# Logspace computations in graph products

Volker Diekert and Jonathan Kausch

FMI, Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart, Germany
{diekert,kausch}@fmi.uni-stuttgart.de

**Abstract.** We consider three important and well-studied algorithmic problems in group theory: the word, geodesic, and conjugacy problem. We show transfer results from individual groups to graph products. We concentrate on logspace complexity because the challenge is actually in small complexity classes, only. The most difficult transfer result is for the conjugacy problem. We have a general result for graph products, but even in the special case of a graph group the result is new. Graph groups are closely linked to the theory of Mazurkiewicz traces which form an algebraic model for concurrent processes. Our proofs are combinatorial and based on well-known concepts in trace theory. We also use rewriting techniques over traces. For the group-theoretical part we apply Bass-Serre theory. But as we need explicit formulae and as we design concrete algorithms all our group-theoretical calculations are completely explicit and accessible to non-specialists.

## 1 Introduction

**Background.** Algorithmic questions concerning finitely generated groups have been studied for more than 100 years starting with the fundamental work of Tietze and Dehn in the beginning of the 20th century. In this paper we investigate three algorithmic problems for graph products $G$ with a finite and symmetric generating set $\Sigma$. The question for us is whether they can be decided in logspace.

1. **Word problem**. Let $w \in \Sigma^*$. Is $w = 1$ in the group $G$?
2. **Geodesic problem**. Let $w \in \Sigma^*$. Compute a geodesic, i.e., a shortest word representing $w \in G$; and, if a linear order on $\Sigma$ is defined, compute the lexicographical first word among all geodesics, i.e., compute a shortlex normal form of $w$.
3. **Conjugacy problem**. Let $u, v \in \Sigma^*$. Are $u$ and $v$ conjugated in $G$?

The complexity of the first and third problem depends on $G$ only, whereas for the second problem we have to specify $\Sigma \subseteq G$, too. Over the past few decades the search and design of algorithms for decision problems like the ones above has developed into an active research area, where algebraic methods and computer science techniques join in a fruitful way, see e.g. the recent surveys [20,28]. Of particular interest are those problems which can be solved efficiently in parallel.

More precisely, we are interested in *deterministic logspace*, called simply *logspace* in the following. This is a complexity class at the lower level in the NC-hierarchy[1]:

$$\mathsf{NC}^1 \subseteq \mathsf{logspace} \subseteq \mathsf{LOGCFL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} = \bigcup_{i \geq 1} \mathsf{NC}^i \subseteq \mathsf{P} \subseteq \mathsf{NP} \quad (1)$$

No separation result between $\mathsf{NC}^1$ and $\mathsf{NP}$ is known but it is believed (by some) that all of the above inclusions in (1) are strict. A fundamental result in the context of group-theoretical algorithms was shown by Lipton, Zalcstein and Simon in [18,30]: The word problem of finitely generated linear groups belongs to logspace. The class of groups with a word problem in logspace is further investigated in [32]. Another important result due to Cai (resp. Lohrey) is that the word problem of hyperbolic groups is in $\mathsf{NC}^2$ [2] (resp. in $\mathsf{LOGCFL}$ by [19]). The class $\mathsf{LOGCFL}$ coincides with the (uniform) class $\mathsf{SAC}^1$. It is a subclass of $\mathsf{NC}^2$. Often, it is not enough to solve the word problem, but one has to compute a normal form. This leads to the problem of computing geodesics. This problem and various related problems were studied e.g. in [10,12,14,24,26]. These results imply that there are groups with an easy word problem (in logspace), but where simple questions related to geodesics are computationally hard, for example $\mathsf{NP}$-complete for certain wreath products or free metabelian group of rank 2. Finally, the conjugacy problem is a classical decision problem which is notoriously more difficult than the word problem. Whereas for a wide range of groups the word problem is decidable (and often easily decidable) the conjugacy problem is not known to be decidable. This includes e.g. automatic groups (word problem is in $\mathcal{O}(n^2)$) or one-relator groups (word problem is decidable) to mention two classes. Miller's group [22] has a decidable word problem (at most cubic time[2], actually logspace) but undecidable conjugacy problem. Actually, there are finitely generated subgroups of $F_2 \times F_2$ (hence subgroups of $\mathrm{SL}(4, \mathbb{Z})$, hence linear groups) with unsolvable conjugacy problem [23, Thm. 5.2].

Here, we continue and generalize the work of [5] from graph groups to graph products of groups having a word problem in logspace. We show transfer results for all three problems mentioned above. However, techniques of [5] for graph groups (which used linear representations for right-angled Coxeter groups) are not available in the present paper, simply because linear representations do not exist for the individual groups, in general. For graph products we start with a list $L$ of groups $G_\alpha$. Next, we endow $L$ with an irreflexive and symmetric relation $I \subseteq L \times L$. This means $(L, I)$ is a finite undirected graph and each node $\alpha \in L$ is associated with a node group $G_\alpha$. The graph product $G$ is then the free product of the $G_\alpha$'s modulo defining relations $gh = hg$ for all $g \in G_\alpha$ and $h \in G_\beta$ where $(\alpha, \beta) \in I$. Thus, it is a free product with partial commutation. If $I$ is empty then $G$ is the free product $\star_{\alpha \in L} G_\alpha$. If $(L, I)$ is a complete graph then $G$ is the direct product $\prod_{\alpha \in L} G_\alpha$. Our setting includes the important special case where all node groups are isomorphic to $\mathbb{Z}$. This is exactly the case when $G$ is *free partially*

[1] $\mathsf{NC}^i$ is the class of languages which are accepted by (uniform) boolean circuits of polynomial size, depth $O(\log^i(n))$ and constant fan-in, see e.g. [31] for a textboook.

[2] Mark Sapir, personal communication

*commutative*. These groups are also known as *graph groups* (see [9]) or *right-angled Artin groups* (RAAGs). If all node groups are isomorphic to $\mathbb{Z}/2\mathbb{Z}$ then we obtain a *right-angled Coxeter group*. Graph groups embed into right-angled Coxeter groups in a canonical way [16], and Coxeter groups are known to be linear. Hence, graph groups and Coxeter groups have a word problem in logspace. Graph groups received in recent years a lot of attention in group theory because of their rich subgroup structure [1,4]. On the algorithmic side, (un)decidability results were obtained for many important decision problems in graph groups [3,7]. The theory of free partially commutative groups is also directly linked to the theory of Mazurkiewicz traces which is important in computer science since it yields an algebraic framework for concurrent systems [17,21,8].

**Results.** Our achievement has a strikingly simple formulation: If the word problem (geodesic problem resp., conjugacy problem resp.) of all node groups is in logspace then the same is true for the graph product.

An analogous assertion holds for various other complexity classes closed under logspace reductions like $\mathsf{NC}, \mathsf{P}$ or $\mathsf{NP}$ by similar arguments as used in this paper. We treat "logspace" because it concerns the smallest natural complexity class where we can assert such a statement because the word problem of non-abelian free groups has to be solved, which is $\mathsf{NC}^1$-hard by [27]; and the best known upper bound is logspace. So, it is possible that logspace is truly the smallest class in all non-trivial cases.

Our results with respect to the word problem generalize in particular [13, Prop. 19] solving thereby an open problem. A transfer result with respect to the word problem was known before for free products [32], but unknown for graph products, in general. For a compressed variant of the word problem, it is known that polynomial time decidability is preserved by graph products [15]. Our results here imply that the word problem of a graph products of linear groups is solvable in logspace. It is still open whether a graph product of linear groups is linear again. The results here support a positive answer to this question asked in [16]; but not much beyond the classical result of [33] is known. Our method also yields a logspace-reduction of the conjugacy problem for graph products of linear groups to the conjugacy problem in the node groups. This is somewhat the best we can expect because, as we mentioned above, there are finitely generated linear groups with unsolvable conjugacy problem.

Our proof is inductive on the number of nodes $\alpha \in L$ and the algebraic description of a graph product as a certain amalgamated product. Amalgamated products are basic components in Bass-Serre theory[3] [29]; and indeed, our proof of Corollary 4 is an application of explicit Bass-Serre theory. The proof is still technical and not easy. On the positive side we had to make all calculations explicit. Thus, no a priori knowledge in Bass-Serre theory is necessary for understanding the logspace solution of the word problem in graph products.

[3] Bass-Serre theory is a cornerstone in modern combinatorial group theory. It showed us the direction to the proof, but the abstract theory does not give complexity results, directly.

## 2 Notation

**Words.** An *alphabet* is a set (with a linear order) and its elements are called *letters*. By $\Sigma^*$ we denote the free monoid over $\Sigma$ and its elements are called *words*. For a word $w \in \Sigma^*$ we denote by $|w|$ its *length* and if $a \in \Sigma$, then $|w|_a$ counts how often the letter $a$ appears in $w$. Thus, $|w| = \sum_{a\in\Sigma} |w|_a$. By $\text{alph}(w) = \{a \in \Sigma \mid |w|_a \geq 1\}$ we denote the *alphabet* of $w$. The *empty word* has length 0; and it is denoted by 1 as other neutral elements in monoids or groups. In the paper, $L$ is a finite list (with a linear order) and $\Sigma$, $\Sigma_\alpha$, $\Gamma$, $\Gamma_\alpha$ denote alphabets. We have $\Sigma_\alpha \subseteq \Gamma_\alpha$, $\Sigma = \bigcup_{\alpha\in L} \Sigma_\alpha$ and $\Sigma$ is finite, $\Gamma = \bigcup_{\alpha\in L} \Gamma_\alpha$ and $\Gamma$ is typically infinite. All alphabets are endowed with an *involution*. This is a mapping $x \mapsto \overline{x}$ such that $\overline{\overline{x}} = x$. The involution is extended to words by $\overline{a_1 \cdots a_n} = \overline{a_n} \cdots \overline{a_1}$ where $a_i$ are letters. For a group $G$ the involution is here always defined by taking the inverse, i.e., $\overline{g} = g^{-1}$ for $g \in G$. As we represent group elements by words, we prefer the notation $\overline{g}$ rather than $g^{-1}$ for group elements, too.

**Groups.** Our frame is given by groups $G_\alpha$ (for $\alpha$ in the list $L$) which are assumed to be generated by some finite subset $\Sigma_\alpha$ with $\Sigma_\alpha = \Sigma_\alpha^{-1} \subseteq G_\alpha \setminus \{1\}$. Moreover we define an alphabet $\Gamma_\alpha = G_\alpha \setminus \{1\}$. Hence $\Gamma_\alpha$ is infinite, in general. This means there is a natural surjective homomorphism from $\Sigma_\alpha^*$ onto $G_\alpha$ which respects the involution. Moreover, every letter $a \in \Gamma_\alpha$ can be represented by a word $w_a \in \Sigma_\alpha^*$ such that $w_a = a$ in $G_\alpha$.

**Graphs.** Here, graphs are without self-loops and multiple edges. They are node-labeled. The undirected graphs specify the "independence" relation. Directed graphs specify "dependence graphs" which are used to represent group elements in graph products. We say that graphs are *identical*, if they are isomorphic as node-labeled (directed) graphs. Thus, graphs are viewed as "abstract graphs".

**Complexity.** We use standard notation from complexity theory, [25,31]. In particular, we use the result that the composition of logspace computable functions is logspace computable. A function $f$ is computable in logspace if it is computable by some deterministic Turing machine such that the work tape is bounded by $\mathcal{O}(\log n)$ where $n$ denotes the input length. The output length is then bounded by some polynomial in $n$ and every logspace-computable function is computable in $\mathsf{P}$, i.e. deterministic polynomial time.

## 3 Word problem in certain amalgamated products

Our results concern graph products. The results in this section serve as a tool during an induction process. They are slightly more general than needed there. The situation in this section is as follows. We consider finitely generated groups $A, B, P$ such that $A \leq P$ is a subgroup of $P$ and we let $G = P \star_A (A \times B)$ be the amalgamated product over $A$. The identity on $P$ and the projection of $A \times B$ onto $A$ induce a projection $\pi : G \to P$. Let $H$ be the kernel of $\pi$, then we have a short exact sequence

$$1 \to H \to G \xrightarrow{\pi} P \to 1.$$

Moreover, since $\pi(a) = a$ for $a \in A$, the homomorphism $\pi$ and the identity on $A \times B$ induce a homomorphism of $G$ onto $P \times B$. Thus, a necessary condition for an element to be 1 in $G$ is that its image in $P \times B$ is 1.

**Theorem 1.** *Assume that the word problem of $P$ and $B$ can be solved in logspace and that the membership problem for $A$ in $P$ can be solved in logspace, too. Then the word problem of $G$ can be solved in logspace.*

**Proof.** Let $w = g_0 b_1 g_1 \cdots b_m g_m$ be a word with $g_i \in P$ and $b_j \in B$. We want to decide whether $w = 1 \in G$. In a first step, we simply compute $\pi(w) = g_0 g_1 \cdots g_m \in P$ and we check whether $\pi(w) = 1$. This can be done in logspace and for the rest of the proof we may assume $\pi(w) = 1$ (because otherwise $w \neq 1 \in G$) and hence we have $w \in H$. Moreover, we may also assume that $b_1 \cdots b_m = 1 \in B$.

The structure of $H$ is well understood by Bass-Serre theory. The group $H$ is a free product of groups $B^p = pBp^{-1}$ for certain $p \in P$. For those readers who are familiar with Bass-Serre theory let us note that $H$ is the fundamental group of a graph of groups which is a "star" with a trivial group in the center:

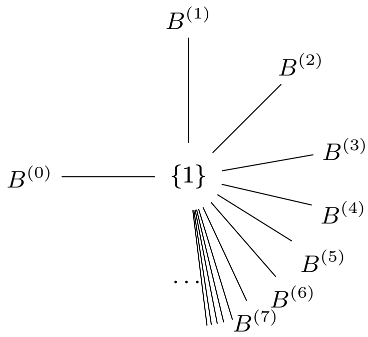



**Fig. 1.** Star with $[P : A]$ rays, because $P \leq G$ induces a bijection $P/A = H \backslash G/(A \times B)$.

However, knowing the structure of $H$ is not enough, since we must be able to compute in logspace an effective representation of $w$ in the free product. Moreover, $H$ is not finitely generated, in general. (This happens if $B$ is non-trivial and the index $[P : A]$ is infinite, which is the case of interest). So, instead of using Bass-Serre theory as a black box, we take a more elementary approach.

Let $N$ be an index set and $\{p_\nu \in P \mid \nu \in N\}$ be a subset of $P$ such that $\overline{p_\mu} p_\nu \notin A$ for all $\mu \neq \nu$. This means $\{p_\nu A \mid \nu \in N\}$ is a set of pairwise disjoint cosets. For each $\nu \in N$ we let $B^{(\nu)}$ be a copy of the group $B$. For $b^{(\nu)} \in B^{(\nu)}$ we let $b$ be the corresponding group element in $B$. Let $\psi_\nu : B^{(\nu)} \to H$ be the injective homomorphism defined by $\psi_\nu(b^{(\nu)}) = \overline{p_\nu} b p_\nu$. This induces a homomorphism

$\psi : \star_{\nu \in N} B^{(\nu)} \to H$. Now, we have $H \leq G$ and since $\overline{p_\mu} p_\nu \notin A$ for all $\mu \neq \nu$, a standard argument for amalgamated products shows that $\psi$ is injective.

Remember that we have $w = g_0 b_1 g_1 \cdots b_m g_m \in H$ with $g_i \in P$ and $b_j \in B$. Define and compute $p_i = g_0 \cdots g_i \in P$ for $0 \leq i < m$. Then we have $w = p_0 b_1 \overline{p_0} p_1 b_2 \overline{p_1} \cdots p_{m-1} b_m \overline{p_{m-1}}$ because $\overline{p_{m-1}} = g_m$. Thus, we can compute in logspace for each $i$ the minimal index $\nu(i) \in \{0, \ldots, m-1\}$ such that $\overline{p_{\nu(i)}}\, p_i \in A$. Here we use that the membership problem for $A$ in $P$ is computable in logspace. Define $N = \{\nu(i) \mid 0 \leq i < m\}$ and the homomorphism $\psi_\nu : B^{(\nu)} \to H$ as above. We obtain[4] $w = \psi(b_1^{(\nu_1)} \cdots b_m^{(\nu_m)})$ with $\nu_i = \nu(i)$.

Next, consider the homomorphism $\varphi : \star_{\nu \in N} B^{(\nu)} \to B$ where $\varphi(b^{(\nu)}) = b$. We have $\varphi(\psi^{-1}(w)) = \varphi(b_1^{(\nu_1)} \cdots b_m^{(\nu_m)}) = b_1 \cdots b_m \in B$. Recall that $b_1 \cdots b_m = 1$, hence $\psi^{-1}(w) \in K$, where $K = \ker(\varphi)$ denotes the kernel of $\varphi$. Bass-Serre theory tells us that $K$ is free, but we need to find and rewrite $\psi^{-1}(w)$ in some basis $X$ of some finitely generated free subgroup $F(X) \leq K$, so we do the explicit calculations.

For simplicity of notation we may assume that the input word $w$ is written as $w = b_1^{(\nu_1)} \cdots b_m^{(\nu_m)} \in K$ with $m \geq 1$ and $1 \neq b_i \in B^{(\nu(i))}$. Since $w \in K$, we have $m \geq 2$. We mimic what we have done above; and we define $g_i^{(\ell)} = (b_1 \cdots b_i)^{(\ell)} \in B^{(\ell)}$. In particular, $g_1^{(\ell)} = b_1^{(\ell)}$ and $g_m^{(\ell)} = 1$. For each $1 \leq i < m$, consider the factor $b_i^{(k)} b_{i+1}^{(\ell)}$ of $w$ with $k = \nu_i$ and $\ell = \nu_{i+1}$. Replace $b_i^{(k)} b_{i+1}^{(\ell)}$ by

$$b_i^{(k)}\, (\overline{b_i}^{(\ell)} \cdots \overline{b_1}^{(\ell)})(b_1{}^{(\ell)} \cdots b_i{}^{(\ell)})\, b_{i+1}^{(\ell)} = b_i^{(k)}\, \overline{g_i}^{(\ell)} g_{i+1}^{(\ell)}.$$

The input word $w$ becomes (after this logspace procedure) a word of the form $w = g_1^{(\nu_1)} \overline{g_1}^{(\nu_2)}\, g_2^{(\nu_2)} \overline{g_2}^{(\nu_3)} \cdots g_{m-1}^{(\nu_{m-1})} \overline{g_{m-1}}^{(\nu_m)}$ with $g_i^{(\nu_i)} \overline{g_i}^{(\nu_{i+1})} \in K$. We use the notation $(i, g, j) = g^{(i)} \overline{g}^{(j)} \in K$ and we rewrite $w$ as a product over these triples with $1 \leq i, j < m$ and $g \in B$. The triples define a subset of $K$ of size less than $m^3$. We have $(i, g, j)^{-1} = (j, g, i) \in K$. But the set of $(i, g, j)$ is not a basis of $K$ since e.g., $(i, g, k)(k, g, j) = (i, g, j)$. In particular, we have $(i, g, j) = (i, g, 0)(0, g, j)$ for all $i, j$. The next logspace computation rewrites $w$ as a product in triples $(i, g, j) = g^{(i)} \overline{g}^{(j)}$ such that $1 \neq g \in B$ and $i \neq j$, $g^{(i)} \in B^{(i)}$ and $\overline{g}^{(j)} \in B^{(j)}$, $\varphi(g^{(i)}) = g$ and $\varphi(\overline{g}^{(j)}) = g^{-1}$. Thus, we find in logspace a smallest set $X = \{(i, g, 0) \mid i \neq 0, g \neq 1\}$ and a word $u \in (X \cup \overline{X})^*$ such that $w = \xi(u)$ for the mapping $\xi : X \to K$, $\xi(i, g, 0) = g^{(i)} \overline{g}^{(0)}$. The set $X$ has at most $m^2$ elements and the word $u$ can be viewed as an element in the free group $F(X)$. Standard logspace-computable encodings embed $F(X)$ into the free group $F(x, y)$ with two generators. For example the $i$-th generator in $x$ can be mapped to $x^i y x^{-i}$. Another standard logspace-computable encoding embeds $F(x, y)$ into the special linear group $\mathrm{SL}(2, \mathbb{Z})$ of $2 \times 2$ integer matrices[5]. We can evaluate the

[4] What we have done so far is "essentially" a logspace reduction from the word problem in $G$ to the word problem in some free product $\star_{\nu \in N} B^{(\nu)}$. It is not a logspace reduction in the literal sense, because $N$ depends on the input word, but on the positive side $|N| \leq m$.

[5] For example, $x \mapsto \begin{pmatrix} 0 & 1 \\ -1 & -2 \end{pmatrix}$ and $y \mapsto \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix}$.

matrix corresponding to $u$ in logspace by the Chinese remainder theorem. If the evaluation is the identity matrix then we have $w = 1$ (see [18] for details). Thus, we may assume that the matrix is not the identity. Hence, $1 \neq u \in F(X)$. But this is not enough, we have to show that $1 \neq u \in F(X)$ implies $w \neq 1$. The assertion $\xi(u) = w \neq 1$ follows from the following lemma. Thus, this lemma finishes the proof of Theorem 1. □

**Lemma** *The homomorphism $\xi : F(X) \to K$ is injective. This means $X$ forms a basis of a free subgroup of $K$ containing the element $w$.*

**Proof.** We know $\xi(u) = w$. Consider now any non-empty freely reduced word $u$ in $(X \cup \overline{X})^*$ and let $\xi(u)$ be its image in $K$. We have to show that $\xi(u) \neq 1$. We can write $u = v\,(i, g, j)$, where $v \in (X \cup \overline{X})^*$ is a freely reduced word and $(i, g, j) \in (X \cup \overline{X})$. We show:

– The last factor of $\xi(u)$ in the free product $\star_{\nu \in N} B^{(\nu)}$ is $\overline{g}^{(j)}$.
– If $j = 0$, then the last two factors of $\xi(u)$ are $h^{(i)}\overline{g}^{(0)}$ for some $h \in B$.

For $u = (i, g, j)$ we have $\xi(u) = g^{(i)}\overline{g}^{(j)}$ as desired. Hence, $v$ is not empty and we can write $u = v'(k, f, \ell)(i, g, j)$. By induction the last factor of $\xi(v)$ is $\overline{f}^{(\ell)}$.

For $\ell \neq i$ we conclude that the last three factors of $\xi(u)$ are $\overline{f}^{(\ell)} g^{(i)}\overline{g}^{(j)}$. Hence, we may assume that $\ell = i$. Therefore $u = v'(k, f, i)(i, g, j)$.

For $f \neq g$ the last two factors of $\xi(u)$ are $(\overline{f}g)^{(i)}\overline{g}^{(j)}$.

Now, assume $f = g$, then we must have $k \neq j$ because $u$ is freely reduced. But then we must have $i = \ell = 0$. Therefore, $u = v'(k, g, 0)(0, g, j)$ with $k \neq j$. By induction, the last two factors of $\xi(v)$ are $h^{(k)}\overline{g}^{(0)}$. Hence, the last two factors of $\xi(u)$ are $h^{(k)}\overline{g}^{(j)}$. In particular, $\xi(u) \neq 1$. □

## 4 Graph products

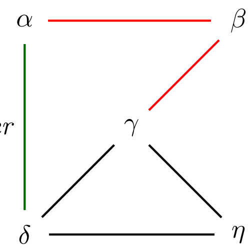
A graph product over groups is defined by the following data. There is a finite list $L$ and for each $\alpha \in L$ there is an associated finitely generated non-trivial group $G_\alpha$. In addition, there is an irreflexive symmetric relation $I \subseteq L \times L$, which is called an *independence relation.* This means, $(L, I; (G_\alpha)_{\alpha \in L})$ is a node-labeled undirected graph. The *graph product* $G = G(L, I; (G_\alpha)_{\alpha \in L})$ is defined as the quotient group of the free product $\star_{\alpha \in L} G_\alpha$ with defining relations $g_\alpha h_\beta = h_\beta g_\alpha$ for all $g_\alpha \in G_\alpha$, $h_\beta \in G_\beta$, $(\alpha, \beta) \in I$. If all $G_\alpha$ are finitely presented, then the graph product $G$ is finitely presented, too. If the independence relation $I$ is empty, then $G$ is a free product. If $(L, I)$ is a complete graph then $G$ is a direct product. The *universal property* of $G$ is that a homomorphism of $G$ to another group $G'$ is given by a family of homomorphisms $h_\alpha : G_\alpha \to G'$ such that $h_\alpha(x)h_\beta(y) = h_\beta(y)h_\alpha(x)$ for all $(x, y) \in G_\alpha \times G_\beta$ where $(\alpha, \beta) \in I$.

Graph products have an algebraic decomposition as in Section 3. Start with any "base" node $\beta \in L$ and let $B = G_\beta$. Consider the subgraph $(L', I')$ which is induced by $L \setminus \{\beta\}$. This yields a corresponding graph product $P$. The *link* of $\beta$ is the subgraph which is induced by the set of nodes $\alpha \in L'$ where $(\alpha, \beta) \in I$. Let

$A$ be the graph product corresponding to the link of $\beta$. Then $A$ is a subgroup of $P$ and $A \times B$ is a subgroup of $G$.

**Example 2** *Consider a graph product $G$ as depicted as follows.*

$G =$ *graph product over*




*The link of $\beta$ is $\{\alpha, \gamma\}$ and $A$ is the free product $A = G_\alpha \star G_\gamma$. Removing the node $\beta$ we obtain a smaller graph and the link of $\alpha$ becomes the singleton $\{\delta\}$. Removing $\alpha$ leaves us with a triangle with nodes $\gamma$, $\delta$, $\eta$ which yields the direct product $G_\gamma \times G_\delta \times G_\eta$. Going backwards we see that $P$ is the amalgamated product $P = (G_\alpha \times G_\delta) \star_{G_\delta} (G_\gamma \times G_\delta \times G_\eta)$ which contains $A$. Finally, $G = (G_\beta \times A) \star_A P$.*

**Proposition 3** *The natural inclusions of $P$ and of $A \times B$ into $G$ induce an isomorphism between $P \star_A (A \times B)$ and $G$.*

**Proof.** Trivial. Both sides satisfy the same universal property. □

**Corollary 4** *The word problem of a graph product $G = G(L, I; (G_\alpha)_{\alpha \in L})$ is solvable in logspace if and only if the word problem of all node groups $G_\alpha$ is in logspace.*

**Proof.** If the word problem of $G$ is in logspace then the same holds for all finitely generated subgroups. For the other direction we write $G$ as $P \star_A (A \times B)$ according to Proposition 3. Now, if $\pi_A : P \to A$ denotes the natural projection which is the identity on $A$ and sends all elements outside $A$ to 1, then we have $\pi_A(w) = w \iff w \in A$. Thus, the membership problem "$w \in A$?" reduces in logspace to an instance of the word problem "$w = \pi_A(w)$?" in $P$. By induction, the word problem of $P$ is solvable in logspace. Hence, Theorem 1 yields the result. □

### 4.1 Dependence graph representation

In order to represent elements in a graph product we use its dependence graph representation which was first introduced by Mazurkiewicz in trace theory for free partially commutative monoids [21]. This representation takes the complement relation $D = L \times L \setminus I$ into account. The relation $D$ is called *dependence relation*. The idea is that it is enough to "remember" the ordering between dependent letters; an idea which actually goes back to Keller [17].

We use $\Gamma_\alpha = G_\alpha \setminus \{1\}$ as (infinite) alphabets for representing group elements in the groups $G_\alpha$. A word in $\Gamma_\alpha^*$ denotes in a natural way an element in $G_\alpha$. The empty word denotes $1 \in G_\alpha$, all other elements of $G_\alpha$ have a representation as a single letter. If $a_1 \cdots a_n \in \Gamma_\alpha^*$ is a word then we denote by $[a_1 \cdots a_n]$ the corresponding element in $\Gamma_\alpha \cup \{1\}$ such that $a_1 \cdots a_n = [a_1 \cdots a_n]$ in the group $G_\alpha$. In addition, we let $\Gamma$ be the disjoint union over all $\Gamma_\alpha$ where $\alpha \in L$. In concrete algorithms we cannot work with $\Gamma$ directly. Instead we use for each $\Gamma_\alpha$ a finite subset $\Sigma_\alpha = \Sigma_\alpha^{-1} \subseteq \Gamma_\alpha$ such that $\Sigma_\alpha$ generates $G_\alpha$. We let $\Sigma \subseteq \Gamma$ be the the union over all $\Sigma_\alpha$. The way we represent words $w$ over $\Gamma^*$ is to write them with brackets $w = [u_1] \cdots [u_n]$ where each $u_i$ is a word in $\Sigma_\alpha^*$ for some $\alpha \in L$. If $u_i \neq 1 \in G_\alpha$ then $[u_i]$ becomes a letter of $\Gamma$. Since we work only with graph products where the word problem in node groups is solvable in logspace, we may always assume that $u_i \neq 1 \in G_\alpha$. Thus, $w = [u_1] \cdots [u_n]$ is a word in $\Gamma^*$ of length $|w| = n$. We start with an input word in $\Sigma^*$ and an initial bracketing is somewhat arbitrary as long as we group only letters from one $\Sigma_\alpha$ together.

Assume $A \cup B \subseteq L$ such that $A \times B \subseteq I$. Let $\Gamma_A = \bigcup_{\alpha \in A} \Gamma_\alpha$ and $\Gamma_B = \bigcup_{\beta \in B} \Gamma_\beta$. Then we call words $u \in \Gamma_A^*$ and $v \in \Gamma_B^*$ *independent*. They can be shuffled into each other without changing the image in $G$. In particular, if $u$ and $v$ are independent then $uv = vu$ in $G$. Thus, independence implies commutativity in $G$, but the converse is false because the independence relation $I$ is irreflexive. This is a subtle but important feature to have unique normal forms in the graph representation. As a special case, let $\beta \in L$ and denote $I(\beta) = \left( \bigcup_{(\alpha,\beta) \in I} \Gamma_\alpha \right)^*$. Then $u \in \Gamma_\beta^*$ and $v \in I(\beta)$ are examples of independent words. For a word $w = a_1 \cdots a_n \in \Gamma^*$ we define a node-labeled acyclic graph $D(w) = [V, E, \lambda]$, its *dependence graph*, as follows:

- The vertex set $V$ is $\{1, \ldots, n\}$.
- The label $\lambda(i)$ of a vertex $i$ is the letter $a_i \in G_{\alpha_i}$.
- Arcs are from $i$ to $j$ for $i < j$ where labels $\lambda(i)$, $\lambda(j)$ are dependent. Thus, $E = \{(i,j) \in V \times V \mid i < j \wedge (\alpha_i, \alpha_j) \in D\}$.

We view $D(w)$ as an abstract graph. This means we let $D(w) = D(w')$, if $D(w)$ and $D(w')$ are isomorphic as node-labeled directed graphs. For words $w, w' \in \Gamma^*$ we write $w \equiv w'$, if $D(w)$ and $D(w')$ are isomorphic. For example, if $a \in \Gamma_\alpha$ and $b \in \Gamma_\beta$ with $(\alpha, \beta) \in I$, then $ab \equiv ba$ and $D(ab) = D(ba)$. If $a, a' \in \Gamma_\alpha$ then $D(aa')$ has two vertices and one arc, but $D([aa'])$ has at most one vertex, hence $D(aa') \neq D([aa'])$ and $aa' \not\equiv [aa']$.

Given an abstract graph $D(w) = [V, E, \lambda]$ we associate to it a group element $g \in G$ as follows. We choose a topological sorting of $V$, this means we identify $V$ with $\{1, \ldots, n\}$ such that $(i,j) \in E$ implies $i < j$. Then we let $g = \lambda(1) \cdots \lambda(n) \in G$. It is easy to see by induction on $n$ that $g$ is well-defined. The graph $D(w)$ can be reconstructed by its *Hasse diagram*. The Hasse diagram removes all transitive edges. This means, we remove arc $(i,k)$ from $E$ as soon as there are $(i,j), (j,k) \in E$. The advantage of the Hasse diagram is that it is much smaller. For example the outdegree of every node is bounded by $|L|$ whereas the outdegree of a node in $D(w)$ can be $|w| - 1$.

The following rewriting procedure on dependence graphs relies on their Hasse diagrams. Let $D(w)$ be a dependence graph of some word $w \in \Gamma^*$.
**Rewriting procedure:** As long as for some $\beta \in L$ there is an arc in the Hasse diagram from $i$ to $j$ with labels $b$, $b' \in \Gamma_\beta$ do the following:

- Multiply $b \cdot b' = [bb']$ in $G_\beta$.
- If $[bb'] = 1$ then remove vertices $i$ and $j$ and their incident arcs.
- If $[bb'] \neq 1$ then remove vertex $j$ and its incident arcs. Relabel vertex $i$ by the letter $[bb'] \in \Gamma_\beta$.

The procedure transforms a dependence graph into a dependence graph with less vertices, but it does not change the corresponding group element in $G$. The rewriting procedure terminates in at most $|w|$ steps. It yields a dependence graph $D(\widehat{w})$ with the property that labels of neighbors in the Hasse diagram belong to different nodes in $L$. A dependence graph with this property is called *reduced*. A word $w \in \Gamma^*$ is called *reduced* if its dependence graph is reduced. We use the following characterization in order to check that a word $w$ and its dependence graph $D(w)$ are reduced.

**Lemma 5** *A word $w \in \Gamma^*$ is reduced if and only if it does not contain any factor $bub'$ such that $b, b' \in \Gamma_\beta$ and $u \in I(\beta)$ where $\beta \in L$.*

**Proof.** If a factor $bub'$ with $b, b' \in \Gamma_\beta$ and $u \in I(\beta)$ appears in $w$ then vertices $i$ and $j$ corresponding to the letters $b$ and $b'$ are neighbors in the Hasse diagram of $D(w)$ which is therefore not reduced. For example, in Figure 2 the dependence graph of the word $ab\bar{a}ca\bar{b}$ on the left is not reduced but the dependence graph of the word $bc\bar{b}a$ on the right is reduced.

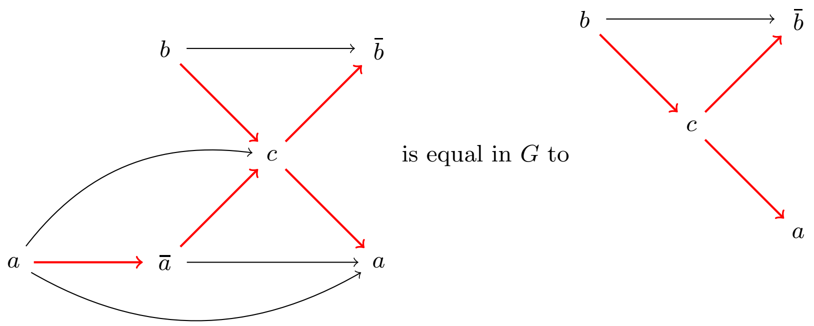



**Fig. 2.** Dependence graphs (Hasse diagrams in red) of $ab\bar{a}ca\bar{b}$ and $bc\bar{b}a$.

For the other direction, assume that no such factor appears. Let $w = a_1 \cdots a_n$ and $[V, E, \lambda]$ be its dependence graph. Consider $1 \leq i < j \leq n$ such that $\lambda(i), \lambda(j) \in \Gamma_\beta$ for some $\beta \in L$. Then there is some $i < k < j$ such that $\lambda(k) \in \Gamma_\alpha$ and $(\alpha, \beta) \in D$. Hence, $D(w)$ is reduced. $\square$

If a word $w$ is reduced, then its dependence graph is a unique normal form for the corresponding element in the graph product $G$. This follows from the following technical result.

**Proposition 6 ([6])** *The rewriting procedure is confluent and yields normal forms for group elements in the graph product. In particular, reduced dependence graphs are isomorphic if and only if the associated group elements are the same.*

In the following, if $w \in \Gamma^*$ then $[w]$ denotes a reduced word such that $w = [w] \in G$. The dependence graph $D([w])$ is uniquely defined by $w$ (up to isomorphism). The normal form is therefore $D([w])$ rather than the reduced word $[w]$. Note that the notation $[w]$ is a generalization of the notation $[a_1 \cdots a_n]$ used above. Proposition 6 reduces the word problem of $G$ to the word problems of the node groups as follows: We start with a word $w \in \Gamma^*$. In order to run the rewriting procedure on $D(w)$, we just have to decide word problems in node groups $G_\beta$. At the end we have $w = 1 \in G$ if and only if the procedure terminates in the empty graph. It is however far from obvious that we can implement the procedure in logspace. The following theorem is crucial. It uses Corollary 4 as a black box.

**Theorem 7.** *Let $G = G(L, I; (G_\alpha)_{\alpha \in L})$ be a finitely generated graph product. Assume that the word problem for each node group $G_\alpha$ is in logspace. Then there is a logspace computation which transforms an input word $w$ over generators into a reduced dependence graph for $w$.*

**Proof.** The proof ends after Lemma 9. The result does not depend on the choice of generators. Therefore, we may assume $w \in \Sigma^*$ where $\Sigma = \bigcup_{\alpha \in L} \Sigma_\alpha$ and each $\Sigma_\alpha = \Sigma_\alpha^{-1}$ generates $G_\alpha$. Every letter of the countable alphabet $\Gamma$ is represented internally by some word in $\Sigma^*$. Moreover, as $\Sigma \subseteq \Gamma$ the input $w$ is a product of letters over $\Gamma$. We perform $|L|$ rounds of logspace reductions. In each round we minimize the number of letters $a_i \in \Gamma_\alpha$ for some $\alpha \in L$. For this we introduce the following notion. We say that $w = u_0 a_1 u_1 \cdots a_n u_n$ is the *$\alpha$-factorization* of a word $w \in \Gamma^*$, if we have $a_i \in \Gamma_\alpha$ for $1 \le i \le n$ and $u_i \in (\Gamma \setminus \Gamma_\alpha)^*$ for $0 \le i \le n$. The $\alpha$-factorization exists and it is unique. We use the following procedure.
**The $\alpha$-reduction:** Let $w = u_0 a_1 u_1 \cdots a_n u_n$ be its $\alpha$-factorization. For $n = 0$ do nothing. For $n > 0$ start with $i = 1$.

– From left-to-right: Stop at $a_i$. Compute the maximal $m \ge i$ (by calling instances of the word problem in $G$) such that

$$a_i u_i \cdots a_m u_m = a_i \cdots a_m u_i \cdots u_m \in G$$

– Replace $a_i u_i \cdots a_m u_m$ by $a u_i \cdots u_m$ where $a = [a_i \cdots a_m] \in G_\alpha = \Gamma_\alpha \cup \{1\}$.
– If $m = n$ then the $\alpha$-reduction is finished, otherwise change $i$ to $m + 1$, stop there and continue the left-to-right phase.

The overall procedure performs $\alpha$-reductions for all $\alpha \in L$ in any order. The output can be read as a word $\widehat{w}$ over $\Gamma$. Reading from left to right we compute in a final round the actual dependence graph $D(\widehat{w})$. Each $\alpha$-reduction can be done in logspace due to Corollary 4. Since there are only a constant number of rounds the overall procedure is in logspace, too. It remains to prove that the output $D(\widehat{w})$ is the reduced dependence graph corresponding to the input word $w$. The proof will be based on the next two lemmata.

We define a word $w \in \Gamma^*$ to be $\alpha$-reduced, if every other word $w' \in \Gamma^*$ having less letters from $\Gamma_\alpha$ denotes a different group element in $G$. Due to Proposition 6 a word $w$ is reduced if and only if it is $\alpha$-reduced for all $\alpha \in L$.

**Lemma 8** *Let $w = u_0 a_1 u_1 \cdots a_n u_n \in \Gamma^*$ be its $\alpha$-factorization. Then $w$ is $\alpha$-reduced if and only if $a_i u_i a_{i+1} \neq a_i a_{i+1} u_i \in G$ for all $1 \leq i < n$.*

**Proof.** If $a_i u_i a_{i+1} = a_i a_{i+1} u_i \in G$ for some $1 \leq i < n$, then $w$ is not $\alpha$-reduced. Hence, it is enough to show that if $a_i u_i a_{i+1} \neq a_i a_{i+1} u_i \in G$ for all $1 \leq i < n$ then $w$ is $\alpha$-reduced. This is true, if $w$ is reduced. Hence we may assume that $w$ is not reduced. Then there exists $\beta \in L$ and a factor $bub'$ with $b, b' \in G_\beta$ and $u \in I(\beta)$. Since $a_i u_i a_{i+1} \neq a_i a_{i+1} u_i$ we must have $\alpha \neq \beta$. If the factor $bub'$ is a factor inside some $u_i$, then we can rewrite it by $[bb']u$ and we obtain a word $w'$ which satisfies the same property, but which length over $\Gamma$ is shorter. Hence by induction on the length $w'$ is $\alpha$-reduced. This implies that $w$ is $\alpha$-reduced, too.

Thus we may assume that for some $i < j$ we have $u_i = p_i b q_i$ and $u_j = p_j b' q_j$ with $q_i a_{i+1}$, $a_j p_j \in I(\beta)$. In particular, $u_i a_{i+1} = p_i q_i a_{i+1} b \in G$ and we have $w = w'$ in $G$ where $w' = u_0 a_1 u_1 \cdots u_{i-1} a_i p_i q_i a_{i+1} b u_{i+1} \cdots a_n u_n$. By induction on $|j - i|$ we obtain that $w'$ is $\alpha$-reduced. This implies again that $w$ is $\alpha$-reduced. □

Let $w = u_0 a_1 u_1 \cdots a_n u_n \in \Gamma^*$ be its $\alpha$-factorization and $0 \leq i \leq n$. We say that $u_0 a_1 u_1 \cdots a_i u_i$ is an $\alpha$-prefix, if there are no $0 < k < \ell$ such that $k \leq i$ and $a_k u_k \cdots a_\ell u_\ell = [a_k \cdots a_\ell] u_k \cdots u_\ell \in G$ with $\ell \leq n$. Note that $u_0$ is an $\alpha$-prefix. Moreover, $w$ is an $\alpha$-prefix of itself if and only if $w$ is $\alpha$-reduced by Lemma 8.

**Lemma 9** *Let $w = u_0 a_1 u_1 \cdots a_n u_n \in \Gamma^*$ be its $\alpha$-factorization and $0 \leq i < n$ such that $u_0 a_1 u_1 \cdots a_i u_i$ is an $\alpha$-prefix and let $m$ be maximal such that*
$a_{i+1} u_{i+1} \cdots a_m u_m = [a_{i+1} \cdots a_m] u_{i+1} \cdots u_m \in G$.

*Then $u_0 a_1 u_1 \cdots a_i u_i [a_{i+1} \cdots a_m] u_{i+1} \cdots u_m$ is an $\alpha$-prefix of*
$u_0 a_1 u_1 \cdots a_i u_i [a_{i+1} \cdots a_m] u_{i+1} \cdots u_m a_{m+1} u_{m+1} \cdots a_n u_n$.

**Proof.** Straightforward since $i < m \leq n$ and $m$ was chosen to be maximal. □

In order to finish the proof of Theorem 7 it is enough to show that the logspace procedure "$\alpha$-reduction" computes an $\alpha$-reduced word. The invariant of the procedure is that in the left-to-right phase $\alpha$-prefixes are computed. This follows from Lemma 9. At the end of the $\alpha$-reduction the word $w$ itself becomes an $\alpha$-prefix. But then Lemma 8 tells us that $w$ is an $\alpha$-reduced word. Thus, if after an $\alpha$-reduction we perform a $\beta$-reduction then the word is $\alpha$- and $\beta$-reduced, and so on. Hence, the result of Theorem 7. □

Theorem 7 implies the following result.

**Corollary 10** *The word problem of a graph product of linear groups is solvable in logspace.*

The next corollary shows that shortlex normal forms can be computed in logspace if this is possible for all node groups. In the statement of the result we assume that we have a linear order on the set of nodes $L$ and that each node group $G_\alpha$ is finitely generated by some linearly ordered set $\Sigma_\alpha$. (For simplicity assume $\Sigma_\alpha = \Sigma_\alpha^{-1}$.) We use $\Sigma = \bigcup_{\alpha \in L} \Sigma_\alpha$ as a generating alphabet for $G = G(L, I; (G_\alpha)_{\alpha \in L})$. The linear order is as follows. If $\alpha < \beta$ then every letter in $\Sigma_\alpha$ is before $\Sigma_\beta$. If $\alpha = \beta$ then we use the order in $\Sigma_\alpha$. The *shortlex order* on $\Sigma^*$ is defined as usual: If $|u| < |v|$ then $u < v$ in the shortlex order. If $|u| = |v|$ and $u$ is lexicographically before $v$ then $u < v$ in the shortlex order. The *shortlex normal form* of an element $g$ is the unique minimal word $w \in \Sigma^*$ which satisfies $w = g$ in $G$. The following corollary is an immediate consequence of Theorem 7. It generalizes the main result in [5].

**Corollary 11** *Let $\Sigma$ and the graph product $G = G(L, I; (G_\alpha)_{\alpha \in L})$ as above. If for each node group $G_\alpha$ the shortlex normal form is computable in logspace, then the shortlex normal form in $G$ is computable in logspace.*

The following proposition will be used in the next section for solving conjugacy in graph products. Due to Theorem 7 it generalizes Corollary 4. It states that "pattern matching" over dependence graphs is possible in logspace.

**Proposition 12** *Let $G = G(L, I; (G_\alpha)_{\alpha \in L})$ be a graph product such that the word problem of all node groups $G_\alpha$ is in logspace. Then the following problem can be solved in logspace. Input: Words $p$, $t \in \Gamma^*$. Problem: Do $x, y$ exist such that $t \equiv xpy$?*

**Proof.** We may assume that $1 \le |p|$ and $|p|_\alpha \le |t|_\alpha$ for all $\alpha \in L$. First, we compute $D(t) = [V, E, \lambda]$ and $D(p) = [V', E', \lambda']$. Let $|t| = |V| = n$ and $M' \subseteq V'$ be the set of minimal vertices in $D(p)$. There are at most $n^{|M'|}$ positions in $t$ which may correspond to $M'$. The logspace procedure may investigate each of them one after another. So we may think that a copy $M \subseteq V$ of $M'$ is fixed. In a next round we keep in $V$ only those vertices $u$ which can be reached by a directed path from some vertex in $M$. All other vertices are deleted. Now we run a symmetric procedure with maximal vertices. After that we may assume that the sets of minimal vertices of $p$ and $t$ and the sets of maximal vertices of $p$ and $t$ coincide. However, now there are $x, y$ such that $t \equiv xpy$ if and only if $D(t) = D(p)$. This can be checked in logspace because the word problem of all node groups $G_\alpha$ is in logspace. Details are left to the reader. $\square$

## 5 Conjugacy

Two group elements $u, v \in G$ are *conjugate* if there exists a $z \in G$ with $z^{-1}uz = v$ in $G$. If $u$ and $v$ are conjugate we write $u \sim v$. The conjugacy problem is to decide on input words $u, v$ whether or not $u \sim v$ as elements of $G$. The aim of this section is to prove the following result.

**Theorem 13.** *The conjugacy problem of a graph product is solvable in logspace if and only if the conjugacy problem of all node groups is in logspace.*

The easy direction of Theorem 13 is the implication from left to right, because for all $u \in G_\alpha$ and all $z \in G$ where the reduced dependence graph contains a vertex with a label in some $G_\beta$ with $(\alpha, \beta) \in D$ and $\alpha \neq \beta$ we have $zuz^{-1} \notin G_\alpha$. Hence, for $u, v \in G_\alpha$ we have $u \sim v$ in $G_\alpha$ if and only if $u \sim v$ in $G$. Thus, we have to show only that if the conjugacy problem of all node groups $G_\alpha$ is in logspace then the conjugacy problem of $G$ is in logspace. Before we prove the other direction let us state an immediate consequence of Theorem 13. It solves an open problem for two prominent classes of finitely generated groups.

**Corollary 14** *The conjugacy problem of a graph group or a right-angled Coxeter group can be solved in logspace.*

The proof of Theorem 13 (and its corollaries) covers the rest of this section. The logspace algorithm can be found at the end of this section, too. Using Theorem 7 we may assume that the input words $u, v \in \Gamma^*$ are reduced. Actually we work with their dependence graph representations. Therefore it is convenient to have a special notation. We write $w \equiv w'$ if $D(w)$ and $D(w')$ are isomorphic. Recall that, if $w \in \Gamma^*$ is reduced, then $w \equiv w'$ if and only if $w = w'$ in the graph product $G$. If $w = u_0 a_1 u_1 \cdots a_n u_n$ is the $\alpha$-factorization then we call $n$ the $\alpha$-length. We denote it by $|w|_\alpha$. Thus $|w|_\alpha$ is the number of vertices in the dependence graph of $w$ having a label in $\Gamma_\alpha$. For later use we also define the *alphabet* of a word $w \in \Gamma^*$ by $\mathrm{alph}(w) = \{\alpha \in L \mid |w|_\alpha \geq 1\}$. If $w$ is reduced, then it depends on the image in $G$, only. Thus, we can define the alphabet of group elements, too. We also say that a word is *connected* if $\mathrm{alph}(w)$ induces a connected subgraph in the dependence graph $(L, D)$. Assume that $\mathrm{alph}(w)$ splits into connected components $A_1 \cup \cdots \cup A_k$. Then we have $w \equiv w_1 \cdots w_k$ with $\alpha(w_i) = A_i$ and $w_i w_j \equiv w_j w_i$ for all $1 \leq i < j \leq k$. If $w$ is reduced, then all $w_i$ are reduced. Therefore we can split every group element of $G$ into connected components which commute pairwise. Next, we use the following fact.

**Lemma 15** *Let $u \in \Gamma^*$ be reduced. Then there exists a unique minimal $\widetilde{u}$ such that $u \equiv p\widetilde{u}\overline{p}$ for some $p \in \Gamma^*$.*

**Proof.** If there is no $a \in \Gamma_\alpha$ such that $u \equiv a u_1 \overline{a}$ then we must choose $p = 1$. Otherwise we rewrite $u$ into $u_1$. If we have also $b \in \Gamma_\beta$ such that $u \equiv b u_2 \overline{b}$ and $a \neq b$ then we have $(\alpha, \beta) \in I$ and $u \equiv ab u_3 \overline{b}\,\overline{a}$. Thus the rewriting procedure is strongly confluent and therefore $p$ exists and the reduced dependence graph of $\widetilde{u} \in \Gamma^*$ is uniquely defined by $u$. □

We say that a reduced word $u$ is *cyclically reduced* if the dependence graph of $u$ does not contain any minimal vertex $i$ and any maximal vertex $j$ such that $i \neq j$ but $\lambda(i), \lambda(j) \in \Gamma_\alpha$ for some $\alpha \in L$. Thus, a reduced word $u$ is not cyclically reduced if and only if $u \equiv au'a'$ for some $a, a' \in \Gamma_\alpha$. If $u$ is not cyclically reduced then $u \sim u'[a'a]$ and the length of $u'[a'a]$ is shorter than $u$. Therefore it is enough to solve the conjugacy problem for cyclically reduced words, but the initial problem is to compute them in logspace. The key observation to overcome this difficulty is the following lemma.

**Lemma 16** *Let $u \in \Gamma^*$ be reduced. Then there are reduced words $p, r, m, s$ such that*

- $u \equiv prms\overline{p}$
- $|[sr]|_\alpha = |r|_\alpha = |s|_\alpha \leq 1$ *for all* $\alpha \in L$
- $m[sr]$ *is cyclically reduced and* $u \sim m[sr]$.

**Proof.** Choose $p$ and $\widetilde{u}$ according to Lemma 15. Next, there is a unique maximal $r$ such that $\widetilde{u} \equiv rms$ with $|[sr]|_\alpha = |r|_\alpha \leq 1$ for all $\alpha \in L$. This follows because $p$ has maximal length. Actually for some subset $M \subseteq L$ of pairwise independent nodes we have $r \equiv \prod_{\alpha \in M} a_\alpha$ and $s \equiv \prod_{\alpha \in M} b_\alpha$ such that $a_\alpha, b_\alpha \in \Gamma_\alpha$. Moreover $[b_\alpha a_\alpha] \in \Gamma_\alpha$. Thus, $m[sr] \equiv m \prod_{\alpha \in M} [b_\alpha a_\alpha]$ is reduced. It is cyclically reduced because $r$ has been chosen to be maximal and $u$ is reduced. The assertion $u \sim m[sr]$ is trivial. □

**Lemma 17** *There is a logspace computation which on input $u \in \Gamma^*$ outputs a cyclically reduced word $u'$ such that $u \sim u'$.*

**Proof.** The idea is to compute a cyclically reduced word $m[sr]$ with $u \sim m[sr]$ by reducing $w = uu$. Let us see what happens if we start the reduction process on $w = uu$ where, according to Lemma 16, $u \equiv prms\overline{p}$ is reduced. We can write $w \equiv prms\overline{p}prms\overline{p}$. The word $prm[sr]ms\overline{p}$ is a reduced word because $m[sr]$ and $[sr]m$ are cyclically reduced. Hence $[w] \equiv prm[sr]ms\overline{p}$. In order to determine the factor $m[sr]$ inside $[w]$ we compute for each $\alpha \in L$ the $\alpha$-lengths $|w|_\alpha$, $|[w]|_\alpha$, $|p|_\alpha$, and $|r|_\alpha$. In a first logspace computation we determine the $\alpha$-factorization of $u$. This gives us $n \in \mathbb{N}$ with $n = |u|_\alpha$ and therefore $2n = |w|_\alpha$. A second logspace computation using Theorem 7 yields $k \in \mathbb{N}$ with $k = |[w]|_\alpha$. Let $\varepsilon = |r|_\alpha$. We know $\varepsilon = |[sr]|_\alpha = |r|_\alpha = |s|_\alpha \leq 1$. We obtain

$$\begin{aligned} 2n &= 4\,|p|_\alpha + 2\,|m|_\alpha + 4\varepsilon \\ k &= 2\,|p|_\alpha + 2\,|m|_\alpha + 3\varepsilon \end{aligned}$$

Thus, $2n - k = 2\,|p|_\alpha + \varepsilon$. If $k$ is even then $\varepsilon = 0$ otherwise $\varepsilon = 1$. Knowing $\varepsilon = |r|_\alpha$ we know $|p|_\alpha$ and $|m|_\alpha$, too. We conclude that the $i$-th vertex of $D([w])$ which has a label in $\Gamma_\alpha$ belongs to the factor $m[sr]$ if and only if

$$|p|_\alpha + \varepsilon < i < k - |p|_\alpha - \varepsilon - |m|_\alpha\,.$$

□

**Lemma 18** *Let $x, y \in \Gamma^*$ be cyclically reduced such that $x \sim y$. Then $|x|_\alpha = |y|_\alpha$ for all $\alpha \in L$. In particular,* $\mathrm{alph}(x) = \mathrm{alph}(y)$.

**Proof.** Let $z \in \Gamma^*$ be reduced of minimal length such that $xz = zy \in G$. Assume by contradiction that there exists some $\alpha \in \mathrm{alph}(x) \setminus \mathrm{alph}(y)$. On the right hand side no reduction can involve letters from $\Gamma_\alpha$, hence $|zy|_\alpha = |[zy]|_\alpha = |[z]|_\alpha$, but $|xz|_\alpha \geq 1 + |z|_\alpha$. Hence a reduction between $x$ and $z$ must occur. Hence there exist $a, a' \in \Gamma_\alpha$ such that we can write $x \equiv x'a$, $z \equiv a'z'$. If $aa' = 1 \in G$ then $ax'z' = axz = azy = z'y$ and hence $ax'z' = z'y$. Since $x$ is cyclically reduced, the word $ax'$ is reduced, too. By induction on the length of $z$ we obtain $|ax'|_\alpha = |y|_\alpha$ and $|y|_\alpha \geq 1$ which is a contradiction. Thus, $aa' \neq 1 \in G$ and $[aa'] \in \Gamma_\alpha$. Therefore, the $\alpha$-length of $x'[aa']z'$ is equal to the $\alpha$-length of $z = a'z'$. As $a'$ is minimal in $z$ we conclude $[aa'] = a'$, hence $a = 1 \in G$, which is again a contradiction. □

For a subset $C \subseteq L$ let $G_C$ be the graph product which is defined with respect to the independence relation $(C, I_C)$ where $I_C = I \cap C \times C$. Recall that $G_C$ is a retract of $G$ with respect to the canonical projection $\pi_C : G \to G_C$ since $\pi_C(g) = g$ for all $g \in G_C$. The following lemma shows that it is enough to decide conjugacy on connected words.

**Lemma 19** *Let $x, y \in \Gamma^*$ be reduced such that* $\mathrm{alph}(x) = \mathrm{alph}(y) = A \cup B$ *with $A \times B \subseteq I$. Write $x = x_A x_B, y = y_A y_B$ with* $\mathrm{alph}(x_C) = \mathrm{alph}(y_C) = C$ *for $C \in \{A, B\}$. Then we have $x \sim y$ in $G$ if and only if $x_C \sim y_C$ in $G$ for $C \in \{A, B\}$. Moreover, $x_C \sim y_C$ in $G$ if and only if $x_C \sim y_C$ in $G_C$.*

**Proof.** Consider the canonical projection $\pi_C : G \to G_C$ and let $z_C = \pi_C(z)$. If $xz = zy$, then $x_C z_C = \pi_C(xz) = \pi_C(zy) = z_C y_C$. Hence $x_C \sim y_C$ in $G_C$. This implies $x_C \sim y_C$ in $G$. Now, let $x_C \sim y_C$ in $G$ for $C \in \{A, B\}$. Choose $z'$ and $z''$ such that $x_A z' = z' y_A$ and $x_B z'' = z'' y_B$. We obtain $x_A \pi_A(z') = \pi_A(z') y_A$ and $x_B \pi_B(z'') = \pi_B(z'') y_B$. It follows $xz = zy \in G$ for $z = \pi_A(z') \pi_B(z'')$. □

We are now ready to prove the remaining implication of Theorem 13. For this we may assume that the conjugacy problem in all $G_\alpha$ is solvable in logspace. In order to solve conjugacy in logspace for $G$, it is enough to consider cyclically reduced and connected input words $x$ and $y$ such that $\mathrm{alph}(x) = \mathrm{alph}(y)$. Let us consider the special case where $|\mathrm{alph}(x)| \leq 1$ first. Then we have $x$, $y \in G_\alpha$ for some $\alpha \in L$. Another consequence of Lemma 19 is that now $x \sim y$ in $G$ if and only if $x \sim y$ in $G_\alpha$. This is the only place where we use that the conjugacy problem is solvable in logspace for all node groups. Thus, we may assume that $|\mathrm{alph}(x)| \geq 2$. This leads us to combinatorics on dependence graphs in the spirit of [8].

Let us define the notion of *transposition*. We say that words $u, v \in \Gamma^*$ are *transposed* if there are $r, s \in \Gamma^*$ such that $u \equiv rs$ and $v \equiv sr$. Thus, the definition is based on dependence graphs. Transposition is a reflexive and symmetric relation. But unlike the usual definition for words it is not transitive, in general. (The usual definition is the special case where $I$ is empty.) By $u \approx v$ we denote the transitive closure of transposition. We can view $\approx$ as an equivalence relation on dependence graphs. A crucial observation is that if $u \approx v$ and $u$ is cyclically reduced then $v$ is cyclically reduced, too. In the following, we consider $\approx$ only for cyclically reduced words. Clearly, if $u \approx v$, then $u \sim v$ in $G$, but the converse does not hold in general. To see this let $a \sim a'$ in some $G_\alpha$ with $a \neq a'$. Then $a, a' \in \Gamma_\alpha$ are cyclically reduced, but $a \not\approx a'$. Using transpositions on cyclically reduced words we never can multiply letters together which are neighbors in the Hasse diagram and we obtain an analogue to Duboc's classical result which characterizes $\approx$ for partially commutative monoids [11].

**Proposition 20** *Let $u, v \in \Gamma^*$ be cyclically reduced words. Then we have $u \approx v$ if and only if the following two conditions hold. First, we have $|u|_\alpha = |v|_\alpha$ for all $\alpha \in L$ and second, there are reduced words $p, q$ such that $puq \equiv v^{|L|}$.*

**Proof.** Duboc's result [11] (see also [8, Thm. 3.3.3]) is stated for Mazurkiewicz traces. It can be applied because $u, v \in \Gamma^*$ are cyclically reduced. Actually her proof can be applied verbatim in our setting.

First, let $u \approx v$. It follows $|u|_\alpha = |v|_\alpha$ for all $\alpha \in L$. We may assume $u \not\equiv v$ and we use induction on the number of transpositions to transform $v$ into $u$. Since $u \not\equiv v$ there are $r$, $s$ such that $v \equiv rs$ and such that the number of transpositions to transform $sr$ into $u$ has decreased. By induction, there are reduced words $p', q'$ such that $p'uq' \equiv (sr)^k$ for some $k \in \mathbb{N}$. Let $p = rp'$ and $q = q's$ then we see $puq \equiv (rs)^{k+1}$. It remains to show that we can bound the exponent $k$ by $|L|$. To see this let $puq \equiv v_1 \cdots v_k$ for some $k$ such that each $v_\ell \equiv v$. Without restriction, $u$ is connected. A minimal vertex $i_0$ of $u$ must be located in $v_1$. Now, for a vertex $j$ of $u$ we let $d(i_0, j)$ be the length of a shortest path from $i_0$ to $j$ in the dependence graph $D(u)$. We claim, that if $d = d(i_0, j)$, then $j$ appears as a vertex in the prefix $v_1 \cdots v_{d+1}$. The claim follows by induction on $d$. Let $i$ be a vertex of $u$ which appears in $v_1 \cdots v_d$ and $\lambda(i) \in \Gamma_\alpha$, $\lambda(j) \in \Gamma_\beta$, with $(\alpha, \beta) \in D$. On a path from $i$ to $j$ in $v_1 \cdots v_k$ there are at most $|u|_\beta$ vertices with a label in $\Gamma_\beta$. Since $|u|_\beta = |v|_\beta$, we conclude the claim. Since always $d \leq |L| - 1$, we obtain that there are reduced words $p, q$ such that $puq \equiv v^{|L|}$.

For the other direction let $|u|_\alpha = |v|_\alpha$ for all $\alpha \in L$ and $p, q$ be reduced words such that $puq \equiv v^k$ for some $k \in \mathbb{N}$. If we have $|p| = 0$ then $u \equiv v$ since $|u|_\alpha = |v|_\alpha$ for all $\alpha \in L$. Thus we have $p \equiv ap'$ for some $a \in \Gamma_\alpha$ and $p' \in \Gamma^*$. We conclude $v \equiv av'$ for some $v' \in \Gamma^*$. This leads to $p'uqa \equiv (v'a)^k$. By induction on the length of $p$ we obtain $u \approx (v'a) \approx (av') \equiv v$. Hence the result. □

**Corollary 21** *Let $G = G(L, I; (G_\alpha)_{\alpha \in L})$ be a graph product and $\Gamma$ as above be such that the word problem of all node groups $G_\alpha$ is in logspace. Then the following problem can be solved in logspace. Input: Cyclically reduced words $u$, $v \in \Gamma^*$. Problem: Do we have $u \approx v$?*

**Proof.** This is a direct consequence of Proposition 12 and Proposition 20. □

Using Corollary 21 the proof of Theorem 13 is reduced to showing the following combinatorial proposition.

**Proposition 22** *Let $G = G(L, I; (G_\alpha)_{\alpha \in V})$ be a graph product, $\Gamma$ as above, and let $x, y \in \Gamma^*$ be cyclically reduced and connected words such that* $\mathrm{alph}(x) =$

alph($y$) *with* |alph($x$)| $\geq 2$*. Then we have* $x \sim y$ *in the group* $G$ *if and only if* $x \approx y$*.*

**Proof.** Let $x \sim y$. We have to show $x \approx y$. Choose some reduced word $z \in \Gamma^*$ of minimal length such that $xz = zy$ in $G$. By Lemma 19 we have $\mathrm{alph}\, z \subseteq \mathrm{alph}\, x$. Assume that $xz$ was not reduced. Then we have $x \equiv x'a$ and $z \equiv a'z'$ such that $[aa'] \in G_\alpha$ for some $\alpha \in L$. Since $a'$ is minimal in $z$ and $\mathrm{alph}\, z \subseteq \mathrm{alph}\, x$ we conclude that $a'$ is also minimal in $x$. Actually there is a minimal vertex in $x'$ with label $a'$, because $x$ is connected and $|\mathrm{alph}(x)| \geq 2$. This implies $x = a'x''a$ which is a contradiction since $x$ is cyclically reduced.

Thus, $xz$ is reduced and therefore $zy$ is reduced, too. This implies $xz \equiv zy$ because $xz = zy$ in $G$. We can apply the Levi Lemma for traces [8, Thm. 3.2.2]. It yields the existence of $p, r, s, q \in \Gamma^*$ such that $x \equiv pr, z \equiv sq, z \equiv ps, y \equiv rq$ with $r$ and $s$ independent. If $|p| = 0$ or $|q| = 0$ then $x = y$, hence we may assume $|s| < |z|$. Moreover, $rps \equiv rsq \equiv srq$ because $rs \equiv sr$. Thus, by induction we obtain $rp \approx rq \equiv y$. Now, $rp$ and $x \equiv pr$ are transposed, hence $x \approx y$. □

We now have all the ingredients to describe the algorithm which proves Theorem 13 (and Corollary 14).

**The Algorithm for solving conjugacy in a graph product.**
**Input:** $u$, $v \in \Gamma^*$. **Question:** $u \sim v$ in $G$?

1. Compute $u$, $v$ in reduced form using Theorem 7.
2. Compute $u$, $v$ in cyclically reduced form using Lemma 16.
3. Reduce to the case that $u$, $v$ are cyclically reduced and connected using Lemma 19.
4. Compute alph($u$), alph($v$). If $\mathrm{alph}(u) \neq \mathrm{alph}(v)$ then $u \not\sim v$ in $G$. Hence without restriction, $\mathrm{alph}(u) = \mathrm{alph}(v)$.
5. If $|\mathrm{alph}(u)| \leq 1$ then $u \sim v$ in $G$ if and only if $u$ and $v$ are conjugated in the corresponding node group. Hence without restriction, $|\mathrm{alph}(u)| \geq 2$.
6. Since $|\mathrm{alph}(u)| \geq 2$, we have now by Proposition 22 that $u \sim v$ in $G$ if and only if $u \approx v$. Decide $u \approx v$ using Corollary 21.

## 6 Conclusion

The paper shows transfer results for the logspace complexity of important group-theoretical decision problems from node groups to graph products. This concerns the word problem, computing geodesics, and the conjugacy problem. The first two results were known for RAAGs (graph groups) before, but not for graph products in general. The earlier proof for RAAGs relied on geometry and linear representations and these methods are not available for graph products, in general. The present proof is purely combinatorial. Our results concerning the conjugacy problem are new even for RAAGs, and they go clearly far beyond that. Our results also support a conjecture that a graph product of linear groups is again linear. A proof of this conjecture might proceed using a similar induction scheme as used here, but this is highly speculative and not in the scope of purely combinatorial methods.

An interesting question is whether analogous transfer results hold in complexity classes below logspace. The main obstacle is the word problem for free groups in two generators. The precise complexity of this word problem is a long standing open question in algorithmic group theory.

A promising line of future research is to extend the results beyond graph products. For example, the results in Section 3 can easily be extended from direct products to semi-direct products. But this is only the first step.

## References


1. M. Bestvina and N. Brady. Morse theory and finiteness properties of groups. *Inventiones Mathematicae*, 129(3):445–470, 1997.
2. J.-Y. Cai. Parallel computation over hyperbolic groups. In *Proc. 24th ACM Symp. on Theory of Computing, STOC 92*, pages 106–115. ACM-press, 1992.
3. J. Crisp, E. Godelle, and B. Wiest. The conjugacy problem in right-angled Artin groups and their subgroups. *Journal of Topology*, 2(3):442–460, 2009.
4. J. Crisp and B. Wiest. Embeddings of graph braid and surface groups in right-angled Artin groups and braid groups. *Algebraic & Geometric Topology*, 4:439–472, 2004.
5. V. Diekert, J. Kausch, and M. 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.
6. V. Diekert and M. Lohrey. Word equations over graph products. *International Journal of Algebra and Computation*, 18:493–533, 2008. Conference version in FSTTCS 2003.
7. V. Diekert and A. Muscholl. Solvability of equations in free partially commutative groups is decidable. *International Journal of Algebra and Computation*, 16:1047–1070, 2006. Journal version of ICALP 2001, 543–554, LNCS 2076.
8. V. Diekert and G. Rozenberg, editors. *The Book of Traces.* World Scientific, Singapore, 1995.
9. C. Droms. Graph groups, coherence and three-manifolds. *Journal of Algebra*, 106(2):484–489, 1985.
10. C. Droms, J. Lewin and H. Servatius. The length of elements in free solvable groups. *Proc. Amer. Math. Soc.*, 119:27–33, 1993.
11. Ch. Duboc. On some equations in free partially commutative monoids. *Theoretical Computer Science*, 46:159–174, 1986.
12. M. Elder. A linear-time algorithm to compute geodesics in solvable Baumslag-Solitar groups. *Illinois Journal of Mathematics*, 54(1):109–128, 2010.
13. M. Elder, G. Elston, and G. Ostheimer. On groups that have normal forms computable in logspace. *J. Algebra*, 381:260–281, 2013.
14. M. Elder and A. Rechnitzer. Some geodesic problems in groups. *Groups. Complexity. Cryptology*, 2(2):223–229, 2010.
15. N. Haubold, M. Lohrey and C. Mathissen. Compressed Decision Problems for Graph Products and Applications to (outer) Automorphism Groups. *IJAC*, 22, 2012.

16. T. Hsu and D. T. Wise. On linear and residual properties of graph products. *Michigan Mathematical Journal*, 46(2):251–259, 1999.
17. R. M. Keller. Parallel program schemata and maximal parallelism I. Fundamental results. *Journal of the Association for Computing Machinery*, 20(3):514–537, 1973.
18. R. J. Lipton and Y. Zalcstein. Word problems solvable in logspace. *Journal of the Association for Computing Machinery*, 24(3):522–526, 1977.
19. M. Lohrey. Decidability and complexity in automatic monoids. *International Journal of Foundations of Computer Science*, 16(4):707–722, 2005.
20. M. Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups Complexity Cryptology*, 4:241–299, 2012.
21. A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.
22. C. F. Miller III. *On group-theoretic decision problems and their classification*, volume 68 of *Annals of Mathematics Studies*. Princeton University Press, 1971.
23. C. F. Miller III. Decision problems for groups – survey and reflections. *Springer*, 1992.
24. A. Myasnikov, V. Roman'kov, A. Ushakov, and A. Vershik. The word and geodesic problems in free solvable groups. *Transactions of the American Mathematical Society*, 362:4655–4682, 2010.
25. Ch. Papadimitriou. *Computation Complexity*. Addison-Wesley, 1994.
26. M. Paterson and A. Razborov. The set of minimal braids is co-NP-complete. *J. Algorithms*, 12:393–408, 1991.
27. D. Robinson. *Parallel Algorithms for Group Word Problems*. PhD thesis, University of California, San Diego, 1984.
28. M. Sapir. Asymptotic invariants, complexity of groups and related problems. *Bulletin of Mathematical Sciences*, 1(2):277–364, 2011.
29. J.-P. Serre. *Trees*. Springer, 1980. French original 1977.
30. H.-U. 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.
31. H. Vollmer. *Introduction to Circuit Complexity*. Springer, Berlin, 1999.
32. S. Waack. Tape complexity of word problems. In F. Gécseg, editor, *Proceedings of Fundamentals of Computation Theory (FCT'81)*, volume 117 of *Lecture Notes in Computer Science*, pages 467–471. Springer, 1981.
33. B. A. F. Wehrfritz. Generalized free products of linear groups. *Proc. LMS*, 27:402–424, 1973.