proceedings of the American Mathematical Society*, 94(1):46–48, 1985.

[Maz77] Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

[Maz87] Antoni Mazurkiewicz. Trace theory. In W. Brauer et al., editors, *Petri Nets, Applications and Relationship to other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 279–324, Heidelberg, 1987. Springer-Verlag.

[Mih58] K. A. Mihailova. The occurrence problem for direct products of groups. *Dokl. Akad. Nauk SSSR*, 119:1103–1105, 1958. English translation in: Math. USSR Sbornik, 70: 241–251, 1966.

[Mil92] Charles F. Miller III. Decision problems for groups – survey and reflections. In *Algorithms and Classification in Combinatorial Group Theory*, pages 1–60. Springer, 1992.

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago Journal of Theoretical Computer Science*, 1997.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):17:1–17:24, September 2008.

[Rob93] David Robinson. *Parallel Algorithms for Group Word Problems*. PhD thesis, University of California, San Diego, 1993.

[RST84] Walter L. Ruzzo, Janos Simon, and Martin Tompa. Space-bounded hierarchies and probabilistic computations. *Journal of Computer and System Sciences*, 28(2):216 – 230, 1984.

[San47] I. N. Sanov. A property of a representation of a free group. volume 57 of *Dokl. Akad. Nauk. SSSR*, pages 657–659, 1947.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177 – 192, 1970.

[Ser89] H. Servatius. Automorphisms of graph groups. *Journal of Algebra*, 126(1):34–60, 1989.

[Sho89] Takayoshi Shoudai. The lexicographically first topological order problem is NLOG-complete. *Information Processing Letters*, 33(3):121 – 124, 1989.

[Sim79] Hans-Ulrich Simon. Word problems for groups and contextfree recognition. In *Proceedings of Fundamentals of Computation Theory (FCT'79), Berlin/Wendisch-Rietz (GDR)*, pages 417–422. Akademie-Verlag, 1979.

[ST98] M. Santha and S. Tan. Verifying the determinant in parallel. *Computational Complexity*, 7(2):128–151, 1998.

[Tod91] Seinosuke Toda. Counting problems computationally equivalent to computing the determinant, 1991.

[Tod92] Seinosuke Toda. Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE Transactions on Information and Systems*, 75(1):116–124, 1992.

[Vin91a] V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Structure in Complexity Theory Conference, 1991., Proceedings of the Sixth Annual*, pages 270–284, Jun 1991.

[Vin91b] V. Vinay. *Semi-unboundedness and complexity classes*. PhD thesis, PhD thesis, Indian Institute of Science, Bangalore, 1991.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity*. Springer, Berlin, 1999.

[Waa90] Stephan Waack. The parallel complexity of some constructions in combinatorial group theory. *Journal of Information Processing and Cybernetics*, 26(5-6):265–281, 1990.

[Weh73] B. A. F. Wehrfritz. Generalized free products of linear groups. *Proceedings of the London Mathematical Society*, 27:402–424, 1973.

[Wei16] Armin Weiß. A logspace solution to the word and conjugacy problem of generalized Baumslag-Solitar groups. *ArXiv e-prints*, abs/1602.02445, 2016.

Index