# Compressed Conjugacy and the Word Problem for Outer Automorphism Groups of Graph Groups

Niko Haubold, Markus Lohrey, and Christian Mathissen

Institut für Informatik, Universität Leipzig, Germany
{haubold,lohrey,mathissen}@informatik.uni-leipzig.de

**Abstract.** It is shown that for graph groups (right-angled Artin groups) the conjugacy problem as well as a restricted version of the simultaneous conjugacy problem can be solved in polynomial time even if input words are represented in a compressed form. As a consequence it follows that the word problem for the outer automorphism group of a graph group can be solved in polynomial time.

## 1 Introduction

*Automorphism groups* and *outer automorphism groups* of *graph groups* received a lot of interest in the past few years. A graph group $\mathbb{G}(\Sigma, I)$ is given by a finite undirected graph $(\Sigma, I)$ (without self-loops). The set $\Sigma$ is the set of generators of $\mathbb{G}(\Sigma, I)$ and every edge $(a, b) \in I$ gives rise to a commutation relation $ab = ba$. Graph groups are also known as *right-angled Artin groups* or *free partially commutative groups*. Graph groups interpolate between finitely generated free groups and finitely generated free Abelian groups. The automorphism group of the free Abelian group $\mathbb{Z}^n$ is $\mathsf{GL}(n, \mathbb{Z})$ and hence finitely generated. By a classical result of Nielsen, also automorphism groups of free groups are finitely generated, see e.g. [14]. For graph groups in general, it was shown by Laurence [10] (building up on previous work by Servatius [19]) that their automorphism groups are finitely generated. Only recently, Day [4] has shown that $\mathsf{Aut}(\mathbb{G}(\Sigma, I))$ is always finitely presented. An overwiew on structural results on automorphism groups of graph groups can be found in [1].

In this paper, we continue the investigation of algorithmic aspects of automorphism groups of graph groups. In [13] it was shown that the word problem for $\mathsf{Aut}(\mathbb{G}(\Sigma, I))$ can be solved in polynomial time. The proof of this result used compression techniques. It is well-known that the word problem for $\mathbb{G}(\Sigma, I)$ can be solved in linear time. In [13], a compressed (or succinct) version of the word problem for graph groups was studied. In this variant of the word problem, the input word is represented succinctly by a so-called *straight-line program*. This is a context free grammar $\mathbb{A}$ that generates exactly one word $\mathsf{val}(\mathbb{A})$, see Sec. 2.1. Since the length of this word may grow exponentially with the size (number of productions) of the SLP $\mathbb{A}$, SLPs can be seen indeed as a succinct string representation. SLPs turned out to be a very flexible compressed representation of strings, which are well suited for studying algorithms for compressed data, see e.g. [6, 11, 17]. In [13, 18] it was shown that the word problem for the automorphism group $\mathsf{Aut}(G)$ of a group $G$ can be reduced in polynomial time to the *compressed word problem* for $G$, where the input word is succinctly given by an SLP. In [18], it was shown that the compressed word problem for a finitely generated free group $F$ can be solved in polynomial time and in [13] this result was extended to graph groups. It follows that the word problem for $\mathsf{Aut}(\mathbb{G}(\Sigma, I))$ can be solved in polynomial time. Recently, Macdonald [15] has shown that also the compressed word problem for every fully residually free group can be solved in polynomial time.

It is not obvious to carry over these complexity results from $\mathsf{Aut}(\mathbb{G}(\Sigma, I))$ to the *outer* automorphism group $\mathsf{Out}(\mathbb{G}(\Sigma, I)) = \mathsf{Aut}(\mathbb{G}(\Sigma, I))/\mathrm{Inn}(\mathbb{G}(\Sigma, I))$ (see Sec. 2.3 for the definition). Nevertheless, Schleimer proved in [18] that the word problem for the outer automorphism group of a finitely generated free group can be decided in polynomial time. For this, he used a compressed variant of the simultaneous conjugacy problem in free groups. In this paper, we generalize Schleimer's result to graph groups: For every graph $(\Sigma, I)$, the word problem for $\mathsf{Out}(\mathbb{G}(\Sigma, I))$ can be solved in polynomial time. Analogously to Schleimer's approach for free groups, we reduce the word problem for $\mathsf{Out}(\mathbb{G}(\Sigma, I))$ to a compressed variant of the simultaneous conjugacy problem in $\mathbb{G}(\Sigma, I)$. In this problem, we are given an SLP $\mathbb{A}_a$ for every generator $a \in \Sigma$, and the question is whether there exists $x \in \mathbb{G}(\Sigma, I)$ such that $a = x\,\mathsf{val}(\mathbb{A}_a)\,x^{-1}$ for all $a \in \Sigma$. A large part of this paper develops a polynomial time algorithm for this problem. Moreover, we also present a polynomial time algorithm for the compressed version of the classical conjugacy problem in graph groups: In this problem, we are given two SLPs $\mathbb{A}$ and $\mathbb{B}$ and we ask whether there exists $x \in \mathbb{G}(\Sigma, I)$ such that $\mathsf{val}(\mathbb{A}) = x\,\mathsf{val}(\mathbb{B})x^{-1}$ in $\mathbb{G}(\Sigma, I)$. For the non-compressed version of this problem, a linear time algorithm was presented in [21] based on [12]. In [3] this result was generalized to various subgroups of graph groups.

Missing proofs can be found in the full version [9] of this extended abstract.

## 2 Preliminaries

Let $\Sigma$ be a finite alphabet. For a word $s = a_1 \cdots a_m$ ($a_i \in \Sigma$) let $|s| = m$, $\mathsf{alph}(s) = \{a_1, \ldots, a_m\}$, $s[i] = a_i$ for $1 \leq i \leq m$, and $|s|_a = |\{k \mid s[k] = a\}|$ for $a \in \Sigma$. We use $\Sigma^{-1} = \{a^{-1} \mid a \in \Sigma\}$ to denote a disjoint copy of $\Sigma$ and let $\Sigma^{\pm 1} = \Sigma \cup \Sigma^{-1}$. Define $(a^{-1})^{-1} = a$. This defines an involution $^{-1} : \Sigma^{\pm 1} \to \Sigma^{\pm 1}$, which can be extended to an involution on $(\Sigma^{\pm 1})^*$ by setting $(a_1 \cdots a_n)^{-1} = a_n^{-1} \cdots a_1^{-1}$.

### 2.1 Straight-line programs

We are using straight-line programs as a succinct representation of strings with reoccurring subpatterns. A *straight-line program (SLP) over the alphabet $\Gamma$* is a context free grammar $\mathbb{A} = (V, \Gamma, S, P)$, where $V$ is the set of *nonterminals*, $\Gamma$ is the set of *terminals*, $S \in V$ is the *initial nonterminal*, and $P \subseteq V \times (V \cup \Gamma)^*$ is the set of *productions* such that (i) for every $X \in V$ there is exactly one $\alpha \in (V \cup \Gamma)^*$ with $(X, \alpha) \in P$ and (ii) there is no cycle in the relation $\{(X, Y) \in V \times V \mid \exists \alpha : (X, \alpha) \in P, Y \in \mathsf{alph}(\alpha)\}$. These conditions ensure that the language generated by the straight-line program $\mathbb{A}$ contains exactly one word $\mathsf{val}(\mathbb{A})$. Moreover, every nonterminal $X \in V$ generates exactly one word that is denoted by $\mathsf{val}_{\mathbb{A}}(X)$, or briefly $\mathsf{val}(X)$, if $\mathbb{A}$ is clear from the context.

The size of $\mathbb{A}$ is $|\mathbb{A}| = \sum_{(X,\alpha)\in P} |\alpha|$. An SLP can be transformed in polynomial time into an equivalent SLP in *Chomsky normal form*, which means that all productions have the form $A \to BC$ or $A \to a$ with $A, B, C \in V$ and $a \in \Gamma$. For an SLP $\mathbb{A}$ over $\Sigma^{\pm 1}$ (w.l.o.g. in Chomsky normal form) we denote with $\mathbb{A}^{-1}$ the SLP that has for each terminal rule $A \to a$ from $\mathbb{A}$ the terminal rule $A \to a^{-1}$ and for each nonterminal rule $A \to BC$ from $\mathbb{A}$ the nonterminal rule $A \to CB$. Clearly, $\mathsf{val}(\mathbb{A}^{-1}) = \mathsf{val}(\mathbb{A})^{-1}$. Let us state some simple algorithmic problems that can be easily solved in polynomial time:

– Given an SLP $\mathbb{A}$, calculate $|\mathsf{val}(\mathbb{A})|$ and $\mathsf{alph}(\mathsf{val}(\mathbb{A}))$.
– Given an SLP $\mathbb{A}$ and a number $i \in \{1, \ldots, |\mathsf{val}(\mathbb{A})|\}$, calculate $\mathsf{val}(\mathbb{A})[i]$.
– Given an SLP $\mathbb{A}$ (let $\mathsf{val}(\mathbb{A}) = a_1 \cdots a_n$) and two numbers $1 \le i \le j \le n$, compute and SLP $\mathbb{B}$ with $\mathsf{val}(\mathbb{B}) = a_i \cdots a_j$.

In [17], Plandowski presented a polynomial time algorithm for testing whether $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$ for two given SLPs $\mathbb{A}$ and $\mathbb{B}$. A cubic algorithm was presented by Lifshits [11]. In fact, Lifshits gave an algorithm for compressed pattern matching: given SLPs $\mathbb{A}$ and $\mathbb{B}$, is $\mathsf{val}(\mathbb{A})$ a factor of $\mathsf{val}(\mathbb{B})$? His algorithm runs in time $O(|\mathbb{A}| \cdot |\mathbb{B}|^2)$.

## 2.2 Trace monoids and graph groups

We introduce some notions from trace theory, see [5] for more details. An *independence alphabet* is a pair $(\Sigma, I)$ where $\Sigma$ is a finite alphabet and $I \subseteq \Sigma \times \Sigma$ is an irreflexive and symmetric relation. The complementary graph $(\Sigma, D)$ with $D = (\Sigma \times \Sigma) \setminus I$ is called a *dependence alphabet*. The *trace monoid* $\mathbb{M}(\Sigma, I)$ is defined as the quotient $\mathbb{M}(\Sigma, I) = \Sigma^*/\{ab = ba \mid (a,b) \in I\}$ with concatenation as operation and the empty word as the neutral element. This monoid is cancellative and its elements are called *traces*. The trace represented by the word $s \in \Sigma^*$ is denoted by $[s]_I$. For $a \in \Sigma$ let $I(a) = \{b \in \Sigma \mid (a,b) \in I\}$ be the letters that commute with $a$. For traces $u, v$ we denote with $uIv$ the fact that $\mathsf{alph}(u) \times \mathsf{alph}(v) \subseteq I$. For $\Gamma \subseteq \Sigma$ we say that $\Gamma$ is *connected* if the subgraph of $(\Sigma, D)$ induced by $\Gamma$ is connected. For a trace $u$ let $\max(u) = \{a \mid u = va \text{ for } a \in \Sigma, v \in \mathbb{M}(\Sigma, I)\}$ be the set of possible last letters of $u$ and $\min(u) = \{a \mid u = av \text{ for } a \in \Sigma, v \in \mathbb{M}(\Sigma, I)\}$ be the set of possible first letters.

A convenient representation for traces are *dependence graphs*, which are node-labeled directed acyclic graphs. For a word $w \in \Sigma^*$ the dependence graph $D_w$ has vertex set $\{1, \ldots, |w|\}$ where the node $i$ is labeled with $w[i]$. There is an edge from vertex $i$ to $j$ if and only if $i < j$ and $(w[i], w[j]) \in D$. It is easy to see that for two words $w, w' \in \Sigma^*$ we have $[w]_I = [w']_I$ if and only if $D_w$ and $D_{w'}$ are isomorphic. Hence, we can speak of *the* dependence graph of a trace.

For background in combinatorial group theory see [14]. The *free group* generated by $\Sigma$ can be defined as the quotient monoid $F(\Sigma) = (\Sigma^{\pm 1})^*/\{aa^{-1} = \varepsilon \mid a \in \Sigma^{\pm 1}\}$. For an independence alphabet $(\Sigma, I)$ the *graph group* $\mathbb{G}(\Sigma, I)$ is the quotient group $\mathbb{G}(\Sigma, I) = F(\Sigma)/\{ab = ba \mid (a,b) \in I\}$. From the independence alphabet $(\Sigma, I)$ we derive the independence alphabet $(\Sigma^{\pm 1}, \{(a^i, b^j) \mid i, j \in \{-1, 1\}, (a,b) \in I\})$. Abusing notation, we denote the independence relation of this alphabet again with $I$. Note that $(a,b) \in I$ implies $a^{-1}b = ba^{-1}$ in $\mathbb{G}(\Sigma, I)$. Thus, we have $\mathbb{G}(\Sigma, I) = \mathbb{M}(\Sigma^{\pm 1}, I)/\{aa^{-1} = \varepsilon \mid a \in \Sigma^{\pm 1}\}$. Graph groups are also known as right-angled Artin groups and free partially commutative groups.

### 2.3 (Outer) automorphism groups

The *automorphism group* $\mathsf{Aut}(G)$ of a group $G$ is the set of all automorphisms of $G$ with composition as operation and the identity mapping as the neutral element. An automorphism $\varphi$ is called *inner* if there is $x \in G$ such that $\varphi(y) = xyx^{-1}$ for all $y \in G$. The set of all inner automorphisms of $G$ forms the *inner automorphism group* $\mathsf{Inn}(G)$ of $G$. This is easily seen to be a normal subgroup of $\mathsf{Aut}(G)$ and the quotient group $\mathsf{Out}(G) = \mathsf{Aut}(G)/\mathsf{Inn}(G)$ is called the *outer automorphism group* of $G$.

Assume that $\mathsf{Aut}(G)$ is finitely generated (which, in general, won't be the case, even if $G$ is finitely generated) and let $\Psi = \{\psi_1, \dots, \psi_k\}$ be a monoid generating set for $\mathsf{Aut}(G)$, i.e., every automorphism of $G$ can be composed from the automorphisms in $\Psi$. Then $\Psi$ also generates $\mathsf{Out}(G)$ where we identify $\psi_i$ with its coset $\psi_i \cdot \mathsf{Inn}(G) \in \mathsf{Out}(G)$ for $i \in \{1, \dots, k\}$. Then the *word problem* for the outer automorphism group can be viewed as the following decision problem:

INPUT: A word $w \in \Psi^*$.
QUESTION: Does $w = 1$ in $\mathsf{Out}(G)$?

Since an automorphism belongs to the same coset (w.r.t. $\mathsf{Inn}(G)$) as the identity if and only if it is inner, we can rephrase the word problem for $\mathsf{Out}(G)$ as follows:

INPUT: A word $w \in \Psi^*$.
QUESTION: Does $w$ represent an element of $\mathsf{Inn}(G)$ in $\mathsf{Aut}(G)$?

Building on results from [19], Laurence has shown in [10] that automorphism groups of graph groups are finitely generated. Recently, Day [4] proved that automorphism groups of graph groups are in fact finitely presented. In this paper, we present a polynomial time algorithm for the word problem for $\mathsf{Out}(\mathbb{G}(\Sigma, I))$.

## 3 Main results

In this section we will present the main results of this paper, the proofs of which are subject to the rest of the paper. Our group theoretical main result is:

**Theorem 1.** *Let $(\Sigma, I)$ be a fixed independence alphabet. Then, the word problem for the group $\mathsf{Out}(\mathbb{G}(\Sigma, I))$ can be solved in polynomial time.*

In order to solve the word problem for $\mathsf{Out}(\mathbb{G}(\Sigma, I))$ in polynomial time, we will consider (following Schleimer's approach for free groups [18]) compressed conjugacy problems in $\mathbb{G}(\Sigma, I)$. The most general of these compressed conjugacy problems is the *simultaneous compressed conjugacy problem* for $\mathbb{G}(\Sigma, I)$:

INPUT: SLPs $\mathbb{A}_1, \mathbb{B}_1, \dots, \mathbb{A}_n, \mathbb{B}_n$ over $\Sigma^{\pm 1}$.
QUESTION: $\exists x \in (\Sigma^{\pm 1})^* \; \forall i \in \{1, \dots, n\} : \mathsf{val}(\mathbb{A}_i) = x \, \mathsf{val}(\mathbb{B}_i) x^{-1}$ in $\mathbb{G}(\Sigma, I)$?

The simultaneous (non-compressed) conjugacy problem also appears in connection with group-based cryptography [16]. Unfortunately, we don't know, whether the simultaneous compressed conjugacy problem can be solved in polynomial time. But, in order to deal with the word problem for $\mathsf{Out}(\mathbb{G}(\Sigma, I))$, a restriction of this problem suffices, where the SLPs $\mathbb{B}_1, \dots, \mathbb{B}_n$ from the simultaneous compressed conjugacy problem are the letters from $\Sigma$. We call this problem the *restricted simultaneous compressed conjugacy problem*, briefly $\mathsf{RSCCP}(\Sigma, I)$:

INPUT: SLPs $\mathbb{A}_a$ ($a \in \Sigma$) over $\Sigma^{\pm 1}$.
QUESTION: $\exists x \in (\Sigma^{\pm 1})^* \; \forall a \in \Sigma : \mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}(\Sigma, I)$?

An $x$ such that $\mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}(\Sigma, I)$ for all $a \in \Sigma$ is called a *solution* of the $\mathsf{RSCCP}(\Sigma, I)$-instance. The following theorem will be shown in Sec. 5:

**Theorem 2.** *Let $(\Sigma, I)$ be a fixed independence alphabet. Then, $\mathsf{RSCCP}(\Sigma, I)$ can be solved in polynomial time. Moreover, in case a solution exists, one can compute an SLP for a solution in polynomial time.*

*Proof of Thm. 1 using Thm. 2.* Fix a finite monoid generating set $\Phi$ for $\mathsf{Aut}(\mathbb{G}(\Sigma, I))$. Let $\varphi = \varphi_1 \cdots \varphi_n$ with $\varphi_1, \dots, \varphi_n \in \Phi$ be the input. By [18] we can compute in polynomial time SLPs $\mathbb{A}_a$ ($a \in \Sigma$) over $\Sigma^{\pm 1}$ with $\mathsf{val}(\mathbb{A}_a) = \varphi(a)$ in $\mathbb{G}(\Sigma, I)$ for all $a \in \Sigma$. The automorphism $\varphi$ is inner iff there exists $x$ such that $\mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}(\Sigma, I)$ for all $a \in \Sigma$. This can be decided in polynomial time by Thm. 2. $\square$

Finally, we will also consider a compressed variant of the classical conjugacy problem for $\mathbb{G}(\Sigma, I)$. Recall that the *conjugacy problem* for a finitely generated group $G$ asks, whether two given elements $g, h \in G$ are *conjugated*, i.e., whether there exists $x \in G$ with $g = xhx^{-1}$. The *compressed conjugacy problem* for the graph group $\mathbb{G}(\Sigma, I)$, $\mathsf{CCP}(\Sigma, I)$ for short, is the following problem:

INPUT: SLPs $\mathbb{A}$ and $\mathbb{B}$ over $\Sigma^{\pm 1}$.
QUESTION: Are $\mathsf{val}(\mathbb{A})$ and $\mathsf{val}(\mathbb{B})$ conjugated in $\mathbb{G}(\Sigma, I)$?

**Theorem 3.** *Let $(\Sigma, I)$ be a fixed independence alphabet. Then, $\mathsf{CCP}(\Sigma, I)$ can be solved in polynomial time.*

We will prove Thm. 3 in Sec. 7. It is important in Thm. 1–3 that we fix the independence alphabet $(\Sigma, I)$. It is open whether these results also hold if $(\Sigma, I)$ is part of the input.

## 4 Further facts for traces

In this section, we state some simple facts on trace monoids, which will be needed later. Fix the trace monoid $\mathbb{M}(\Sigma, I)$. A trace $u$ is said to be a *prefix* of a trace $w$, briefly $u \preceq w$, if $uv = w$ for some trace $v$. The prefixes of a trace $w$ correspond to the downward-closed node sets of the dependence graph of $w$. Analogously, a trace $v$ is a *suffix* of a trace $w$ if $uv = w$ for some trace $u$. For two traces $u, v$, the *infimum* $u \sqcap v$ is the largest trace $s$ w.r.t. $\preceq$ such that $s \preceq u$ and $s \preceq v$; it always exists [2]. With $u \setminus v$ we denote the unique trace $t$ such that $u = (u \sqcap v)t$; uniqueness follows from the fact that $\mathbb{M}(\Sigma, I)$ is cancellative. Note that $u \setminus v = u \setminus (u \sqcap v)$. The *supremum* $u \sqcup v$ of two traces $u, v$ is the smallest trace $s$ w.r.t. $\preceq$ such that $u \preceq s$ and $v \preceq s$ if any such trace exists. We can define the supremum of several traces $w_1, \dots, w_n$ by induction: $w_1 \sqcup \cdots \sqcup w_n = (w_1 \sqcup \cdots \sqcup w_{n-1}) \sqcup w_n$.

**Lemma 4 ([2]).** *The trace $u \sqcup v$ exists if and only if $(u \setminus v) \, I \, (v \setminus u)$, in which case we have $u \sqcup v = (u \sqcap v)\,(u \setminus v)\,(v \setminus u)$.*

*Example 5.* We consider the following independence alphabet $(\Sigma, I)$ and the corresponding dependence alphabet $(\Sigma, D)$:

$$(\Sigma, I)\ \begin{tikzcd} c \arrow[r, dash] & a \arrow[dl, dash] \arrow[d, dash] & \\ e \arrow[r, dash] & d \arrow[r, dash] & b \end{tikzcd} \qquad (\Sigma, D)\ \begin{tikzcd} a \arrow[dr, dash] & & e \arrow[dl, dash] \arrow[d, dash] & \\ & b \arrow[r, dash] & c \arrow[r, dash] & d \end{tikzcd}$$

Consider the words $u = aeadbacdd$ and $v = eaabdcaeb$. The dependence graphs $D_u$ and $D_v$ look as follows, where we label the vertex $i$ with the letter $u[i]$ (resp. $v[i]$):

$$D_u\ \begin{tikzcd} a \arrow[r] & a \arrow[dr] & & a & & \\ e \arrow[rr] & & b \arrow[ur] \arrow[dr] & & & \\ d \arrow[rrr] & & & c \arrow[r] & d \arrow[r] & d \end{tikzcd} \qquad D_v\ \begin{tikzcd} a \arrow[r] & a \arrow[dr] & & a \arrow[r] & b \\ e \arrow[rr] & & b \arrow[ur] \arrow[dr] & & e \arrow[u] \\ d \arrow[rrr] & & & c \arrow[ur] & \end{tikzcd}$$

Then we have $u \sqcap v = aeadbac =: p$. Since $u \setminus p = dd$ and $v \setminus p = eb$ we have $(u \setminus p)I(v \setminus p)$ and hence the supremum $s = u \sqcup v = aeadbacddeb$ is defined. The dependence graphs for $p$ and $s$ are:

$$D_p\ \begin{tikzcd} a \arrow[r] & a \arrow[dr] & & a \\ e \arrow[rr] & & b \arrow[ur] \arrow[dr] & \\ d \arrow[rrr] & & & c \end{tikzcd} \qquad D_s\ \begin{tikzcd} a \arrow[r] & a \arrow[dr] & & a \arrow[r] & b & \\ e \arrow[rr] & & b \arrow[ur] \arrow[dr] & & e \arrow[u] & \\ d \arrow[rrr] & & & c \arrow[ur] \arrow[r] & d \arrow[r] & d \end{tikzcd}$$

The following lemma is a basic statement for traces, see for example [5, Sec. 1.3]:

**Lemma 6 (Levi's Lemma).** *Let $u_1, u_2, v_1, v_2$ be traces with $u_1u_2 = v_1v_2$. Then there exist traces $x, y_1, y_2, z$ such that $y_1 I y_2$, $u_1 = xy_1$, $u_2 = y_2z$, $v_1 = xy_2$, and $v_2 = y_1z$.*

A *trace rewriting system* $R$ over $\mathbb{M}(\Sigma, I)$ is just a finite subset of $\mathbb{M}(\Sigma, I) \times \mathbb{M}(\Sigma, I)$ [5]. The *one-step rewrite relation* $\to_R \subseteq \mathbb{M}(\Sigma, I) \times \mathbb{M}(\Sigma, I)$ is defined as: $x \to_R y$ if and only if there are $u, v \in \mathbb{M}(\Sigma, I)$ and $(\ell, r) \in R$ such that $x = u\ell v$ and $y = urv$. A trace $u$ is *$R$-irreducible* if no trace $v$ with $u \to_R v$ exists. The set of all $R$-irreducible traces is denoted by $\mathsf{IRR}(R)$. If $R$ is Noetherian and confluent (see e.g. [5, Sec. 5.1] for definitions), then for every trace $u$, there exists a unique *normal form* $\mathsf{NF}_R(u) \in \mathsf{IRR}(R)$ such that $u \xrightarrow{*}_R \mathsf{NF}_R(u)$.

Let us now work in the trace monoid $\mathbb{M}(\Sigma^{\pm 1}, I)$. For a trace $u = [a_1 \cdots a_n]_I \in \mathbb{M}(\Sigma^{\pm 1}, I)$ we denote with $u^{-1}$ the trace $u^{-1} = [a_n^{-1} \cdots a_1^{-1}]_I$. It is easy to see that this definition is independent of the chosen representative $a_1 \cdots a_n$ of the trace $u$. It follows that we have $[\mathsf{val}(\mathbb{A})]_I^{-1} = [\mathsf{val}(\mathbb{A}^{-1})]_I$ for an SLP $\mathbb{A}$. For the rest of the paper, we fix the trace rewriting system $R = \{([aa^{-1}]_I, [\varepsilon]_I) \mid a \in \Sigma^{\pm 1}\}$ over the trace monoid $\mathbb{M}(\Sigma^{\pm 1}, I)$. This system is Noetherian (trivial) and, by [5, 20], also confluent. For traces $u, v \in \mathbb{M}(\Sigma^{\pm 1}, I)$ we have $u = v$ in $\mathbb{G}(\Sigma, I)$ if and only if $\mathsf{NF}_R(u) = \mathsf{NF}_R(v)$. Using these facts, it was shown in [5, 20] that the word problem for $\mathbb{G}(\Sigma, I)$ can be solved in linear time (on the RAM model).

We close this section with some results concerning SLP-compressed traces. A simple observation is that for given SLPs $\mathbb{A}$ and $\mathbb{B}$ one can decide in polynomial time whether $[\mathsf{val}(\mathbb{A})]_I = [\mathsf{val}(\mathbb{B})]_I$. The projection lemma for traces [5, Cor. 1.4.8] allows to reduce this question to equality testing for SLP-compressed strings [17]. Much harder to prove is:

**Theorem 7 ([13]).** *For a given SLP $\mathbb{A}$ over $\Sigma^{\pm 1}$ one can compute in polynomial time an SLP $\mathbb{B}$ with $[\mathsf{val}(\mathbb{B})]_I = \mathsf{NF}_R([\mathsf{val}(\mathbb{A})]_I)$.*

Thm. 7 implies that the compressed word problem for a graph group can be solved in polynomial time.

**Theorem 8 ([13]).** *For given SLPs $\mathbb{A}_0$ and $\mathbb{A}_1$ over $\Sigma^{\pm 1}$, we can compute in polynomial time SLPs $\mathbb{P}$, $\mathbb{D}_0$, $\mathbb{D}_1$ with $[\mathsf{val}(\mathbb{P})]_I = [\mathsf{val}(\mathbb{A}_0)]_I \sqcap [\mathsf{val}(\mathbb{A}_1)]_I$ and $[\mathsf{val}(\mathbb{D}_i)]_I = [\mathsf{val}(\mathbb{A}_i)]_I \setminus [\mathsf{val}(\mathbb{A}_{1-i})]_I$ for $i \in \{0,1\}$.*

An immediate corollary of Thm. 8 and Lemma 4 is:

**Corollary 9.** *Let $r$ be a fixed constant. Then, for given SLPs $\mathbb{V}_1, \ldots, \mathbb{V}_r$ over $\Sigma^{\pm 1}$, one can check in polynomial time whether $[\mathsf{val}(\mathbb{V}_1)]_I \sqcup \cdots \sqcup [\mathsf{val}(\mathbb{V}_r)]_I$ exists, and in case it exists one can compute an SLP $\mathbb{S}$ with $[\mathsf{val}(\mathbb{S})]_I = [\mathsf{val}(\mathbb{V}_1)]_I \sqcup \cdots \sqcup [\mathsf{val}(\mathbb{V}_r)]_I$ in polynomial time.*

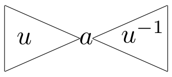
It is important that we fix the number $r$ of SLPs in Cor. 9: Each application of Thm. 8 increase the size of the SLP polynomially. Hence, a non-fixed number of applications might lead to an exponential blow-up.

## 5 Restricted simultaneous compressed conjugacy

A *double $a$-cone* ($a \in \Sigma^{\pm 1}$) is an $R$-irreducible trace of the form $uau^{-1}$ with $u \in \mathbb{M}(\Sigma^{\pm 1}, I)$. We first state several results on double $a$-cones, which will be used in the proof of Thm. 2. The following characterization can be easily shown:

**Lemma 10.** *A trace $uau^{-1}$ is a double $a$-cone if and only if $u \in \mathsf{IRR}(R)$ and $\max(u) \cap (\{a, a^{-1}\} \cup I(a)) = \emptyset$.*

It follows that every letter in a double $a$-cone either lies before or after the central letter $a$. Its dependence graph always has the following form:



By the following lemma, each double $a$-cone has a unique factorization of the form $u_1 b u_2$ with $|u_1| = |u_2|$.

**Lemma 11.** *Let $v = uau^{-1}$ be a double $a$-cone and let $v = u_1 b u_2$ with $b \in \Sigma^{\pm 1}$ and $|u_1| = |u_2|$. Then $a = b$, $u_1 = u$ and $u_2 = u^{-1}$.*

Lemma 11 together with standard techniques for SLP-compressed strings (in particular, the polynomial equality test for SLP-compressed strings [11, 17]) implies:

**Lemma 12.** *For a given SLP $\mathbb{A}$ with $[\mathsf{val}(\mathbb{A})]_I \in \mathsf{IRR}(R)$ and $a \in \Sigma^{\pm 1}$, one can check in polynomial time whether $[\mathsf{val}(\mathbb{A})]_I$ is a double $a$-cone, and in case it is, one can compute in polynomial time an SLP $\mathbb{V}$ with $[\mathsf{val}(\mathbb{A})]_I = [\mathsf{val}(\mathbb{V})\, a\, \mathsf{val}(\mathbb{V}^{-1})]_I$.*

**Lemma 13.** *Let $w \in \mathbb{M}(\Sigma^{\pm 1}, I)$ be $R$-irreducible and $a \in \Sigma^{\pm 1}$. Then there exists $x \in \mathbb{M}(\Sigma^{\pm 1}, I)$ with $w = xax^{-1}$ in $\mathbb{G}(\Sigma, I)$ if and only if $w$ is a double $a$-cone.*

Lemma 13 can be shown by induction on the number of $R$-rewrite steps from $xax^{-1}$ to $w \in \mathsf{IRR}(R)$. Finally, our main lemma on double $a$-cones is:

**Lemma 14.** *Let $w_a, v_a \in \mathbb{M}(\Sigma^{\pm 1}, I)$ ($a \in \Sigma$) be $R$-irreducible such that $w_a = v_a a v_a^{-1}$ in $\mathbb{M}(\Sigma^{\pm 1}, I)$ for all $a \in \Sigma$ (thus, every $w_a$ is a double $a$-cone). If there is a trace $x \in \mathbb{M}(\Sigma^{\pm 1}, I)$ with $\forall a \in \Sigma : xax^{-1} = w_a$ in $\mathbb{G}(\Sigma, I)$, then $s = \bigsqcup_{a \in \Sigma} v_a$ exists and $sas^{-1} = w_a$ in $\mathbb{G}(\Sigma, I)$ for all $a \in \Sigma$.*

Now we are in the position to prove Thm. 2: Let $\mathbb{A}_a$ ($a \in \Sigma$) be the input SLPs. We have to check whether there exists $x$ such that $\mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}(\Sigma, I)$ for all $a \in \Sigma$. Thm. 7 allows us to assume that $[\mathsf{val}(\mathbb{A}_a)]_I \in \mathsf{IRR}(R)$ for all $a \in \Sigma$. We first check whether every trace $[\mathsf{val}(\mathbb{A}_a)]_I$ is a double $a$-cone. By Lemma 12 this is possible in polynomial time. If there exists $a \in \Sigma$ such that $[\mathsf{val}(\mathbb{A}_a)]_I$ is not a double $a$-cone, then we can reject by Lemma 13. Otherwise, we can compute (using again Lemma 12) SLPs $\mathbb{V}_a$ ($a \in \Sigma$) such that $[\mathsf{val}(\mathbb{A}_a)]_I = [\mathsf{val}(\mathbb{V}_a)\, a\, \mathsf{val}(\mathbb{V}_a^{-1})]_I$ in $\mathbb{M}(\Sigma^{\pm 1}, I)$. Finally, by Lemma 14, it suffices to check whether $s = \bigsqcup_{a \in \Sigma} [\mathsf{val}(\mathbb{V}_a)]_I$ exists and whether $\mathsf{NF}_R(sas^{-1}) = [\mathsf{val}(\mathbb{A}_a)]_I$ for all $a \in \Sigma$. This is possible in polynomial time by Thm. 7 and Cor. 9 (recall that $|\Sigma|$ is a constant in our consideration). Moreover, if the supremum $s$ exists, then we can compute in polynomial time an SLP $\mathbb{S}$ with $[\mathsf{val}(\mathbb{S})]_I = s$, which is a solution for our $\mathsf{RSCCP}(\Sigma, I)$-instance. $\square$

## 6 A pattern matching algorithm for connected patterns

For the proof of Thm. 3 we need a pattern matching algorithm for compressed traces. For traces $v, w$ we say that $v$ is a factor of $w$ if there are traces $x, y$ with $w = xvy$. We consider the following problem and show that it can be solved in polynomial time if the independence alphabet $(\Sigma, I)$ satisfies certain conditions.

INPUT: An independence alphabet $(\Sigma, I)$ and two SLPs $\mathbb{T}$ and $\mathbb{P}$ over $\Sigma$.
QUESTION: Is $[\mathsf{val}(\mathbb{P})]_I$ a factor of $[\mathsf{val}(\mathbb{T})]_I$?

We write $\mathsf{alph}(\mathbb{T})$ and $\mathsf{alph}(\mathbb{P})$ for $\mathsf{alph}(\mathsf{val}(\mathbb{T}))$ and $\mathsf{alph}(\mathsf{val}(\mathbb{P}))$, respectively. We assume in the following that the SLPs $\mathbb{T} = (V, \Sigma, S, P)$ and $\mathbb{P}$ are in Chomsky normal form. Let $\Gamma \subseteq \Sigma$. We denote by $\pi_\Gamma$ the homomorphism $\pi_\Gamma : \mathbb{M}(\Sigma, I) \to \mathbb{M}(\Gamma, I \cap (\Gamma \times \Gamma))$ with $\pi_\Gamma(a) = a$ for $a \in \Gamma$ and $\pi_\Gamma(a) = \varepsilon$ for $a \in \Sigma \setminus \Gamma$. Let $V^\Gamma = \{X^\Gamma \mid X \in V\}$ be a disjoint copy of $V$. For each production $p \in P$ define a new production $p^\Gamma$ as follows. If $p$ is of the form $X \to a$ ($a \in \Sigma$), then let $p^\Gamma = (X^\Gamma \to \pi_\Gamma(a))$. If $p \in P$ is of the form $X \to YZ$ ($Y, Z \in V$) define $p^\Gamma = (X^\Gamma \to Y^\Gamma Z^\Gamma)$. We denote with $\mathbb{T}^\Gamma$ the SLP $(V^\Gamma, \Gamma, S^\Gamma, P^\Gamma)$ where $P^\Gamma = \{p^\Gamma \mid p \in P\}$. Obviously, $\mathsf{val}(\mathbb{T}^\Gamma) = \pi_\Gamma(\mathsf{val}(\mathbb{T}))$.

In order to develop a polynomial time algorithm for the problem stated above we need a succinct representation for an occurrence of $\mathbb{P}$ in $\mathbb{T}$. Since $[\mathsf{val}(\mathbb{P})]_I$ is a factor of $[\mathsf{val}(\mathbb{T})]_I$ iff there is a prefix $u \preceq [\mathsf{val}(\mathbb{T})]_I$ such that $u[\mathsf{val}(\mathbb{P})]_I \preceq [\mathsf{val}(\mathbb{T})]_I$, we will in fact compute prefixes with the latter property and represent a prefix $u$ by its Parikh image $(|u|_a)_{a \in \Sigma}$. Hence, we say a sequence $O = (O_a)_{a \in \Sigma} \in \mathbb{N}^\Sigma$ is an *occurrence* of a trace $v$ in a trace $w$ iff there is a prefix $u \preceq w$ such that $uv \preceq w$, and $O = (|u|_a)_{a \in \Sigma}$. Note that our definition of an occurrence of $\mathbb{P}$ in $\mathbb{T}$ does not exactly correspond to the intuitive notion of an occurrence as a convex subset of the dependence graph of $[\mathsf{val}(\mathbb{T})]_I$.

In fact, to a convex subset of the dependence graph of $[\mathsf{val}(\mathbb{T})]_I$, which is isomorphic to the dependence graph of $[\mathsf{val}(\mathbb{P})]_I$, there might correspond several occurrences $O$, since for an $a \in \Sigma$ that is independent of $\mathsf{alph}(\mathbb{P})$ we might have several possibilities for the value $O_a$. However, if we restrict to letters that are dependent on $\mathsf{alph}(\mathbb{P})$, then our definition of an occurrence coincides with the intuitive notion. For $\Gamma \subseteq \Sigma$ we write $\pi_\Gamma(O)$ for the restriction $(O_a)_{a\in\Gamma}$. Furthermore, we say that $O$ is an occurrence of $\mathbb{P}$ in $\mathbb{T}$ if $O$ is an occurrence of $[\mathsf{val}(\mathbb{P})]_I$ in $[\mathsf{val}(\mathbb{T})]_I$.

Let $X$ be a nonterminal of $\mathbb{T}$ with production $X \to YZ$ and let $O$ be an occurrence of $[\mathsf{val}(\mathbb{P})]_I$ in $[\mathsf{val}(X)]_I$. If there are $a, b \in \mathsf{alph}(\mathbb{P})$ with $O_a < |\mathsf{val}(Y)|_a$ and $O_b + |\mathsf{val}(\mathbb{P})|_b > |\mathsf{val}(Y)|_b$, then we say that $O$ is an occurrence of $\mathbb{P}$ *at the cut* of $X$. Assume w.l.o.g. that $|\mathsf{val}(\mathbb{P})| \geq 2$, otherwise we simply have to check whether a certain letter occurs in $\mathsf{val}(\mathbb{T})$. By this assumption, $[\mathsf{val}(\mathbb{P})]_I$ is a factor of $[\mathsf{val}(\mathbb{T})]_I$ iff there is a nonterminal $X$ of $\mathbb{T}$ for which there is an occurrence of $\mathbb{P}$ at the cut of $X$.

**Lemma 15 ([12]).** *Let $v, w \in \mathbb{M}(\Sigma, I)$. A sequence $(n_a)_{a\in\Sigma} \in \mathbb{N}^\Sigma$ is an occurrence of $v$ in $w$ iff $(n_a, n_b)$ is an occurrence of $\pi_{\{a,b\}}(v)$ in $\pi_{\{a,b\}}(w)$ for all $(a, b) \in D$.*

An *arithmetic progression* is a subset of $\mathbb{N}^\Sigma$ of the form $\{(i_a)_{a\in\Sigma} + k \cdot (d_a)_{a\in\Sigma} \mid 0 \leq k \leq \ell\}$. This set can be represented by the triple $((i_a)_{a\in\Sigma}, (d_a)_{a\in\Sigma}, \ell)$. The *descriptional size* $|((i_a)_{a\in\Sigma}, (d_a)_{a\in\Sigma}, \ell)|$ of the arithmetic progression $((i_a)_{a\in\Sigma}, (d_a)_{a\in\Sigma}, \ell)$ is $\log_2(\ell) + \sum_{a\in\Sigma}(\log_2(i_a) + \log_2(d_a))$. We will use Lemma 15 in order to compute the occurrences of $\mathbb{P}$ in $\mathbb{T}$ in form of a family of arithmetic progressions. To this aim, we follow a similar approach as Genest and Muscholl for message sequence charts [7]. In particular Lemma 16 below was inspired by [7, Prop. 1]. For the rest of this section we make the following assumption:

$$\mathsf{alph}(\mathbb{P}) = \mathsf{alph}(\mathbb{T}) = \Sigma \text{ is connected.} \tag{1}$$

Whereas $\mathsf{alph}(\mathbb{P}) = \mathsf{alph}(\mathbb{T})$ is a real restriction, the assumption that $\Sigma = \mathsf{alph}(\mathbb{T})$ is connected is not a real restriction; we simply solve several pattern matching instances if it is not satisfied. Let $X$ be a nonterminal of $\mathbb{T}$ and let $O$ be an occurrence of $\mathbb{P}$ at the cut of $X$. Since the pattern is connected there must be some $(a, b) \in D$ such that $\pi_{\{a,b\}}(O)$ is at the cut of $X^{\{a,b\}}$. We will therefore compute occurrences of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X^{\{a,b\}}$. It is well known that the occurrences of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X^{\{a,b\}}$ form an arithmetic progression $((i_a, i_b), (d_a, d_b), \ell)$ and that $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ is of the form $u^n v$ for some $n \geq \ell$ and strings $u, v \in \{a, b\}^*$ with $v \preceq u$, $|u|_a = d_a$ and $|u|_b = d_b$. Moreover, the arithmetic progression $((i_a, i_b), (d_a, d_b), \ell)$ can be computed in time $|\mathbb{T}|^2|\mathbb{P}|$ (see [11][1]). Now suppose we have computed the occurrences of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X^{\{a,b\}}$ in form of an arithmetic progression. The problem now is how to find (for the possibly exponentially many occurrences in the arithmetic progression) matching occurrences of projections onto all other pairs in $D$. The following lemma states that either there is a pair $(a, b) \in D$ such that the projection onto $\{a, b\}$ is the first or the last element of an arithmetic progression, or all projections lie at the cut of the same nonterminal.

[1] In fact, in [11] it was shown that the arithmetic progression $(i_a+i_b, d_a+d_b, \ell)$ can be computed in polynomial time. From this the arithmetic progression, $((i_a, i_b), (d_a, d_b), \ell)$ can easily be computed.

**Lemma 16.** *Let $X$ be a nonterminal of $\mathbb{T}$ and let $O$ be an occurrence of $\mathbb{P}$ at the cut of $X$. Then either (i) $\pi_{\{a,b\}}(O)$ is at the cut of $X^{\{a,b\}}$ for all $(a,b) \in D$ with $a \neq b$, or (ii) there are $(a,b) \in D$ such that $\pi_{\{a,b\}}(O)$ is the first or last element of the arithmetic progression of occurrences of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X^{\{a,b\}}$.*

Lemma 16 motivates that we partition the set of occurrences into two sets. Let $O$ be an occurrence of $\mathbb{P}$ in $\mathbb{T}$ at the cut of $X$. We call $O$ *single* (for $X$) if there is $(a,b) \in D$ such that the projection $\pi_{\{a,b\}}(O)$ is the first or the last element of the arithmetic progression of occurrences of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X^{\{a,b\}}$. Otherwise, we call $O$ *periodic* (for $X$). By Lemma 16, if $O$ is periodic, then $\pi_{\{a,b\}}(O)$ is an element of the arithmetic progression of occurrences of $\mathsf{val}(\mathbb{P}^{\{a,b\}})$ at the cut of $X^{\{a,b\}}$ for all $(a,b) \in D$ (but neither the first nor the last element). Prop. 17 below shows that we can decide in polynomial time whether there are single occurrences of $\mathbb{P}$ in $\mathbb{T}$. The basic idea is that due to assumption (1), an occurrence of $\mathsf{val}(\mathbb{P})$ in $\mathsf{val}(\mathbb{T})$ is completely determined as soon as we have determined the position of a single node of the dependence graph of $[\mathsf{val}(\mathbb{P})]_I$ in the dependence graph of $[\mathsf{val}(\mathbb{T})]_I$.

**Proposition 17.** *Given $(a,b) \in D$, a nonterminal $X$ of $\mathbb{T}$ and an occurrence $(O_a, O_b)$ of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X^{\{a,b\}}$, one can decide in time $(|\mathbb{T}| + |\mathbb{P}|)^{O(1)}$ whether this occurrence is a projection of an occurrence of $\mathbb{P}$ at the cut of $X$.*

It remains to show that for every nonterminal $X$ of $\mathbb{T}$ we can compute in polynomial time the periodic occurrences. To this aim we define the amalgamation of arithmetic progressions. Let $\Gamma, \Gamma' \subseteq \Sigma$ with $\Gamma \cap \Gamma' \neq \emptyset$. Consider two arithmetic progressions $p = ((i_a)_{a\in\Gamma}, (d_a)_{a\in\Gamma}, \ell)$ and $p' = ((i'_a)_{a\in\Gamma'}, (d'_a)_{a\in\Gamma'}, \ell')$. The *amalgamation* of $p$ and $p'$ is $p \otimes p' = \{v = (v_a)_{a\in\Gamma\cup\Gamma'} \mid \pi_\Gamma(v) \in p$ and $\pi_{\Gamma'}(v) \in p'\}$. The following lemma follows from elementary facts about simultaneous congruences:

**Lemma 18.** *Let $\Gamma, \Gamma' \subseteq \Sigma$ with $\Gamma \cap \Gamma' \neq \emptyset$, and let $p = ((i_a)_{a\in\Gamma}, (d_a)_{a\in\Gamma}, \ell)$ and $p' = ((i'_a)_{a\in\Gamma'}, (d'_a)_{a\in\Gamma'}, \ell')$ be two arithmetic progressions. Then $p \otimes p'$ is an arithmetic progression which can be computed in time $(|p| + |p'|)^{O(1)}$.*

The next proposition can be shown using Lemma 15 and 18.

**Proposition 19.** *Let $X$ be a nonterminal of $\mathbb{T}$. The periodic occurrences of $\mathbb{P}$ at the cut of $X$ form an arithmetic progression which can be computed in time $(|\mathbb{T}| + |\mathbb{P}|)^{O(1)}$.*

We now get the following theorem easily.

**Theorem 20.** *Given an independence alphabet $(\Sigma, I)$, and two SLPs $\mathbb{P}$ and $\mathbb{T}$ over $\Sigma$ such that $\mathsf{alph}(\mathbb{P}) = \mathsf{alph}(\mathbb{T})$, we can decide in polynomial time whether $[\mathsf{val}(\mathbb{P})]_I$ is a factor of $[\mathsf{val}(\mathbb{T})]_I$.*

*Proof.* Let $X$ be a nonterminal of $\mathbb{T}$. Using [11] we compute for each pair $(a,b) \in D$ the arithmetic progression of occurrences of $\pi_{a,b}(\mathsf{val}(\mathbb{P}))$ at the cut of $X^{\{a,b\}}$. By applying Prop. 17 to the first and to the last elements of each of these arithmetic progressions, we compute in polynomial time the single occurrences at the cut of $X$. The periodic occurrences can be computed in polynomial time using Prop. 19. The result follows, since $[\mathsf{val}(\mathbb{P})]_I$ is a factor of $[\mathsf{val}(\mathbb{T})]_I$ iff there is a nonterminal $X$ of $\mathbb{T}$ for which there is a single occurrence of $\mathbb{P}$ at the cut of $X$ or a periodic occurrence of $\mathbb{P}$ at the cut of $X$. □

In [9], a slight generalization of Theorem 20 is shown.

## 7 Compressed conjugacy

In order to prove Thm. 3 we need some further concepts from [20]. If for a trace $x$ we have $\mathsf{NF}_R(x) = uyu^{-1}$ in $\mathbb{M}(\Sigma^{\pm 1}, I)$ for traces $y, u$ such that $\min(y) \cap \min(y^{-1}) = \emptyset$, then we call $y$ the *core* of $x$, $\mathsf{core}(x)$ for short; it is uniquely defined [20]. Note that a trace $t$ is a double $a$-cone if and only if $t \in \mathsf{IRR}(R)$ and $\mathsf{core}(t) = a$. The following result, which follows by combining results from [12] and [21], allows us to transfer the conjugacy problem in $\mathbb{G}(\Sigma, I)$ to a problem on traces:

**Theorem 21 ([12, 21]).** *Let $u, v \in \mathbb{M}(\Sigma^{\pm 1}, I)$. Then $u$ is conjugated to $v$ in $\mathbb{G}(\Sigma, I)$ if and only if: (i) $|\mathsf{core}(u)|_a = |\mathsf{core}(v)|_a$ for all $a \in \Sigma^{\pm 1}$ and (ii) $\mathsf{core}(u)$ is a factor of $\mathsf{core}(v)^{2|\Sigma|}$.*

In order to apply Thm. 21 to SLP-compressed traces, we need a polynomial time algorithm for computing an SLP that represents $\mathsf{core}([\mathsf{val}(\mathbb{A})]_I)$ for a given SLP $\mathbb{A}$. The following lemma is crucial:

**Lemma 22.** *Let $x \in \mathsf{IRR}(R)$ and $d = x \sqcap x^{-1}$. Then $\mathsf{NF}_R(d^{-1}xd) = \mathsf{core}(x)$.*

*Example 23.* We take the independence alphabet from Example 5 and consider the trace $x = [c^{-1}d^{-1}a^{-1}ba^{-1}cabdc^{-1}d^{-1}a^{-1}b^{-1}dca]_I \in \mathbb{M}(\Sigma^{\pm 1}, I)$, whose dependence graph looks as follows:

$$\begin{tikzcd}
c^{-1} \arrow[rr] \arrow[dr] & & d^{-1} \arrow[r] & c \arrow[r] \arrow[dr] & d \arrow[r] & c^{-1} \arrow[r] \arrow[drr] & d^{-1} \arrow[r] \arrow[r, bend left, dashed, no head] & d \arrow[r] & c \\
a^{-1} \arrow[r] & b \arrow[r] \arrow[urr] & a^{-1} \arrow[r] \arrow[r, bend right, dashed, no head] & a \arrow[r] & b \arrow[r] \arrow[ur] & a^{-1} \arrow[rr] & & b^{-1} \arrow[r] \arrow[ur] & a
\end{tikzcd}$$

We have $\mathsf{NF}_R(x) = [c^{-1}d^{-1}a^{-1}bcbdc^{-1}a^{-1}b^{-1}ca]_I$:

$$\begin{tikzcd}
c^{-1} \arrow[rr] \arrow[dr] & & d^{-1} \arrow[r] & c \arrow[r] \arrow[dr] & d \arrow[r] & c^{-1} \arrow[drr] & & & c \\
a^{-1} \arrow[r] & b \arrow[urr] & & & b \arrow[r] \arrow[ur] & a^{-1} \arrow[rr] & & b^{-1} \arrow[r] \arrow[ur] & a
\end{tikzcd}$$

Hence, the core of $x$ is $\mathsf{core}(x) = [d^{-1}cbdc^{-1}a^{-1}]_I$ (the middle part in the above diagram). Note that we have $\mathsf{NF}_R(x) \sqcap \mathsf{NF}_R(x^{-1}) = c^{-1}a^{-1}b$. This trace occurs as a prefix of $\mathsf{NF}_R(x)$ and its inverse occurs as a suffix of $\mathsf{NF}_R(x)$. By cyclically cancelling $c^{-1}a^{-1}b$ in $\mathsf{NF}_R(x)$, we obtain $d^{-1}cbdc^{-1}a^{-1} = \mathsf{core}(x)$.

Thm. 7 and 8 and Lemma 22 imply:

**Corollary 24.** *Fix an independence alphabet $(\Sigma^{\pm 1}, I)$. Then, for a given SLP $\mathbb{A}$ over the alphabet $\Sigma^{\pm 1}$ one can compute in polynomial time an SLP $\mathbb{B}$ with $[\mathsf{val}(\mathbb{B})]_I = \mathsf{core}([\mathsf{val}(\mathbb{A})]_I)$.*

We can now infer Thm. 3: Let $\mathbb{A}$ and $\mathbb{B}$ be two given SLPs over $\Sigma^{\pm 1}$. We want to check, whether $\mathsf{val}(\mathbb{A})$ and $\mathsf{val}(\mathbb{B})$ represent conjugated elements of the graph group $\mathbb{G}(\Sigma, I)$. Using Cor. 24, we can compute in polynomial time SLPs $\mathbb{C}$ and $\mathbb{D}$ with $[\mathsf{val}(\mathbb{C})]_I = \mathsf{core}([\mathsf{val}(\mathbb{A})]_I)$ and $[\mathsf{val}(\mathbb{D})]_I = \mathsf{core}([\mathsf{val}(\mathbb{B})]_I)$. By Thm. 21, it suffices to check, whether (i) $|\mathsf{core}([\mathsf{val}(\mathbb{C})]_I)|_a = |\mathsf{core}([\mathsf{val}(\mathbb{D})]_I)|_a$ for all $a \in \Sigma^{\pm 1}$ and (ii) whether $\mathsf{core}([\mathsf{val}(\mathbb{C})]_I)$ is a factor of $\mathsf{core}([\mathsf{val}(\mathbb{D})]_I)^{2|\Sigma|}$. Condition (i) can be easily checked in polynomial time, since the number of occurrences of a symbol in a compressed string can be computed in polynomial time. Moreover, condition (ii) can be checked in polynomial time by Thm. 20, since (by condition (i)) we can assume that $\mathsf{alph}(\mathsf{val}(\mathbb{C})) = \mathsf{alph}(\mathsf{val}(\mathbb{D}))$. $\square$

## 8 Open problems

Though we have shown that some cases of the simultaneous compressed conjugacy problem for graph groups (see Sec. 3) can be decided in polynomial time, it remains unclear whether this holds also for the general case. It is also unclear to the authors, whether the general compressed pattern matching problem for traces, where we drop restriction $\mathsf{alph}(\mathbb{T}) = \mathsf{alph}(\mathbb{P})$, can be decided in polynomial time. Finally, it is not clear, whether Thm. 1–3 also hold if the independence alphabet is part of the input.

## References

1. R. Charney. An introduction to right-angled Artin groups. *Geometriae Dedicata*, 125:141–158, 2007.
2. R. Cori, Y. Métivier, and W. Zielonka. Asynchronous mappings and asynchronous cellular automata. *Information and Computation*, 106(2):159–202, 1993.
3. J. Crisp, E. Godelle, and B. Wiest. The conjugacy problem in right-angled Artin groups and their subgroups. *Journal of Topology*, 2(3), 2009.
4. M. B. Day. Peak reduction and finite presentations for automorphism groups of right-angled Artin groups. *Geometry & Topology*, 13(2):817–855, 2009.
5. V. Diekert. *Combinatorics on Traces*. LNCS 454. Springer, 1990.
6. L. Gasieniec, M. Karpinski, W. Plandowski, and W. Rytter. Efficient algorithms for Lempel-Ziv encoding. In *Proc. SWAT 1996*, LNCS 1097, 392–403. Springer, 1996.
7. B. Genest and A. Muscholl. Pattern matching and membership for hierarchical message sequence charts. *Theory of Computing Systems*, 42(4):536–567, 2008.
8. C. Hagenah. *Gleichungen mit regulären Randbedingungen über freien Gruppen*. PhD thesis, University of Stuttgart, Institut für Informatik, 2000.
9. N. Haubold, M. Lohrey, and C. Mathissen. Compressed conjugacy and the word problem for outer automorphism groups of graph groups. `http://arxiv.org/abs/1003.1233`, 2010.
10. M. R. Laurence. A generating set for the automorphism group of a graph group. *Journal of the London Mathematical Society. Second Series*, 52(2):318–334, 1995.
11. Y. Lifshits. Processing compressed texts: A tractability border. In *Proc. CPM 2007*, LNCS 4580, 228–240, Springer, 2007.
12. H.-N. Liu, C. Wrathall, and K. Zeger. Efficient solution to some problems in free partially commutative monoids. *Information and Computation*, 89(2):180–198, 1990.
13. M. Lohrey and S. Schleimer. Efficient computation in groups via compression. In *Proc. CSR 2007*, LNCS 4649, 249–258. Springer, 2007.
14. R. C. Lyndon and P. E. Schupp. *Combinatorial Group Theory*. Springer, 1977.
15. J. Macdonald. Compressed words and automorphisms in fully residually free groups. *International Journal of Algebra and Computation*, 2009. to appear.
16. A. Myasnikov, V. Shpilrain, and A. Ushakov. *Group-based Cryptography*. Birkhäuser, 2008.
17. W. Plandowski. Testing equivalence of morphisms on context-free languages. In *Proc. ESA'94*, LNCS 855, 460–470. Springer, 1994.
18. S. Schleimer. Polynomial-time word problems. *Commentarii Mathematici Helvetici*, 83(4):741–765, 2008.
19. H. Servatius. Automorphisms of graph groups. *Journal of Algebra*, 126(1):34–60, 1989.
20. C. Wrathall. The word problem for free partially commutative groups. *Journal of Symbolic Computation*, 6(1):99–104, 1988.
21. C. Wrathall. Free partially commutative groups. In *Combinatorics, computing and complexity*, pages 195–216. Kluwer Academic Press, 1989.