International Journal of Algebra and Computation

COMPRESSED DECISION PROBLEMS FOR GRAPH PRODUCTS AND APPLICATIONS TO (OUTER) AUTOMORPHISM GROUPS

NIKO HAUBOLD
Institut für Informatik, Universität Leipzig, Germany
haubold@informatik.uni-leipzig.de

MARKUS LOHREY
Institut für Informatik, Universität Leipzig, Germany
lohrey@informatik.uni-leipzig.de

CHRISTIAN MATHISSEN
Institut für Informatik, Universität Leipzig, Germany
mathissen@informatik.uni-leipzig.de

Received (Day Month Year)
Revised (Day Month Year)

Communicated by [editor]

It is shown that the compressed word problem of a graph product of finitely generated groups is polynomial time Turing-reducible to the compressed word problems of the vertex groups. A direct corollary of this result is that the word problem for the automorphism group of a right-angled Artin group or a right-angled Coxeter group can be solved in polynomial time. Moreover, it is shown that a restricted variant of the simultaneous compressed conjugacy problem is polynomial time Turing-reducible to the same problem for the vertex groups. A direct corollary of this result is that the word problem for the outer automorphism group of a right-angled Artin group or a right-angled Coxeter group can be solved in polynomial time. Finally, it is shown that the compressed variant of the ordinary conjugacy problem can be solved in polynomial time for right-angled Artin groups.

Keywords: graph products; decision problems for groups; algorithms for compressed strings.

1. Introduction

Since it was introduced by Dehn in 1910, the *word problem* for groups has emerged to a fundamental computational problem linking group theory, topology, mathematical logic, and computer science, see [35] for references. The word problem for a finitely generated group $\mathbb{G}$ asks, whether a given word over the generators of $\mathbb{G}$ represents the identity of $\mathbb{G}$. Dehn proved that the word problem is decidable for surface groups. Another mile stone is the work of Magnus, showing that the word

problem is decidable for one-relator groups. On the other hand, almost 50 years after the appearance of Dehn's work, Novikov [39] and independently Boone [4] proved the existence of a finitely presented group with undecidable word problem. Despite this negative result, many natural classes of groups with decidable word problem are known. We already mentioned one-relator groups. Other examples are finitely generated linear groups and automatic groups. With the rise of computational complexity theory, also the complexity of the word problem attracted attention. This topic has gained further relevance by potential applications of combinatorial group theory for secure cryptographic systems [38]. For finitely generated linear groups and automatic groups, for instance, very efficient algorithms for the word problem are known (logarithmic space for finitely generated linear groups [31] and quadratic time for automatic groups [16]).

In this paper, we are mainly concerned with the word problem for (outer) automorphism groups of certain groups. In order to solve the word problem in these groups efficiently, a "compressed" variant of the word problem was introduced in [33,34,44]. In the *compressed word problem* for a group $\mathbb{G}$, the input word over the generators is not given explicitly but succinctly via a so called *straight-line program* (SLP for short). This is a context free grammar $\mathbb{A}$ that generates exactly one word $\mathsf{val}(\mathbb{A})$, see Section 2.2. 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. [2,17,29,33,37,41,42]. In [34,44] it was shown that the word problem for the automorphism group $\mathsf{Aut}(\mathbb{G})$ of a group $\mathbb{G}$ can be reduced in polynomial time to the *compressed* word problem for $\mathbb{G}$. In [44], it was shown that the compressed word problem for a finitely generated free group $\mathbb{F}$ can be solved in polynomial time. Hence, the word problem for $\mathsf{Aut}(\mathbb{F})$ turned out to be solvable in polynomial time [44], which solved an open problem from [25]. This result was generalized to graph groups (right-angled Artin groups) [34] and fully residually free groups [36]. In [22], it was shown that the compressed word problem for an HNN-extension $\langle H, t \mid t^{-1}at = \varphi(a)(a \in A)\rangle$ with A a finite subgroup of the base group H can be reduced in polynomial time to the compressed word problem for H, and a corresponding result for amalgamated free products was shown as well.

The first main result of this paper states that a similar transfer result also holds for the graph product construction. The graph product construction is a well-known construction in mathematics, see e.g. [20,24], that generalizes both free products and direct products: An independence relation on the vertex groups of the graph product specifies, which groups are allowed to commute elementwise. Theorem 26 states that the compressed word problem for a graph product of groups is polynomial time Turing-reducible to the compressed word problems for the vertex groups. In particular, if for each vertex group the compressed word problem can be solved in polynomial time, then the same holds for a graph product of these groups. As a corollary, it follows that the word problem for the automorphism group of a graph

group or a right-angled Coxeter group can be solved in polynomial time.

It is not straightforward to carry over these complexity results from automorphism groups to *outer automorphism groups*. Nevertheless, Schleimer was able to prove in [44] 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. We call this problem the *restricted simultaneous compressed conjugacy problem* for the finitely generated group $\mathbb{G}$. This problem is parametrized by a finite subset $B \subseteq \mathbb{G}$. The input for the restricted simultaneous compressed conjugacy problem for $\mathbb{G}$ w.r.t. B consists of an SLP $\mathbb{A}_a$ (over the generators of $\mathbb{G}$) for each $a \in B$ and it is asked whether there exists a single $x \in \mathbb{G}$ such that $a = x\,\mathsf{val}(\mathbb{A}_a)\,x^{-1}$ in $\mathbb{G}$ for all $a \in B$. It is not difficult to show that the word problem for the outer automorphism group of $\mathbb{G}$ can be reduced in polynomial time to the restricted simultaneous compressed conjugacy problem for $\mathbb{G}$ w.r.t. an arbitrary finite generating set of $\mathbb{G}$. Our second main result states the following: If $\mathbb{G}$ is a graph product of groups $\mathbb{G}_1, \ldots, \mathbb{G}_n$ and for all $1 \leq i \leq n$, Σ_i is a finite generating set of $\mathbb{G}_i$, then the restricted simultaneous compressed conjugacy problem for the graph product $\mathbb{G}$ w.r.t. the finite generating $\bigcup_{i=1}^{n} \Sigma_i$ is polynomial time Turing-reducible to the restricted simultaneous compressed conjugacy problems for the vertex groups $\mathbb{G}_i$ w.r.t. Σ_i (Theorem 27). As a corollary, it follows that the word problem for the outer automorphism group of a graph group or a right-angled Coxeter group can be solved in polynomial time.

In the final part of the paper, we present a polynomial time algorithm for the compressed version of the classical conjugacy problem in a graph group $\mathbb{G}$: In this problem, we have given two SLPs $\mathbb{A}$ and $\mathbb{B}$ and we ask whether there exists $x \in \mathbb{G}$ such that $\mathsf{val}(\mathbb{A}) = x\,\mathsf{val}(\mathbb{B})x^{-1}$ in $\mathbb{G}$. For our polynomial time algorithm, we have to develop a pattern matching algorithm for SLP-compressed Mazurkiewicz traces, which is inspired by a pattern matching algorithm for hierarchical message sequence charts from [18]. For the non-compressed version of the conjugacy problem in a graph group, a linear time algorithm was presented in [47] based on [32]. In [10] this result was generalized to various subgroups of graph groups.

Some of the results in this paper were announced without proof in the extended abstracts [23,34].

2. Preliminaries

Let Γ be a finite alphabet. With ε we denote the empty word. For a word $s = a_1 \cdots a_m$ $(a_1, \ldots, a_m \in \Gamma)$ let

- $|s| = m$, $\mathsf{alph}(s) = \{a_1, \ldots, a_m\}$,
- $s[i] = a_i$ for $1 \leq i \leq m$,
- $s[i:j] = a_i \cdots a_j$ for $1 \leq i \leq j \leq m$ and $s[i:j] = \varepsilon$ for $i > j$,
- $s[:i] = s[1:i] = a_1 \cdots a_i$ for $0 \leq i \leq m$,
- $s[i:] = s[i:m] = a_i \cdots a_m$ for $1 \leq i \leq m+1$, and
- $|s|_a = |\{k \mid s[k] = a\}|$.

We use $\Gamma^{-1} = \{a^{-1} \mid a \in \Gamma\}$ to denote a disjoint copy of Γ and let $\Gamma^{\pm 1} = \Gamma \cup \Gamma^{-1}$. Define $(a^{-1})^{-1} = a$; this defines an involution $^{-1} : \Gamma^{\pm 1} \to \Gamma^{\pm 1}$, which can be extended to an involution on $(\Gamma^{\pm 1})^*$ by setting $(a_1 \cdots a_n)^{-1} = a_n^{-1} \cdots a_1^{-1}$. For $\Delta \subseteq \Gamma$ we define the projection morphism $\pi_\Delta : \Gamma^* \to \Delta^*$ by $\pi_\Delta(a) = a$ for $a \in \Delta$ and $\pi_\Delta(a) = \varepsilon$ for $a \in \Gamma \setminus \Delta$. For a map $f : A \to B$ and $A' \subseteq A$ we denote with $f{\restriction_{A'}}$ the map $f{\restriction_{A'}} : A' \to B$ such that $f{\restriction_{A'}}(x) = f(x)$ for all $x \in A'$.

2.1. *Computational complexity*

We assume that the reader is familiar with the basic concepts of complexity theory, see e.g. [40] for more details. With P we denote the complexity class deterministic polynomial time. Let A and B be two computational problems. We write $A \leq_m^{\log} B$ if A is (many-one) *logspace reducible* to B. This means that there exists a Turing machine with a logarithmic working tape that computes a mapping f such that for all inputs x we have: $x \in A$ if and only if $f(x) \in B$. We write $A \leq_T^P B$, if A is *polynomial time Turing-reducible* to B. This means that A can be decided by a deterministic polynomial time Turing machine that uses B as an oracle.

The relations $\leq_m^{\log}$ and $\leq_T^P$ are transitive, and $A \leq_m^{\log} B$ implies $A \leq_T^P B$. Moreover $A \leq_T^P B \in \mathsf{P}$ implies $A \in \mathsf{P}$.

If $A, B_1, \ldots, B_n$ are computational problems and $\leq \in \{\leq_m^{\log}, \leq_T^P\}$, then we write $A \leq \{B_1, \ldots, B_n\}$ if $A \leq \bigcup_{i=1}^n (\{i\} \times B_i)$ (the set $\bigcup_{i=1}^n (\{i\} \times B_i)$ is the disjoint union of the B_i with every element from B_i marked by i).

2.2. *CCP-expressions and straight-line programs*

In this section we introduce straight-line programs, which are used as a compressed representation of strings with reoccurring subpatterns [43]. For our applications, it will be useful to consider a generalization of straight-line programs: so called CCP-systems.

Let V and Γ be disjoint finite alphabets. The set $\mathsf{CCP}(V, \Gamma)$ of *CCP-expressions* over V and Γ (CCP stands for concatenation-cut-projection, which are the three basic string operations involved in CCP-expressions) is inductively defined as follows:

(a) $V \cup \Gamma \cup \{\varepsilon\} \subseteq \mathsf{CCP}(V, \Gamma)$
(b) If $\alpha, \beta \in \mathsf{CCP}(V, \Gamma)$, then $\alpha\beta \in \mathsf{CCP}(V, \Gamma)$ (concatenation).
(c) If $\alpha \in \mathsf{CCP}(V, \Gamma)$ and $i, j \in \mathbb{N}$, then $(\alpha)[i : j] \in \mathsf{CCP}(V, \Gamma)$ (cut).
(d) If $\alpha \in \mathsf{CCP}(V, \Gamma)$ and $\Delta \subseteq \Gamma$, then $\pi_\Delta(\alpha) \in \mathsf{CCP}(V, \Gamma)$ (projection).

The set $\mathsf{CC}(V, \Gamma)$ of *CC-expressions* over V and Γ is defined in the same way, but we omit rule (d) for projection. Note that $(V \cup \Gamma)^* \subseteq \mathsf{CC}(V, \Gamma)$ by (a) and (b). For $\alpha \in \mathsf{CCP}(V, \Gamma)$ we define the size $|\alpha| \in \mathbb{N}$ inductively as follows:

- $|\alpha| = 1$ for $\alpha \in V \cup \Gamma \cup \{\varepsilon\}$
- $|\alpha\beta| = |\alpha| + |\beta|$

- $|(\alpha)[i:j]| = |\alpha| + \lceil \log_2(i) \rceil + \lceil \log_2(j) \rceil$ for $i, j \in \mathbb{N}$
- $|\pi_\Delta(\alpha)| = |\alpha| + 1$ for $\Delta \subseteq \Gamma$

A *CCP-system* is a tuple $\mathbb{A} = (V, \Gamma, \mathsf{rhs}_\mathbb{A}, S)$ such that:

- V (the set of variables) and Γ (the terminal alphabet) are disjoint finite alphabets.
- $\mathsf{rhs}_\mathbb{A}$ (for right-hand side) is a mapping from V to $\mathsf{CCP}(V, \Gamma)$ for which there exists a linear order $\prec$ on V such that for all $X, Y \in V$: If Y occurs in $\mathsf{rhs}_\mathbb{A}(X)$, then $Y \prec X$.
- $S \in V$ (the initial variable).

We define the evaluation mapping $\mathsf{val}_\mathbb{A} : \mathsf{CCP}(V, \Gamma) \to \Gamma^*$ for a CCP-system $\mathbb{A}$ inductively as follows, where $\alpha, \beta \in \mathsf{CCP}(V, \Gamma)$:

- $\mathsf{val}_\mathbb{A}(\varepsilon) = \varepsilon$
- $\mathsf{val}_\mathbb{A}(a) = a$ for $a \in \Gamma$
- $\mathsf{val}_\mathbb{A}(X) = \mathsf{val}_\mathbb{A}(\mathsf{rhs}_\mathbb{A}(X))$ for $X \in V$
- $\mathsf{val}_\mathbb{A}(\alpha\beta) = \mathsf{val}_\mathbb{A}(\alpha)\mathsf{val}_\mathbb{A}(\beta)$ (concatenation of words)
- $\mathsf{val}_\mathbb{A}((\alpha)[i:j]) = (\mathsf{val}_\mathbb{A}(\alpha))[i:j]$ for $i, j \in \mathbb{N}$
- $\mathsf{val}_\mathbb{A}(\pi_\Delta(\alpha)) = \pi_\Delta(\mathsf{val}_\mathbb{A}(\alpha))$ for $\Delta \subseteq \Gamma$

The property for $\mathsf{rhs}_\mathbb{A}$ ensures that the mapping $\mathsf{val}_\mathbb{A}$ is uniquely defined in this way. Finally, let us set $\mathsf{val}(\mathbb{A}) = \mathsf{val}_\mathbb{A}(S)$. Occasionally, we will consider CCP-systems without an initial variable. For such a system $\mathbb{A}$, $\mathsf{val}(\mathbb{A})$ is not defined. If $\mathbb{A}$ is clear from the context, then we will omit the index $\mathbb{A}$ in the notation $\mathsf{val}_\mathbb{A}$ and $\mathsf{rhs}_\mathbb{A}$. We define the size of $\mathbb{A}$ as $|\mathbb{A}| = \sum_{X \in V} |\mathsf{rhs}(X)|$.

Example 1. We consider the CCP-system $\mathbb{A} = (\{A, B, C, D, E\}, \{a, b, c\}, \mathsf{rhs}, E)$ with rhs defined as follows:

$$\begin{array}{ll} \mathsf{rhs}(A) = ab & \mathsf{rhs}(B) = ac \\ \mathsf{rhs}(C) = BA & \mathsf{rhs}(D) = \pi_{\{a,c\}}(C)\pi_{\{b,c\}}(C) \\ \mathsf{rhs}(E) = D[2:4] & \end{array}$$

Then we have:

$$\begin{array}{lll} \mathsf{val}(A) = ab & \mathsf{val}(B) = ac & \mathsf{val}(C) = acab \\ \mathsf{val}(D) = acacb & \mathsf{val}(E) = \mathsf{val}(\mathbb{A}) = cac & \end{array}$$

The size of the CCP-system is the sum of the sizes of all right-hand sides:

$$\begin{array}{lll} |\mathsf{rhs}(A)| = 2 & |\mathsf{rhs}(B)| = 2 & |\mathsf{rhs}(C)| = 2 \\ |\mathsf{rhs}(D)| = 4 & |\mathsf{rhs}(E)| = 1 + 1 + 2 = 4 & \end{array}$$

and therefore $|\mathbb{A}| = 14$.

If the mapping rhs in $\mathbb{A}$ is assumed to be a mapping from V to $\mathsf{CC}(V, \Gamma)$, then $\mathbb{A}$ is called a *CC-system*. Finally, a CCP-system where rhs is a mapping from V

to $(V \cup \Gamma)^*$ is called a *straight-line program* (SLP) [43]. Note that an SLP can be viewed as a context-free grammar, which generates exactly one string.

A CCP-system $\mathbb{A} = (V, \Gamma, \mathsf{rhs}, S)$ is in *normal form* if for all $X \in V$, $\mathsf{rhs}(X)$ is of one of the following forms, where $Y, Z \in V$, $i, j \in \mathbb{N}$ and $\Delta \subseteq \Gamma$: ε, $a \in \Gamma$, YZ, $Y[i : j]$, or $\pi_\Delta(Y)$. It is straightforward to transform an arbitrary CCP-system $\mathbb{A}$ into a CCP-system $\mathbb{B}$ in normal form such that $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$. Note that for SLPs, our normal form corresponds to Chomsky's normal form.

For a CCP-system $\mathbb{A}$ in normal form over a terminal alphabet $\Gamma^{\pm 1}$ we define the CCP-system $\mathbb{A}^{-1}$ inductively as follows: Let X be a nonterminal from $\mathbb{A}$. If $\mathsf{rhs}_\mathbb{A}(X) = a \in \Gamma^{\pm 1}$ then $\mathsf{rhs}_{\mathbb{A}^{-1}}(X) = a^{-1}$. If $\mathsf{rhs}_\mathbb{A}(X) = YZ$ in $\mathbb{A}$ then $\mathsf{rhs}_{\mathbb{A}^{-1}}(X) = ZY$. If $\mathsf{rhs}_\mathbb{A}(X) = Y[i : j]$ in $\mathbb{A}$ then $\mathsf{rhs}_{\mathbb{A}^{-1}}(X) = [k - j + 1 : k - i + 1]$, where $k = |\mathsf{val}_\mathbb{A}(X)|$. If $\mathsf{rhs}_\mathbb{A}(X) = \pi_\Delta(Y)$ in $\mathbb{A}$ for some $\Delta \subseteq \Gamma^{\pm 1}$, then $\mathsf{rhs}_{\mathbb{A}^{-1}}(X) = \pi_{\Delta^{-1}}(Y)$ in $\mathbb{A}^{-1}$. It is easy to see that $\mathsf{val}(\mathbb{A}^{-1}) = \mathsf{val}(\mathbb{A})^{-1}$.

CC-systems are called composition systems in [17]. Composition systems were also heavily used in [34,44] in order to solve compressed word problems efficiently. The following result was shown by Hagenah [21]:

Theorem 2 ([21]). *There is a polynomial time algorithm, which transforms a given CC-system $\mathbb{A}$ into an SLP $\mathbb{B}$ such that $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$ (the input size is $|\mathbb{A}|$ as defined above).*

We need a generalization of Hagenah's result.

Lemma 3. *Let p be a constant. Then there exists a polynomial time algorithm for the following problem:*

INPUT: Finite alphabets $\Gamma_1, \ldots, \Gamma_p$ and a CCP-system $\mathbb{A}$ over $\Gamma = \bigcup_{i=1}^p \Gamma_i$ such that $\Delta \in \{\Gamma_1, \ldots, \Gamma_p\}$ for every subexpression of the form $\pi_\Delta(\alpha)$ that appears in a right-hand side of $\mathbb{A}$.
OUTPUT: An SLP $\mathbb{B}$ over Γ such that $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$.

Proof. Let $\mathbb{A} = (V_\mathbb{A}, \Gamma, \mathsf{rhs}_\mathbb{A}, S)$ be the given CCP-system. Since for a CC-system an equivalent SLP can be constructed in polynomial time by Theorem 2, it suffices to construct a CC-system $\mathbb{B} = (V_\mathbb{B}, \Gamma, \mathsf{rhs}_\mathbb{B}, S_\mathbb{B})$ equivalent to $\mathbb{A}$ in polynomial time. Let

$$\mathcal{C} = \Big\{ \bigcap_{i \in K} \Gamma_i \mid K \subseteq \{1, \ldots, p\} \Big\} \cup \{\Gamma\}.$$

Note that $\mathcal{C}$ has constant size. Let $V_\mathbb{B} = \{X_\Delta \mid X \in V_\mathbb{A}, \Delta \in \mathcal{C}\}$ be the set of variables of $\mathbb{B}$. The right-hand side mapping $\mathsf{rhs}_\mathbb{B}$ will be defined in such a way that $\mathsf{val}(X_\Delta) = \pi_\Delta(\mathsf{val}(X))$. We set $S_\mathbb{B} = S_\Gamma$ (recall that S is the initial variable of $\mathbb{A}$).

If $\mathsf{rhs}_\mathbb{A}(X) = a \in \Gamma$, then we set $\mathsf{rhs}_\mathbb{B}(X_\Delta) = \pi_\Delta(a) \in \{\varepsilon, a\}$. If $\mathsf{rhs}_\mathbb{A}(X) = YZ$, then we set $\mathsf{rhs}_\mathbb{B}(X_\Delta) = Y_\Delta Z_\Delta$. If $\mathsf{rhs}_\mathbb{A}(X) = \pi_\Theta(Y)$ with $\Theta \in \{\Gamma_1, \ldots, \Gamma_p\}$, then we set $\mathsf{rhs}_\mathbb{B}(X_\Delta) = Y_{\Delta \cap \Theta}$. Note that $\Delta \cap \Theta \in \mathcal{C}$.

Finally, consider the case $\mathsf{rhs}_\mathbb{A}(X) = Y[i : j]$. We set $\mathsf{rhs}_\mathbb{B}(X_\Delta) = Y_\Delta[k : \ell]$, where $k = |\pi_\Delta(\mathsf{val}(Y)[: i - 1])| + 1$ and $\ell = |\pi_\Delta(\mathsf{val}(Y)[: j])|$. These lengths can be

computed in polynomial time as follows: Implicitly, we have already computed a CC-system, which generates the string $\mathsf{val}(Y)$. Hence, by adding a single definition, we obtain a CC-system for the string $\mathsf{val}(Y)[:i-1]$. Using Hagenah's algorithm [21] we can transform this CC-system in polynomial time into an equivalent SLP. From this SLP, the length $|\pi_\Delta(\mathsf{val}(Y)[:i-1])|$ can be easily computed bottom-up (the SLP for the string $\mathsf{val}(Y)[:i-1]$ is then not used anymore). The length $|\pi_\Delta(\mathsf{val}(Y)[:j])|$ can be computed similarly. Since the size of $\mathcal{C}$ is constant, the above construction works in polynomial time. □

In the proof of Lemma 3, it is crucial that p is a fixed constant, i.e., not part of the input. Otherwise the construction would lead to an exponential blow-up. It is not clear whether Lemma 3 remains true, when the terminal alphabet Γ is part of the input.

The proofs for the following well known are straightforward:

Lemma 4. *The following tasks can be solved in polynomial time:*

(1) Given an SLP $\mathbb{A}$, compute $|\mathsf{val}(\mathbb{A})|$ and $\mathsf{alph}(\mathsf{val}(\mathbb{A}))$.
(2) Given an SLP $\mathbb{A}$ and a number $i \in \{1, \ldots, |\mathsf{val}(\mathbb{A})|\}$, compute $\mathsf{val}(\mathbb{A})[i]$.[a]
(3) Given an SLP $\mathbb{A}$ and two numbers $1 \leq i \leq j \leq |\mathsf{val}(\mathbb{A})|$, compute an SLP $\mathbb{B}$ with $\mathsf{val}(\mathbb{B}) = \mathsf{val}(\mathbb{A})[i, j]$.

In [41], 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 [29]. 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 $\mathsf{val}(\mathbb{B})$? The running time of his algorithm is $\mathcal{O}(|\mathbb{A}| \cdot |\mathbb{B}|^2)$.

By Theorem 2 all algorithmic tasks from Lemma 4 can be solved in polynomial time for CC-systems (instead of SLPs) as well, and under the hypothesis of Lemma 3 they can be even solved for CCP-systems in polynomial time.

The next lemma will be crucial for our applications of SLPs.

Lemma 5. *For a given sequence $\varphi_1, \ldots, \varphi_n$ of homomorphisms $\varphi_i : \Gamma^* \to \Gamma^*$ $(1 \leq i \leq n)$ and a symbol $a \in \Gamma$ we can compute in logarithmic space an SLP $\mathbb{A}$ such that $\mathsf{val}(\mathbb{A}) = \varphi_1 \cdots \varphi_n(a)$. Moreover, $|\mathbb{A}| = \mathcal{O}(\sum_{b\in\Gamma} \sum_{i=1}^{n} |\varphi_i(b)|)$.*

Proof. Let us take variables $A_{i,b}$, where $0 \leq i \leq n$ and $b \in \Gamma$, and define the right-hand sides as follows:

$$\mathsf{rhs}(A_{0,b}) = b$$
$$\mathsf{rhs}(A_{i,b}) = A_{i-1,a_1} \cdots A_{i-1,a_m}, \text{ where } \varphi_i(b) = a_1 \cdots a_m$$

By induction on i one can easily show that $\mathsf{val}(\mathbb{A}_{i,b}) = \varphi_1 \cdots \varphi_i(b)$. □

[a] By [30], this problem is P-complete.

For our algorithms, it is useful to consider CCP-systems, which are divided into two layers. A *2-level CCP-system* is a tuple $\mathbb{A} = (\mathsf{Up}, \mathsf{Lo}, \Gamma, \mathsf{rhs}, S)$ such that the following holds:

- Up, Lo, and Γ are pairwise disjoint finite alphabets, $S \in \mathsf{Up}$, and $\mathsf{rhs} : \mathsf{Up} \cup \mathsf{Lo} \to \mathsf{CCP}(\mathsf{Up} \cup \mathsf{Lo}, \Gamma)$.
- The tuple $(\mathsf{Up}, \mathsf{Lo}, \mathsf{rhs}\restriction_{\mathsf{Up}}, S)$ is a CCP-system with terminal alphabet Lo.
- The tuple $(\mathsf{Lo}, \Gamma, \mathsf{rhs}\restriction_{\mathsf{Lo}})$ is an SLP (without initial variable) over the terminal alphabet Γ.

The set Up (resp. Lo) is called the set of *upper level variables* (*lower level variables*) of $\mathbb{A}$. Moreover, we set $V = \mathsf{Up} \cup \mathsf{Lo}$ and call it the set of variables of $\mathbb{A}$. The CCP-system $(\mathsf{Up}, \mathsf{Lo}, \mathsf{rhs}\restriction_{\mathsf{Up}}, S)$ is called the *upper part of* $\mathbb{A}$, briefly $\mathsf{up}(\mathbb{A})$, and the SLP (without initial variable) $(\mathsf{Lo}, \Gamma, \mathsf{rhs}\restriction_{\mathsf{Lo}})$ is the *lower part of* $\mathbb{A}$, briefly, $\mathsf{lo}(\mathbb{A})$. The upper level evaluation mapping $\mathsf{uval}_{\mathbb{A}} : \mathsf{CCP}(\mathsf{Up}, \mathsf{Lo}) \to \mathsf{Lo}^*$ of $\mathbb{A}$ is defined as $\mathsf{uval}_{\mathbb{A}} = \mathsf{val}_{\mathsf{up}(\mathbb{A})}$. The evaluation mapping $\mathsf{val}_{\mathbb{A}}$ is defined by $\mathsf{val}_{\mathbb{A}}(X) = \mathsf{val}_{\mathsf{lo}(\mathbb{A})}(\mathsf{val}_{\mathsf{up}(\mathbb{A})}(X))$ for $X \in \mathsf{Up}$ and $\mathsf{val}_{\mathbb{A}}(X) = \mathsf{val}_{\mathsf{lo}(\mathbb{A})}(X)$ for $X \in \mathsf{Lo}$. Finally, we set $\mathsf{val}(\mathbb{A}) = \mathsf{val}_{\mathbb{A}}(S)$. We define the size of $\mathbb{A}$ as $|\mathbb{A}| = \sum_{X \in V} |\mathsf{rhs}(X)|$.

Example 6. Let $\mathbb{A} = (\{F, G, H\}, \{A, B, C, D, E\}, \{a, b, c\}, \mathsf{rhs}, H)$ be a two-level CCP-system with rhs defined as follows:

$$\begin{aligned}
&\mathsf{rhs}(A) = a && \mathsf{rhs}(B) = b && \mathsf{rhs}(C) = c\\
&\mathsf{rhs}(D) = AB && \mathsf{rhs}(E) = AC &&\\
&\mathsf{rhs}(F) = EABCDEA &&&&\\
&\mathsf{rhs}(G) = F[2:6] &&&&\\
&\mathsf{rhs}(H) = \pi_{\{A,C,D\}}(G) &&&&
\end{aligned}$$

Then $\mathsf{up}(\mathbb{A}) = (\{F, G, H\}, \{A, B, C, D, E\}, \mathsf{rhs}\restriction_{\mathsf{Up}}, H)$ with $\mathsf{Up} = \{F, G, H\}$ and $\mathsf{lo}(\mathbb{A}) = (\{A, B, C, D, E\}, \{a, b, c\}, \mathsf{rhs}\restriction_{\mathsf{Lo}})$ with $\mathsf{Lo} = \{A, B, C, D, E\}$. The $\mathsf{uval}_{\mathbb{A}}$-values for the upper level variables are:

$$\begin{aligned}
\mathsf{uval}_{\mathbb{A}}(F) &= EABCDEA\\
\mathsf{uval}_{\mathbb{A}}(G) &= ABCDE\\
\mathsf{uval}_{\mathbb{A}}(H) &= ACD
\end{aligned}$$

The $\mathsf{val}_{\mathbb{A}}$-values for all variables of $\mathbb{A}$ are:

$$\begin{aligned}
\mathsf{val}_{\mathbb{A}}(A) = a \qquad \mathsf{val}_{\mathbb{A}}(B) &= b \qquad \mathsf{val}_{\mathbb{A}}(C) = c\\
\mathsf{val}_{\mathbb{A}}(D) = ab \qquad \mathsf{val}_{\mathbb{A}}(E) &= ac\\
\mathsf{val}_{\mathbb{A}}(F) = \mathsf{val}_{\mathbb{A}}(EABCDEA) &= acabcabaca\\
\mathsf{val}_{\mathbb{A}}(G) = \mathsf{val}_{\mathbb{A}}(ABCDE) &= abcabac\\
\mathsf{val}(\mathbb{A}) = \mathsf{val}_{\mathbb{A}}(H) = \mathsf{val}_{\mathbb{A}}(ACD) &= acab
\end{aligned}$$

2.3. *Decision problems for groups*

For background in combinatorial group theory see [35]. Let $\mathbb{G}$ be a finitely generated group and let Σ be a finite generating set for $\mathbb{G}$. Recall that the *word problem* for $\mathbb{G}$ with respect to Σ is the following decision problem:

INPUT: A word $w \in (\Sigma^{\pm 1})^*$.
QUESTION: Does $w = 1$ hold in $\mathbb{G}$?

The word problem for $\mathbb{G}$ is certainly the most important algorithmic problem for the group $\mathbb{G}$. It is a simple observation that if Σ_1 and Σ_2 are finite generating sets for $\mathbb{G}$, then the word problem for $\mathbb{G}$ with respect to Σ_1 is logspace reducible to the word problem for $\mathbb{G}$ with respect to Σ_2. Hence, modulo logspace reducibility, the computational complexity of the word problem does not depend on the chosen generating set and is a property of the group $\mathbb{G}$. Hence, we can speak of the word problem $\mathsf{WP}(\mathbb{G})$ for the group $\mathbb{G}$. By the seminal result of Novikov [39] and Boone [4], there exist finitely presented groups with an undecidable word problem.

2.3.1. *(Outer) automorphism groups.*

In this paper, we are mainly interested in the complexity of the word problem for certain (outer) automorphism groups. Recall that the *automorphism group* $\mathsf{Aut}(\mathbb{G})$ of a group $\mathbb{G}$ is the set of all bijective homomorphisms from $\mathbb{G}$ to itself with composition as operation and the identity mapping as the identity element. An automorphism φ is called *inner* if there is a group element $x \in \mathbb{G}$ such that $\varphi(y) = xyx^{-1}$ for all $y \in \mathbb{G}$. The set of all inner automorphisms for a group $\mathbb{G}$ forms the *inner automorphism group* of $\mathbb{G}$ denoted by $\mathsf{Inn}(\mathbb{G})$. This is easily seen to be a normal subgroup of $\mathsf{Aut}(\mathbb{G})$ and the quotient group $\mathsf{Out}(\mathbb{G}) = \mathsf{Aut}(\mathbb{G})/\mathsf{Inn}(\mathbb{G})$ is called the *outer automorphism group* of $\mathbb{G}$. In general the (outer) automorphism group of a finitely generated group is not finitely generated.

Let Ψ be a finite subset of $\mathsf{Aut}(\mathbb{G})$ and consider the finitely generated subgroup $\langle \Psi \rangle \leq \mathsf{Aut}(\mathbb{G})$. Let $H \leq \mathsf{Out}(\mathbb{G})$ be the image of $\langle \Psi \rangle$ under the canonical morphism from $\mathsf{Aut}(\mathbb{G})$ to $\mathsf{Out}(\mathbb{G})$. Since an automorphism of $\mathbb{G}$ belongs to the same coset (with respect to $\mathsf{Inn}(\mathbb{G})$) as the identity if and only if it is inner, we can rephrase the word problem for $H \leq \mathsf{Out}(\mathbb{G})$ as follows:

INPUT: A word $w \in (\Psi^{\pm 1})^*$.
QUESTION: Does w represent an element of $\mathsf{Inn}(\mathbb{G})$ in $\mathsf{Aut}(\mathbb{G})$?

2.3.2. *Compressed word problems and compressed conjugacy problems.*

Our main tool for studying the complexity of the word problem for automorphism groups is a variant of the word problem, where the input word is given succinctly by an SLP. The *compressed word problem* for $\mathbb{G}$ with respect to Σ is the following decision problem:

INPUT: An SLP $\mathbb{A}$ over the terminal alphabet $\Sigma^{\pm 1}$.

QUESTION: Does $\mathsf{val}(\mathbb{A}) = 1$ hold in $\mathbb{G}$?

Here, the input size is $|\mathbb{A}|$. As for the ordinary word problem, it is easy to see that for the compressed word problem the complexity does not depend on the chosen generating set (modulo logspace reducibility), which allows to speak of *the* compressed word problem for the group $\mathbb{G}$. The compressed word problem for $\mathbb{G}$ is also denoted by $\mathsf{CWP}(\mathbb{G})$.

It is clear that for a finite group $\mathbb{G}$, $\mathsf{CWP}(\mathbb{G})$ can be solved in polynomial time. In [33], it was shown that the compressed word problem for a finitely generated free group can be solved in polynomial time. This result was generalized to graph groups (see Section 2.5 for the definition) in [34]. Another class of groups with polynomial time compressed word problem are finitely generated nilpotent groups:

Theorem 7. *Let $\mathbb{G}$ be a finitely generated nilpotent group. Then $\mathsf{CWP}(\mathbb{G})$ can be solved in polynomial time.*

Proof. Let $\mathbb{G}$ be a finitely generated nilpotent group. Then $\mathbb{G}$ has a finitely generated torsion-free nilpotent subgroup $\mathbb{H}$ such that the index $[\mathbb{G} : \mathbb{H}]$ is finite [26, Theorem 17.2.2]. By Theorem 8(2) below, it suffices to solve $\mathsf{CWP}(\mathbb{H})$ in polynomial time. There exists $d \geq 1$ such that the finitely generated torsion-free nilpotent group $\mathbb{H}$ can be embedded into the group $\mathsf{UT}_d(\mathbb{Z})$ of upper triangular $(d \times d)$-matrices over $\mathbb{Z}$ [26, Theorem 17.2.5]. Let $\varphi : \mathbb{H} \to \mathsf{UT}_d(\mathbb{Z})$ be this embedding. As remarked in [19], if w is a word of length n over the generators of $\mathbb{H}$, then the absolute value of every entry in the integer matrix $\varphi(w)$ is bounded by $\mathcal{O}(n^{d-1})$. If w is given by an SLP in Chomsky normal form of size m, we can evaluate the SLP bottom-up in the group $\mathsf{UT}_d(\mathbb{Z})$ as follows: For every variable X, we compute the matrix $\varphi(\mathsf{val}(X))$. If $\mathsf{rhs}(X) = YZ$ and the matrices $\varphi(\mathsf{val}(Y))$, $\varphi(\mathsf{val}(Z))$ are already computed, then $\varphi(\mathsf{val}(X))$ is set to the product of these two matrices. Since $|\mathsf{val}(X)| \leq |w| \leq 2^m$, every entry in $\varphi(\mathsf{val}(X))$ can be represented with $\mathcal{O}((d-1)m)$ bits. Hence, the evaluation can be accomplished in polynomial time. □

The following theorem collects some preservation results for the complexity of the compressed word problem (statement (1) and (3) are trivial):

Theorem 8. *The following hold for all finitely generated groups $\mathbb{G}$ and $\mathbb{H}$:*

(1) If $\mathbb{H} \leq \mathbb{G}$, then $\mathsf{CWP}(\mathbb{H}) \leq_m^{\log} \mathsf{CWP}(\mathbb{G})$.
(2) If $\mathbb{G} \leq \mathbb{H}$ and the index $[\mathbb{H} : \mathbb{G}]$ is finite, then $\mathsf{CWP}(\mathbb{H}) \leq_T^P \mathsf{CWP}(\mathbb{G})$ [34].
(3) $\mathsf{CWP}(\mathbb{G} \times \mathbb{H}) \leq_m^{\log} \{\mathsf{CWP}(\mathbb{G}), \mathsf{CWP}(\mathbb{H})\}$
(4) If A is a finite subgroup of both $\mathbb{G}$ and $\mathbb{H}$, then $\mathsf{CWP}(\mathbb{G} *_A \mathbb{H}) \leq_T^P \{\mathsf{CWP}(\mathbb{G}), \mathsf{CWP}(\mathbb{H})\}$ [22] (here, $\mathbb{G} *_A \mathbb{H}$ is the amalgamated free product of $\mathbb{G}$ and $\mathbb{H}$ along A).
(5) If $\mathbb{H}$ is an HNN-extension of $\mathbb{G}$ with finite associated subgroups, then $\mathsf{CWP}(\mathbb{H}) \leq_T^P \mathsf{CWP}(\mathbb{G})$ [22].

In this paper, we will prove a similar preservation theorem for the compressed word problem for graph products of groups, see Section 2.5. Graph products generalize free and direct products.

We are mainly interested in the compressed word problem for a group because of the following application for the word problem for automorphism groups, which was first shown by Schleimer in [44].

Proposition 9 (cf [44]). *Let $\mathbb{G}$ be a finitely generated group and let $\mathbb{H}$ be a finitely generated subgroup of $\mathsf{Aut}(\mathbb{G})$. Then $\mathsf{WP}(\mathbb{H}) \leq_m^{\log} \mathsf{CWP}(\mathbb{G})$.*

The proof of Proposition 9 uses Lemma 5.

In order to solve the word problem for a finitely generated subgroup of $\mathsf{Out}(\mathbb{G})$, we have to deal with compressed conjugacy problems in $\mathbb{G}$. Recall that two elements g and h of a group $\mathbb{G}$ are *conjugated* if and only if there exists $x \in \mathbb{G}$ such that $g = xhx^{-1}$. The classical *conjugacy problem* for $\mathbb{G}$ asks, whether two given elements of $\mathbb{G}$ are conjugated. We will consider a compressed variant of this problem in $\mathbb{G}$, which we call the *compressed conjugacy problem for* $\mathbb{G}$, $\mathsf{CCP}(\mathbb{G})$ [b] for short:

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

In order to solve the word problem for finitely generated subgroups of $\mathsf{Out}(\mathbb{G})$ we need an extension of $\mathsf{CCP}(\mathbb{G})$ to several pairs of input SLPs. Let us call this problem the *simultaneous compressed conjugacy problem* for $\mathbb{G}$:

INPUT: SLPs $\mathbb{A}_1, \mathbb{B}_1, \ldots, \mathbb{A}_n, \mathbb{B}_n$ over the terminal alphabet $\Sigma^{\pm 1}$.
QUESTION: Does there exist $x \in (\Sigma^{\pm 1})^*$ such that $\mathsf{val}(\mathbb{A}_i) = x\,\mathsf{val}(\mathbb{B}_i)x^{-1}$ in $\mathbb{G}$ for all $i \in \{1, \ldots, n\}$?

The simultaneous (non-compressed) conjugacy problem also appears in connection with group-based cryptography [38]. Unfortunately, we do not know, whether the simultaneous compressed conjugacy problem for graph groups (and hence for graph products of finitely generated groups) can be solved in polynomial time. But, in order to deal with the word problem for finitely generated subgroups of $\mathsf{Out}(\mathbb{G})$, a restriction of this problem suffices, where the SLPs $\mathbb{B}_1, \ldots, \mathbb{B}_n$ from the simultaneous compressed conjugacy problem produce generators of the group $\mathbb{G}$. Let $B \subseteq \mathbb{G}$ be a fixed finite non-empty set. The *restricted simultaneous compressed conjugacy problem* for $\mathbb{G}$ and B, briefly $\mathsf{RSCCP}(\mathbb{G}, B)$, is the following computational problem:

INPUT: SLPs $\mathbb{A}_a$ $(a \in B)$ over the terminal alphabet $\Sigma^{\pm 1}$.
QUESTION: Does there exist $x \in (\Sigma^{\pm 1})^*$ with $\mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}$ for all $a \in B$?

An x such that $\mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}$ for all $a \in B$ is called a *solution* of the $\mathsf{RSCCP}(\mathbb{G}, B)$ instance. Note that the restricted simultaneous compressed conjugacy

[b] Note that we use the abbreviation CCP for the compressed conjugacy problem as well as for CCP-expressions. We hope that the actual meaning will always be clear from the context.

problem depends on the chosen set B. In our application for $\mathsf{Out}(\mathbb{G})$, B will be an arbitrary finite generating set for $\mathbb{G}$. The authors cannot disprove that that there exist two different generating sets B_1 and B_2 for $\mathbb{G}$ such that $\mathsf{RSCCP}(\mathbb{G}, B_1)$ is decidable while $\mathsf{RSCCP}(\mathbb{G}, B_2)$ is undecidable.

Proposition 10. *Let $\mathbb{G}$ be a finitely generated group and let $\mathbb{H}$ be a finitely generated subgroup of $\mathsf{Out}(\mathbb{G})$. Then for every finite generating set Σ of $\mathbb{G}$ we have $\mathsf{WP}(\mathbb{H}) \leq_m^{\log} \mathsf{RSCCP}(\mathbb{G}, \Sigma)$.*

Proof. Let Σ be a finite generating set for $\mathbb{G}$ and let Ψ be a finite subset of $\mathsf{Aut}(\mathbb{G})$ such that the corresponding cosets (with respect to $\mathsf{Inn}(\mathbb{G})$) generate $\mathbb{H}$ as a monoid. Let $\psi = \psi_1 \cdots \psi_n$ with $\psi_1, \ldots, \psi_n \in \Psi$ be the input for the word problem for $\mathbb{H}$. We have to check, whether the automorphism ψ is inner. Using Lemma 5 we can compute in polynomial time SLPs $\mathbb{A}_a$ ($a \in \Sigma$) over $\Sigma^{\pm 1}$ with $\mathsf{val}(\mathbb{A}_a) = \psi(a)$ in $\mathbb{G}$ for all $a \in \Sigma$. The automorphism ψ is inner if and only if there exists $x \in \mathbb{G}$ such that $\mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}$ for all $a \in \Sigma$. This is an instance of $\mathsf{RSCCP}(\mathbb{G}, \Sigma)$. □

2.4. *Traces*

We are interested in graph products of groups. Our definition of graph products will be based on traces (partially commutative words). In the following we introduce some notions from trace theory, see [12,14] for more details. An *independence alphabet* is an undirected graph (Σ, I) (without loops). Thus, I is a symmetric and irreflexive relation on Σ. The set Σ may be infinite, but most of the time, it will be finite in this paper. 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*. We denote by $[w]_I$ the trace represented by the word $w \in \Sigma^*$. Let $\mathsf{alph}([w]_I) = \mathsf{alph}(w)$ and $|[w]_I| = |w|$. The *dependence alphabet* associated with (Σ, I) is (Σ, D), where $D = (\Sigma \times \Sigma) \setminus I$. Note that the relation D is reflexive. For $a \in \Sigma$ let $I(a) = \{b \in \Sigma \mid (a, b) \in I\}$ be the letters that commute with a and $D(a) = \{b \in \Sigma \mid (a, b) \in D\}$ be the letters that are dependent from a. For traces $u, v \in \mathbb{M}(\Sigma, I)$ we denote with uIv the fact that $\mathsf{alph}(u) \times \mathsf{alph}(v) \subseteq I$.

An *independence clique* is a subset $\Delta \subseteq \Sigma$ such that $(a, b) \in I$ for all $a, b \in \Delta$ with $a \neq b$. For a *finite* independence clique Δ, we write $[\Delta]_I$ for the trace $[a_1 a_2 \cdots a_n]_I$, where $a_1, a_2, \ldots, a_n$ is an arbitrary enumeration of Δ.

The following lemma is one of the most fundamental facts for trace monoids, see e.g. [14]:

Lemma 11 (Levi's Lemma). *Let $u_1, u_2, v_1, v_2 \in \mathbb{M}(\Sigma, I)$ such that $u_1 u_2 = v_1 v_2$. Then there exist $x, y_1, y_2, z \in \mathbb{M}(\Sigma, I)$ such that $y_1 I y_2$ and $u_1 = xy_1$, $u_2 = y_2 z$, $v_1 = xy_2$, and $v_2 = y_1 z$.*

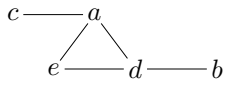
We use Levi's Lemma to prove the following statement:

Lemma 12. *Let $a \in \Sigma$. The decomposition of a trace $w \in \mathbb{M}(\Sigma, I)$ as $w = u_1u_2$ with u_2Ia and $|u_2|$ maximal is unique in $\mathbb{M}(\Sigma, I)$.*

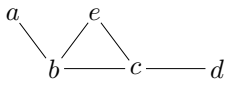
Proof. Let $u_1u_2 = w = v_1v_2$ be such that u_2Ia, v_2Ia and $|u_2|$ and $|v_2|$ are both maximal (hence $|u_2| = |v_2|$). By Levi's Lemma there are traces x, y_1, y_2, z such that y_1Iy_2 and $u_1 = xy_1$, $u_2 = y_2z$, $v_1 = xy_2$, and $v_2 = y_1z$. From u_2Ia and v_2Ia we get y_1Ia and y_2Ia. Maximality of $|u_2| = |v_2|$ and $xy_1u_2 = w = xy_2v_2$ implies $y_1 = y_2 = \varepsilon$. Hence $u_1 = v_1$ and $u_2 = v_2$. □

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 node-labelled graphs. Hence, we can speak of *the* dependence graph of a trace.

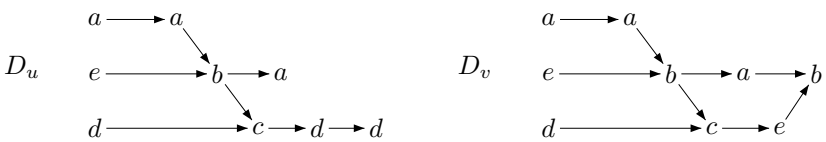
Example 13. We consider the following independence alphabet (Σ, I):



Then the corresponding dependence alphabet is:



We consider the words $u = aeadbacdd$ and $v = eaabdcaeb$. Then the dependence graphs D_u of u and D_v of v look as follows, where we label the vertices i with the letter $u[i]$ (resp. $v[i]$):



Note that we only show Hasse diagrams and hence omit for instance the edge from the first d to the last d in D_u.

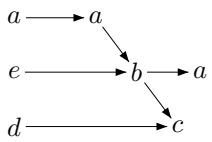
Let E_w be the edge relation for the dependence graph D_w for a trace w. A subset $V \subseteq \{1, \ldots, |w|\}$ is called *downward-closed*, if $(i, j) \in E_w$ and $j \in V$ implies $i \in V$. A subset $V \subseteq \{1, \ldots, |w|\}$ is called *convex*, if $(i, j), (j, k) \in E_w^*$ and $i, k \in V$ implies $j \in V$.

2.4.1. *The prefix and suffix order on traces.*

Let $u, v \in \mathbb{M}(\Sigma, I)$. Then u is a *prefix* (resp. *suffix*) of v if there exists some $w \in \mathbb{M}(\Sigma, I)$ such that $uw = v$ (resp. $wu = v$) in $\mathbb{M}(\Sigma, I)$, for short $u \preceq_p v$ (resp. $u \preceq_s v$). Prefixes of a trace u exactly correspond to downward-closed subsets of the dependence graph of u. The *prefix infimum* (resp. *suffix infimum*) $u \sqcap_p v$ (resp. $u \sqcap_s v$) is the largest trace w w.r.t. $\preceq_p$ (resp. $\preceq_s$) such that $w \preceq_p u$ and $w \preceq_p v$ (resp. $w \preceq_s u$ and $w \preceq_s v$); it always exists [8]. With $u \setminus_p v$ (resp. $u \setminus_s v$) we denote the unique trace w such that $u = (u \sqcap_p v)w$ (resp. $u = w(u \sqcap_s v)$); uniqueness follows from the fact that $\mathbb{M}(\Sigma, I)$ is cancellative. Note that $u \setminus_p v = u \setminus_p (u \sqcap_p v)$ and $u \setminus_s v = u \setminus_s (u \sqcap_s v)$. For $u \in \mathbb{M}(\Sigma, I)$, we denote with $\min(u) \subseteq \Sigma$ (resp. $\max(u) \subseteq \Sigma$) the set of all symbols $a \in \Sigma$ such that $a \preceq_p u$ (resp. $a \preceq_s u$). Clearly, $\min(u)$ and $\max(u)$ are finite independence cliques and $[\min(u)]_I \preceq_p u$ and $[\max(u)]_I \preceq_s u$. Occasionally, we will identify the traces $[\min(u)]_I$ and $[\max(u)]_I$ with the independence cliques $\min(u)$ and $\max(u)$, respectively.

Example 14. We continue Example 13 above. We have $u \sqcap_p v = [aeadbac]_I =: w$ and its dependence graph is:

D_w



Furthermore we have $\min(w) = \{a, d, e\}$ and $\max(w) = \{a, c\}$.

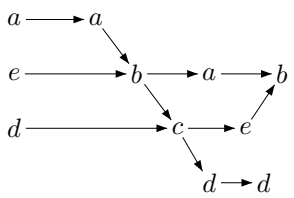
In contrast to the prefix infimum and the suffix infimum, the prefix supremum and the suffix supremum of two traces do not always exist. If it exists, the *prefix supremum* (resp. *suffix supremum*) of two traces $u, v \in \mathbb{M}(\Sigma, I)$ is the smallest trace w w.r.t. $\preceq_p$ (resp. $\preceq_s$) such that $u \preceq_p w$ and $v \preceq_p w$ (resp. $u \preceq_s w$ and $v \preceq_s w$); it is denoted by $u \sqcup_p v$ (resp. $u \sqcup_s v$).

We show in the following some simple facts for traces. The Lemmas 15, 17, and 18 also hold for suffixes and suffix suprema. The next result can be found in [8]:

Lemma 15 ([8]). *The trace $u \sqcup_p v$ exists if and only if $(u \setminus_p v)\, I\, (v \setminus_p u)$, in which case we have $u \sqcup_p v = (u \sqcap_p v)\,(u \setminus_p v)\,(v \setminus_p u)$ (which is $u(v \setminus_p u) = v(u \setminus_p v)$).*

Example 16. We continue Example 14 above. Since $u \setminus_p v = dd$ and $v \setminus_p u = eb$ we have $(u \setminus_p v) I (v \setminus_p u)$ and hence the supremum $s = u \sqcup_p v = [aeadbacddeb]_I$ is defined. The dependence graph for s is:

D_s



We can define the supremum of several traces $u_1, \ldots, u_r$ analogously by induction: $u_1 \sqcup_p \cdots \sqcup_p u_r = (u_1 \sqcup_p \cdots \sqcup_p u_{r-1}) \sqcup_p u_r$. We mention a necessary and sufficient condition for the existence of the supremum of several traces that follows directly from the definition.

Lemma 17. *Let $u_1, \ldots, u_r \in \mathbb{M}(\Sigma, I)$. If $u = u_1 \sqcup_p \cdots \sqcup_p u_{r-1}$ exists then the prefix supremum $s = u_1 \sqcup_p \cdots \sqcup_p u_r$ exists if and only if $(u \setminus_p u_r)\ I\ (u_r \setminus_p u)$. In this case $s = u\ (u_r \setminus_p u)$.*

We need the following lemma from [32]:

Lemma 18. *For $u, v \in \mathbb{M}(\Sigma, I)$ we have $u \preceq_p v$ if and only if the word $\pi_{\{a,b\}}(u)$ is a prefix of the word $\pi_{\{a,b\}}(v)$ for all $(a, b) \in D$.*

2.4.2. *Simple facts for compressed traces.*

The following four lemmas state that several operations can be performed in polynomial time on traces, which are represented by SLPs.

Lemma 19. *The following problem can be decided in polynomial time:*

INPUT: A finite independence alphabet (Σ, I) and SLPs $\mathbb{A}$ and $\mathbb{B}$ with terminal alphabet Σ.
QUESTION: Does $[\mathsf{val}(\mathbb{A})]_I \preceq_p [\mathsf{val}(\mathbb{B})]_I$ hold?

Proof. From $\mathbb{A}$ and $\mathbb{B}$ we can compute in polynomial time for all $(a, b) \in D$ SLPs $\mathbb{A}_{a,b}$ and $\mathbb{B}_{a,b}$ with $\mathsf{val}(\mathbb{A}_{a,b}) = \pi_{\{a,b\}}(\mathsf{val}(\mathbb{A}))$ and $\mathsf{val}(\mathbb{B}_{a,b}) = \pi_{\{a,b\}}(\mathsf{val}(\mathbb{B}))$. By Lemma 18, we have to check for all $(a, b) \in D$, whether the word $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{A}))$ is a prefix of the word $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{B}))$. But this can be easily reduced to an equivalence check: Compute $\ell_{a,b} = |\mathsf{val}(\mathbb{A}_{a,b})|$ (using Lemma 4(1)) and an SLP $\mathbb{C}_{a,b}$ with $\mathsf{val}(\mathbb{C}_{a,b}) = \mathsf{val}(\mathbb{B}_{a,b})[: \ell_{a,b}]$ (using Lemma 4(3)). Finally check whether $\mathsf{val}(\mathbb{C}_{a,b}) = \mathsf{val}(\mathbb{A}_{a,b})$ for all $(a, b) \in D$ using e.g. Plandowski's algorithm [41]. □

An analogous statement can be shown for $\preceq_s$.

Lemma 20. *The following problem can be decided in polynomial time:*

INPUT: A finite independence alphabet (Σ, I) and SLPs $\mathbb{A}$ and $\mathbb{B}$ with terminal alphabet Σ.
QUESTION: Does $[\mathsf{val}(\mathbb{A})]_I = [\mathsf{val}(\mathbb{B})]_I$ hold?

Proof. The lemma follows from Lemma 19, since $[\mathsf{val}(\mathbb{A})]_I = [\mathsf{val}(\mathbb{B})]_I$ if and only if $[\mathsf{val}(\mathbb{A})]_I \preceq_p [\mathsf{val}(\mathbb{B})]_I$ and $[\mathsf{val}(\mathbb{B})]_I \preceq_p [\mathsf{val}(\mathbb{A})]_I$. □

Lemma 21. *There is a polynomial time algorithm for the following problem:*

INPUT: A finite independence alphabet (Σ, I) and an SLP $\mathbb{A}$ with terminal alphabet Σ.

OUTPUT: $\max([\mathsf{val}(\mathbb{A})]_I)$ *and* $\min([\mathsf{val}(\mathbb{A})]_I)$

Proof. W.l.o.g. we can assume that $\mathbb{A}$ is in Chomsky normal form. We show how to compute $\max([\mathsf{val}(\mathbb{A})]_I)$. First we compute $\mathsf{alph}(\mathsf{val}(\mathbb{A}))$ in polynomial time using Lemma 4(1). For $a \in \mathsf{alph}(\mathsf{val}(\mathbb{A}))$ let $k_a \in \{1, \ldots, |\mathsf{val}(\mathbb{A})|\}$ maximal such that $\mathsf{val}(\mathbb{A})[k_a] = a$. This number can be computed in polynomial time by the following recursion:

For a nonterminal X with $\mathsf{rhs}_{\mathbb{A}}(X) = b$ with $b \in \Sigma$ we set $k_a(X) = 1$ if $b = a$ and $k_a(X) = 0$ otherwise. For a nonterminal X with $\mathsf{rhs}_{\mathbb{A}}(X) = YZ$ we set:

$$k_a(X) = \begin{cases} 0 & \text{if } a \notin \mathsf{alph}(\mathsf{val}_{\mathbb{A}}(X)) \\ k_a(Y) & \text{if } a \notin \mathsf{alph}(\mathsf{val}_{\mathbb{A}}(Z)) \\ |\mathsf{val}_{\mathbb{A}}(Y)| + k_a(Z) & \text{else.} \end{cases}$$

We set $k_a = k_a(S)$ where S is the initial nonterminal of $\mathbb{A}$. Then $a \in \max([\mathsf{val}(\mathbb{A})]_I)$ if and only if $a\, I\, \mathsf{alph}(\mathsf{val}(\mathbb{A})[k_a + 1 :])$. This property can be checked in polynomial time by first computing (using Lemma 4(3)) an SLP $\mathbb{B}$ for $\mathsf{val}(\mathbb{A})[k_a + 1 :]$ and then computing $\mathsf{alph}(\mathsf{val}(\mathbb{B}))$ (using Lemma 4(1)). Repeating this procedure for all $a \in \mathsf{alph}(\mathsf{val}(\mathbb{A}))$ we get the set $\max([\mathsf{val}(\mathbb{A})]_I)$. The set $\min([\mathsf{val}(\mathbb{A})]_I)$ can be determined similarly. □

Lemma 22. *There is a polynomial time algorithm for the following problem:*

INPUT: A finite independence alphabet (Σ, I) *and an SLP* $\mathbb{A}$ *with terminal alphabet* Σ.
OUTPUT: CC-expressions α, β *with* $[\mathsf{val}_{\mathbb{A}}(\alpha)]_I = [\mathsf{val}(\mathbb{A})]_I \setminus_s \max([\mathsf{val}(\mathbb{A})]_I)$ *and* $[\mathsf{val}_{\mathbb{A}}(\beta)]_I = [\mathsf{val}(\mathbb{A})]_I \setminus_p \min([\mathsf{val}(\mathbb{A})]_I)$

Moreover, $|\alpha|$ *(resp.* $|\beta|$*) can be bounded by* $\mathcal{O}(|\min([\mathsf{val}(\mathbb{A})]_I)| \cdot \log_2(|\mathsf{val}(\mathbb{A})|))$ *(resp.* $\mathcal{O}(|\max([\mathsf{val}(\mathbb{A})]_I)| \cdot \log_2(|\mathsf{val}(\mathbb{A})|))$*).*

Proof. We show how to compute the expression α in polynomial time. By Lemma 21 we can find the set $\max([\mathsf{val}(\mathbb{A})]_I)$ in polynomial time. Let $k_a \in \{1, \ldots, |\mathsf{val}(\mathbb{A})|\}$ be maximal such that $\mathsf{val}(\mathbb{A})[k_a] = a$ for $a \in \max([\mathsf{val}(\mathbb{A})]_I)$ and $\{k_1, \ldots, k_m\} = \{k_a \mid a \in \max([\mathsf{val}(\mathbb{A})]_I)\}$ with $k_1 < k_2 < \cdots < k_m$. These numbers can be computed in polynomial time as well, see the proof of Lemma 21. Let S be the initial variable of $\mathbb{A}$. We set

$$\alpha = S[: k_1 - 1]S[k_1 + 1 : k_2 - 1] \cdots S[k_{m-1} + 1 : k_m - 1]S[k_m + 1 :].$$

Then $[\mathsf{val}_{\mathbb{A}}(\alpha)]_I = [\mathsf{val}(\mathbb{A})]_I \setminus_s \max([\mathsf{val}(\mathbb{A})]_I)$. Since the positions $k_1, \ldots, k_m$ are represented in binary, each of them needs $\mathcal{O}(\log_2(|\mathsf{val}(\mathbb{A})|))$ many bits. Hence $|\alpha|$ can be bounded by $\mathcal{O}(m \cdot \log_2(|\mathsf{val}(\mathbb{A})|))$. Since $m = |\max([\mathsf{val}(\mathbb{A})]_I)|$ we have $|\alpha| \leq \mathcal{O}(|\max([\mathsf{val}(\mathbb{A})]_I)| \cdot \log_2(|\mathsf{val}(\mathbb{A})|))$. Similarly we can compute the expression β. □

2.4.3. *Trace rewriting systems.*

A *trace rewriting system* R over $\mathbb{M}(\Sigma, I)$ is just a finite subset of $\mathbb{M}(\Sigma, I) \times \mathbb{M}(\Sigma, I)$ [12]. We define the *one-step rewrite relation* $\to_R \subseteq \mathbb{M}(\Sigma, I) \times \mathbb{M}(\Sigma, I)$ by: $x \to_R y$ 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 $y = urv$. With $\xrightarrow{*}_R$ we denote the reflexive transitive closure of $\to_R$. The notion of a confluent and terminating trace rewriting system is defined as for other types of rewriting systems [3]: A trace rewriting system R is called *confluent* if for all $u, v, v' \in \mathbb{M}(\Sigma, I)$ with $u \xrightarrow{*}_R v$ and $u \xrightarrow{*}_R v'$ there exists a trace w with $v \xrightarrow{*}_R w$ and $v' \xrightarrow{*}_R w$. It is called *terminating* if there does not exist an infinite chain $u_0 \to_R u_1 \to_R u_2 \cdots$. 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 with $\mathsf{IRR}(R)$. If R is terminating and confluent, then for every trace u, there exists a unique *normal form* $\mathsf{NF}_R(u) \in \mathsf{IRR}(R)$ such that $u \xrightarrow{*}_R \mathsf{NF}_R(u)$.

2.5. *Graph groups*

The *free group* $F(\Sigma)$ generated by the set Σ can be defined as the quotient monoid

$$F(\Sigma) = (\Sigma^{\pm 1})^* / \{aa^{-1} = \varepsilon \mid a \in \Sigma^{\pm 1}\}.$$

Let us fix the independence alphabet (Σ, I) for this subsection. The *graph group* $\mathbb{G}(\Sigma, I)$ is defined as the quotient group

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

Graph groups are also known as right-angled Artin groups and free partially commutative groups. From (Σ, I) we derive the independence alphabet

$$(\Sigma^{\pm 1}, \{(a^{\varepsilon_1}, b^{\varepsilon_2}) \mid (a, b) \in I,\ \varepsilon_1, \varepsilon_2 \in \{-1, 1\}\}).$$

Abusing notation, we denote the independence relation of this alphabet again with I. We consider 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 a CCP-system $\mathbb{A}$ in normal form. Let us fix for the rest of this subsection the trace rewriting system

$$R = \{([aa^{-1}]_I, [\varepsilon]_I) \mid a \in \Sigma^{\pm 1}\}$$

over the trace monoid $\mathbb{M}(\Sigma^{\pm 1}, I)$. Since R is length-reducing, R is terminating. By [12,46], R is also confluent. Note that $(a, b) \in I$ implies $a^{-1}b = ba^{-1}$ in $\mathbb{G}(\Sigma, I)$. Thus, the graph group $\mathbb{G}(\Sigma, I)$ can be also defined as the quotient

$$\mathbb{G}(\Sigma, I) = \mathbb{M}(\Sigma^{\pm 1}, I) / \{aa^{-1} = \varepsilon \mid a \in \Sigma^{\pm 1}\}.$$

Hence, 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 this fact, it was shown in [12,46] that the word problem for $\mathbb{G}(\Sigma, I)$ can be solved in linear time (on the RAM model).

Building on results from [45], Laurence has shown in [28] that automorphism groups of graph groups are finitely generated. Recently, Day [11] proved that automorphism groups of graph groups are in fact finitely presented. Structural results on automorphism groups of graph groups can be found in [6,7]; for a survey see [5].

2.6. *Graph products*

Let us fix for this subsection a *finite* independence alphabet (W, E) with $W = \{1, \ldots, n\}$ and finitely generated groups $\mathbb{G}_i$ for $i \in \{1, \ldots, n\}$. Let $1_{\mathbb{G}_i}$ be the identity element of $\mathbb{G}_i$. For pairwise disjoint nonempty sets $C_1, \ldots, C_n$ we define the independence relation

$$E[C_1, \ldots, C_n] = \bigcup_{(i,j) \in E} C_i \times C_j \tag{1}$$

on the alphabet $\bigcup_{i=1}^n C_i$. Every independence clique of $(\bigcup_{i=1}^n C_i, E[C_1, \ldots, C_n])$ has size at most n. We define a (possibly infinite) independence alphabet as in [13,27]: Let

$$A_i = \mathbb{G}_i \setminus \{1_{\mathbb{G}_i}\} \quad \text{and} \quad A = \bigcup_{i=1}^n A_i.$$

We assume that $A_1, \ldots, A_n$ are pairwise disjoint. We fix the independence relation

$$I = E[A_1, \ldots, A_n]$$

on A for the rest of this subsection. The independence alphabet (A, I) is the only independence alphabet in this paper, which may be infinite. On $\mathbb{M}(A, I)$ we define the trace rewriting system

$$R = \bigcup_{i=1}^n \Big(\{([aa^{-1}]_I, [\varepsilon]_I) \mid a \in A_i\} \cup \{([ab]_I, [c]_I) \mid a, b, c \in A_i, ab = c \text{ in } \mathbb{G}_i\} \Big). \tag{2}$$

The following lemma was shown in [27]:

Lemma 23. *The trace rewriting system R is confluent.*

Since R is length-reducing, it is also terminating and hence defines unique normal forms. We define the *graph product* $\mathbb{G}(W, E, (\mathbb{G}_i)_{i \in W})$ of $\mathbb{G}_1, \ldots, \mathbb{G}_n$ to be the quotient monoid

$$\mathbb{G}(W, E, (\mathbb{G}_i)_{i \in W}) = \mathbb{M}(A, I)/R.$$

It is easy to see that $\mathbb{G}(W, E, (\mathbb{G}_i)_{i \in W})$ is a group. If $\mathbb{G}_i$ is finitely generated by Σ_i, where $\Sigma_i \cap \Sigma_i = \emptyset$ for $i \neq j$, then $\mathbb{G}(W, E, (\mathbb{G}_i)_{i \in W})$ is finitely generated by $\bigcup_{i \in W} \Sigma_i$. If $E = \emptyset$, then $\mathbb{G}(W, E, (\mathbb{G}_i)_{i \in W})$ is the free product $\mathbb{G}_1 * \mathbb{G}_2 * \cdots * \mathbb{G}_n$ and if (W, E) is a complete graph, then $\mathbb{G}(W, E, (\mathbb{G}_i)_{i \in W})$ is the direct product $\prod_{i=1}^n \mathbb{G}_i$. In this sense, the graph product construction generalizes free and direct products.

The following lemma is important for solving the word problem in a graph product $\mathbb{G} = \mathbb{G}(W, E, (\mathbb{G}_i)_{i \in W})$:

Lemma 24. *Let $u, v \in A^*$. Then $u = v$ in $\mathbb{G}$ if and only if $\mathsf{NF}_R([u]_I) = \mathsf{NF}_R([v]_I)$. In particular we have $u = 1$ in $\mathbb{G}$ if and only if $\mathsf{NF}_R([u]_I) = \varepsilon$.*

Proof. The if-direction is trivial. Let on the other hand $u, v \in A^*$ and suppose that $u = v$ in $\mathbb{G}$. By definition this is the case if and only if $[u]_I$ and $[v]_I$ represent the same element from $\mathbb{M}(A, I)/R$ and are hence congruent. Since R produces a normal form for elements from the same congruence class, this implies that $\mathsf{NF}_R([u]_I) = \mathsf{NF}_R([v]_I)$. □

For the normal form of the product of two R-irreducible traces we have the following lemma, which was shown in [13] (equation (21) in the proof of Lemma 22) using a slightly different notation.

Lemma 25. *Let $u, v \in \mathbb{M}(A, I)$ be R-irreducible. Let $x = u \setminus_s v^{-1}$, $y = v \setminus_p u^{-1}$, $x' = x \setminus_s \max(x)$ and $y' = y \setminus_p \min(y)$. Then $\mathsf{NF}_R(uv) = x'\mathsf{NF}_R(\max(x)\min(y))y'$.*

Note that in Lemma 25 $|\max(x)|$ as well as $|\min(y)|$ are bounded by $n = |W|$. Hence, there are at most n rewrite step in the derivation of $\mathsf{NF}_R(\max(x)\min(y))$ from $\max(x)\min(y)$.

Note that graph groups are exactly the graph products of copies of $\mathbb{Z}$. Graph products of copies of $\mathbb{Z}/2\mathbb{Z}$ are known as *right-angled Coxeter groups*, see [15] for more details. It is not clear, whether the automorphism group of a graph product of finitely generated groups with finitely generated automorphism groups is itself finitely generated. [c] Generalizing the main result from [28], it was shown in [9] that the automorphism group of a graph product of finitely generated Abelian groups is finitely generated. In particular, the automorphism group of a right-angled Coxeter group is finitely generated.

3. Main results and applications

In this section we will present the main results of this paper, the proofs of which are subject to the rest of the paper. Recall that the main goal of this paper is to analyze the complexity of the word problem for finitely generated subgroups of $\mathsf{Aut}(\mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W}))$ and $\mathsf{Out}(\mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W}))$. Using Proposition 9 and 10, we have to study the problems $\mathsf{CWP}(\mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W}))$ and $\mathsf{RSCCP}(\mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W}), \Sigma)$ (for a generating set Σ). The following two theorems are our main results for these problems:

Theorem 26. *Let (W, E) be a fixed finite independence alphabet and let $\mathbb{G}_i$ ($i \in W$) be finitely generated groups. Then $\mathsf{CWP}(\mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W}))$ is polynomial time Turing reducible to the problems $\mathsf{CWP}(\mathbb{G}_i)$ ($i \in W$).*

[c] We conjecture that using the methods from [28] one can indeed show that the automorphism group of a graph product of finitely generated groups with finitely generated automorphism groups is itself finitely generated.

Theorem 27. *Let (W, E) be a fixed finite independence alphabet, $\mathbb{G}_i$ $(i \in W)$ be finitely generated groups, and $\mathbb{G} = \mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W})$. Let $B_i \subseteq \mathbb{G}_i$ be finite non-empty sets $(i \in W)$. Then $\mathsf{RSCCP}(\mathbb{G}, \bigcup_{i\in W} B_i)$ is polynomial time Turing reducible to the problems $\mathsf{CWP}(\mathbb{G}_i)$ and $\mathsf{RSCCP}(\mathbb{G}_i, B_i)$ $(i \in W)$.*

Theorem 26 will be shown in Section 4, whereas Theorem 27 will be shown in Section 8. Theorem 26 (resp., Theorem 27) generalizes a corresponding result from [34] (resp. [23]) for graph groups.

Remark 28. We can use Proposition 10 and Theorem 27 to infer the following result about the word problem of the outer automorphism group of a graph product: Let (W, E) be a fixed finite independence alphabet, let the group $\mathbb{G}_i$ be finitely generated by Σ_i for $i \in W$ and let $\mathbb{G} = \mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W})$. Let furthermore $\mathbb{H}$ be a finitely generated subgroup of $\mathsf{Out}(\mathbb{G})$. Then $\mathsf{WP}(\mathbb{H})$ is polynomial time Turing reducible to the problems $\mathsf{CWP}(\mathbb{G}_i)$ and $\mathsf{RSCCP}(\mathbb{G}_i, \Sigma_i)$ $(i \in W)$.

Since the compressed word problem as well as the restricted simultaneous compressed conjugacy problem for $\mathbb{Z}$ and every finite group can be solved in polynomial time, Theorem 26 and 27 imply:

Corollary 29. *If $\mathbb{G} = \mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W})$ is a graph product, where each $\mathbb{G}_i$ is either finite or isomorphic to $\mathbb{Z}$, then the problems $\mathsf{CWP}(\mathbb{G})$ and $\mathsf{RSCCP}(\mathbb{G}, B)$ (where $B = \bigcup_{i\in W} B_i$ with $B_i \subseteq \mathbb{G}_i$) can be solved in polynomial time. In particular, these problems can be solved in polynomial time for*

- *right-angled Artin groups and*
- *right-angled Coxeter groups.*

Corollary 29 and Proposition 9 and 10 imply:

Corollary 30. *If $\mathbb{G}$ is a graph product of finite groups and copies of $\mathbb{Z}$, then the word problem for every finitely generated subgroup of $\mathsf{Aut}(\mathbb{G})$ or $\mathsf{Out}(\mathbb{G})$ can be solved in polynomial time. In particular, the word problems for automorphism groups and outer automorphism groups of*

- *right-angled Artin groups and*
- *right-angled Coxeter groups*

can be solved in polynomial time.

Finally, in Section 11, we prove that the compressed conjugacy problem for a graph group can be decided in polynomial time:

Theorem 31. *Let (Σ, I) be a fixed finite independence alphabet. Then $\mathsf{CCP}(\mathbb{G}(\Sigma, I))$ can be solved in polynomial time.*

We conjecture the following generalization of Theorem 31:

Conjecture 32. *Let (W, E) be a fixed finite independence alphabet and let $\mathbb{G}_i$ $(i \in W)$ be finitely generated groups. Then $\mathsf{CCP}(\mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W}))$ is polynomial time Turing reducible to the problems $\mathsf{CCP}(\mathbb{G}_i)$ $(i \in W)$.*

4. The compressed word problem for graph products

In this section, we will prove Theorem 26. For this, we combine methods used in [34] and [13]. Let us fix the *finite* independence alphabet (W, E) with $W = \{1, \ldots, n\}$ and finitely generated groups $\mathbb{G}_i$ for $i \in \{1, \ldots, n\}$ for the rest of this section. Let furthermore Σ_i be a finite generating set for $\mathbb{G}_i$ for $i \in \{1, \ldots, n\}$. W.l.o.g. we can assume that Σ_i does not contain the identity element and that $\Sigma_i \cap \Sigma_j = \emptyset$ for $i \neq j$. We define $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$. Let $\mathbb{G}$ denote the graph product $\mathbb{G}(W, E, (\mathbb{G}_i)_{i\in W})$ for the rest of this section. Moreover, let A_i, A, I, and R have the same meaning as in Section 2.6. Note that $\Sigma_i \subseteq A_i$ for all $1 \leq i \leq n$.

For the following discussion, let us fix a 2-level CCP-system

$$\mathbb{B} = (\mathsf{Up}, \mathsf{Lo}, \Sigma^{\pm 1}, \mathsf{rhs}, S)$$

over the terminal alphabet $\Sigma^{\pm 1}$ (the monoid generating set of our graph product $\mathbb{G}$). We introduce several properties for $\mathbb{B}$.

Definition 33 (pure). *$\mathbb{B}$ is pure if for every $X \in \mathsf{Lo}$ there exists $i \in W$ such that $\mathsf{val}(X) \in (\Sigma_i^{\pm 1})^+$ and $\mathsf{val}(X) \neq 1$ in $\mathbb{G}_i$ (hence $\mathsf{val}(X)$ represents a group element from the set A).*

For the following notations, assume that $\mathbb{B}$ is pure. Then, we can define the mapping $\mathsf{type}_{\mathbb{B}} : \mathsf{Lo} \to W$ by $\mathsf{type}_{\mathbb{B}}(X) = i$ if $\mathsf{val}(X) \in (\Sigma_i^{\pm 1})^+$. For $i \in W$ let

$$\mathsf{Lo}(i) = \{X \in \mathsf{Lo} \mid \mathsf{type}_{\mathbb{B}}(X) = i\}.$$

Then the sets $\mathsf{Lo}(1), \ldots, \mathsf{Lo}(n)$ form a partition of Lo. Moreover, using (1) on page 18 we can define an independence relation $I_{\mathbb{B}}$ on Lo by

$$I_{\mathbb{B}} = E[\mathsf{Lo}(1), \ldots, \mathsf{Lo}(n)].$$

Definition 34 (nicely projecting). *$\mathbb{B}$ is nicely projecting if for every subexpression of the form $\pi_\Delta(\alpha)$ $(\Delta \subseteq \mathsf{Lo})$ that appears in a right-hand side of $\mathsf{up}(\mathbb{B})$, there exists $K \subseteq W$ with $\Delta = \bigcup_{i\in K} \mathsf{Lo}(i)$.*

This condition will be needed in order to apply Lemma 3. Note that the number of all sets $\bigcup_{i\in K} \mathsf{Lo}(i)$ with $K \subseteq W$ is bounded by $2^n = \mathcal{O}(1)$.

Definition 35 (irredundant). *$\mathbb{B}$ is irredundant if for all $X, Y \in \mathsf{Lo}$ such that $X \neq Y$ and $\mathsf{type}_{\mathbb{B}}(X) = \mathsf{type}_{\mathbb{B}}(Y) = i$, we have $\mathsf{val}(X) \neq \mathsf{val}(Y)$ in $\mathbb{G}_i$.*

One can think of a pure and irredundant 2-level CCP-system $\mathbb{B}$ as a CCP-system, where the terminal alphabet is a finite subset $B \subseteq A$, with $A = \bigcup_{i\in W} \mathbb{G}_i \setminus \{1_{\mathbb{G}_i}\}$ from Section 2.6. Moreover, each element from $B \cap A_i$ $(i \in W)$ is represented by a unique SLP over the terminal alphabet $\Sigma_i^{\pm 1}$ (namely the lower part $\mathsf{lo}(\mathbb{B})$ with

the appropriate initial variable). If $\mathbb{B}$ is pure but not irredundant then, using oracle access to the compressed word problems for the groups $\mathbb{G}_i$, one can compute a pure and irredundant 2-level CCP-system $\mathbb{C}$ such that $\mathsf{val}(\mathbb{B}) = \mathsf{val}(\mathbb{C})$ in $\mathbb{G}$ as follows: If $\mathbb{B}$ contains two variables $X, Y \in \mathsf{Lo}$ such that $X \neq Y$, $\mathsf{type}_{\mathbb{B}}(X) = \mathsf{type}_{\mathbb{B}}(Y) = i$ and $\mathsf{val}(X) = \mathsf{val}(Y)$ in $\mathbb{G}_i$, one has to replace Y in all right-hand sides by X. Note that this process does not change the set of upper level variables of $\mathbb{B}$.

Definition 36 (saturated). $\mathbb{B}$ *is* saturated *if for every* $X \in \mathsf{Lo}$ *with* $\mathsf{type}_{\mathbb{B}}(X) = i$, *there exists* $Y \in \mathsf{Lo}$ *with* $\mathsf{type}_{\mathbb{B}}(Y) = i$ *and* $\mathsf{val}_{\mathbb{B}}(Y) = \mathsf{val}_{\mathbb{B}}(X)^{-1}$ *in* $\mathbb{G}_i$.

If $\mathbb{B}$ is pure, irredundant and saturated, then for every $X \in \mathsf{Lo}$ with $\mathsf{type}_{\mathbb{B}}(X) = i$, there must be a unique $Y \in \mathsf{Lo}$ with $\mathsf{type}_{\mathbb{B}}(Y) = i$ and $\mathsf{val}_{\mathbb{B}}(Y) = \mathsf{val}_{\mathbb{B}}(X)^{-1}$ in $\mathbb{G}_i$ (we may have $Y = X$ in case $\mathsf{val}_{\mathbb{B}}(X)^2 = 1$ in $\mathbb{G}_i$). This Y is denoted with X^{-1}, and we define $(X_1 \cdots X_n)^{-1} = X_n^{-1} \cdots X_1^{-1}$ for $X_1, \ldots, X_n \in \mathsf{Lo}$.

Definition 37 (well-formed). $\mathbb{B}$ *is well-formed, if it is pure, irredundant, saturated, and nicely projecting.*

Assume that $\mathbb{B}$ is well-formed. We call a trace $w \in \mathbb{M}(\mathsf{Lo}, I_{\mathbb{B}})$ *reduced* if it contains no factor $[YZ]_{I_{\mathbb{B}}}$ with $Y, Z \in \mathsf{Lo}$ and $\mathsf{type}_{\mathbb{B}}(Y) = \mathsf{type}_{\mathbb{B}}(Z)$. Note that $[X_1 \cdots X_m]_{I_{\mathbb{B}}} \in \mathbb{M}(\mathsf{Lo}, I_{\mathbb{B}})$ with $X_1, \ldots, X_m \in \mathsf{Lo}$ is reduced if and only if $[a_1 \cdots a_m]_I \in \mathsf{IRR}(R)$, where $a_j \in A$ is the group element represented by $\mathsf{val}(X_j)$ for $1 \leq j \leq m$. A variable $X \in \mathsf{Up} \cup \mathsf{Lo}$ is reduced if either $X \in \mathsf{Lo}$ or $X \in \mathsf{Up}$ and the trace $[\mathsf{uval}(X)]_{I_{\mathbb{B}}}$ is reduced. Finally, $\mathbb{B}$ is reduced, if every variable X of $\mathbb{B}$ is reduced. We have:

Lemma 38. *Let* $\mathbb{B}$ *be a well-formed and reduced 2-level CCP-system. Then* $\mathsf{val}(\mathbb{B}) = 1$ *in* $\mathbb{G}$ *if and only if* $\mathsf{uval}(\mathbb{B}) = \varepsilon$.

Proof. Clearly, if $\mathsf{uval}(\mathbb{B}) = \varepsilon$, then also $\mathsf{val}(\mathbb{B}) = \varepsilon$ and hence $\mathsf{val}(\mathbb{B}) = 1$ in $\mathbb{G}$. For the other direction we assume for contradiction that $\mathsf{uval}(\mathbb{B}) = X_1 \cdots X_m$ for some $m > 0$. Since $\mathbb{B}$ is pure there are $a_1, \ldots, a_m \in A$ such that $\mathsf{val}(X_i)$ represents the group element a_i for $i \in \{1, \ldots, m\}$. Since $\mathbb{B}$ is reduced, we have $[a_1 \cdots a_m]_I \in \mathsf{IRR}(R)$ and hence $\mathsf{NF}_R([a_1 \cdots a_m]_I) = [a_1 \cdots a_m]_I \neq \varepsilon$. From Lemma 24 it follows that $a_1 \cdots a_m \neq 1$ in $\mathbb{G}$ and hence $\mathsf{val}(\mathbb{B}) \neq 1$ in $\mathbb{G}$. □

Together with Lemma 38, the following proposition can be used to solve the compressed word problem for the graph product $\mathbb{G}$.

Proposition 39. *Given an SLP* $\mathbb{A}$ *over* $\Sigma^{\pm 1}$ *we can compute a reduced and well-formed 2-level CCP-system* $\mathbb{B}$ *with* $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$ *in* $\mathbb{G}$ *in polynomial time using oracle access to the decision problems* $\mathsf{CWP}(\mathbb{G}_i)$ $(1 \leq i \leq n)$.

Proof. Let $\mathbb{A} = (V_{\mathbb{A}}, \Sigma, \mathsf{rhs}_{\mathbb{A}}, S)$ be the given input SLP over $\Sigma^{\pm 1}$. We assume w.l.o.g. that $\mathbb{A}$ is in Chomsky normal form. Moreover, we exclude the trivial case that $\mathsf{rhs}_{\mathbb{A}}(S) \in \Sigma^{\pm 1}$. We construct a sequence of 2-level CCP-systems $\mathbb{A}_j = (\mathsf{Up}_j, \mathsf{Lo}_j, \Sigma^{\pm 1}, \mathsf{rhs}_j, S)$ $(0 \leq j \leq r \leq |V_{\mathbb{A}}|)$ such that the following holds for all $0 \leq j \leq r$:

(a) $\mathbb{A}_j$ is well-formed.
(b) $|\mathbb{A}_j| \leq 2 \cdot |\mathbb{A}| + \mathcal{O}(j \cdot |\mathbb{A}|) \leq 2 \cdot |\mathbb{A}| + \mathcal{O}(|V_{\mathbb{A}}| \cdot |\mathbb{A}|)$
(c) $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{A}_j)$ in $\mathbb{G}$ for all $0 \leq j \leq r$.
(d) If $X \in \mathsf{Up}_j$ is not reduced, then $\mathsf{rhs}_j(X) \in (\mathsf{Up}_j \cup \mathsf{Lo}_j)^2$.
(e) $|\mathsf{uval}(\mathbb{A}_j)| \leq |\mathsf{val}(\mathbb{A})|$

Moreover, the final 2-level CCP-system $\mathbb{B} = \mathbb{A}_r$ will be reduced. Let us write type_j for $\mathsf{type}_{\mathbb{A}_j}$ and I_j for $I_{\mathbb{A}_j}$ in the following.

During the construction of $\mathbb{A}_{j+1}$ from $\mathbb{A}_j$, we will replace the right-hand side YZ ($Y, Z \in \mathsf{Up}_j \cup \mathsf{Lo}_j$) for a non-reduced (in $\mathbb{A}_j$) variable $X \in \mathsf{Up}_j$ by a new right-hand side of size $\mathcal{O}(|\mathbb{A}|)$, so that X is reduced in $\mathbb{A}_{j+1}$ and $\mathsf{val}_{\mathbb{A}_j}(X) = \mathsf{val}_{\mathbb{A}_{j+1}}(X)$ in $\mathbb{G}$. All other right-hand sides for upper level variables will be kept, and constantly many new lower level variables will be added.

We start the construction with the 2-level CCP-system

$$(\{X \in V_{\mathbb{A}} \mid \mathsf{rhs}_{\mathbb{A}}(X) \in V_{\mathbb{A}}^2\}, \{X \in V_{\mathbb{A}} \mid \mathsf{rhs}_{\mathbb{A}}(X) \in \Sigma^{\pm 1}\}, \Sigma^{\pm 1}, \mathsf{rhs}_{\mathbb{A}}, S).$$

Note that S is an upper level variable in this system (which is required for 2-level CCP-systems) since we assume $\mathsf{rhs}_{\mathbb{A}}(S) \notin \Sigma^{\pm 1}$. Moreover, the system is pure and nicely projecting (there are no projection operations in right-hand sides), but not necessarily irredundant and saturated. The latter two properties can be easily enforced by adding for every variable X with $\mathsf{rhs}_{\mathbb{A}}(X) = a \in \Sigma^{\pm 1}$ a variable X^{-1} for a^{-1} and then eliminating redundant lower level variables. The resulting 2-level CCP-system $\mathbb{A}_0$ is well-formed and satisfies $|\mathbb{A}_0| \leq 2 \cdot |\mathbb{A}|$ and $\mathsf{val}(\mathbb{A}_0) = \mathsf{val}(\mathbb{A})$. Hence (a), (b), and (c) are satisfied and also (d) and (e) clearly hold.

For the inductive step of the construction, assume that we have constructed $\mathbb{A}_j = (\mathsf{Up}_j, \mathsf{Lo}_j, \Sigma^{\pm 1}, \mathsf{rhs}_j, S)$ and let $X \in \mathsf{Up}_j$, $Y, Z \in \mathsf{Up}_j \cup \mathsf{Lo}_j$ such that $\mathsf{rhs}_j(X) = YZ$, X is not reduced, but Y and Z are already reduced. In order to make X reduced, we will apply Lemma 25. The following proposition, whose proof is postponed to the next Section 5, makes this application possible.

Proposition 40. *Let (W, E) be a fixed independence alphabet with $W = \{1, \ldots, n\}$. The following problem can be solved in polynomial time:*

INPUT: Pairwise disjoint finite alphabets $\Gamma_1, \ldots, \Gamma_n$, an SLP $\mathbb{B}$ over the terminal alphabet $\Gamma = \bigcup_{i=1}^n \Gamma_i$, and two variables Y and Z from $\mathbb{B}$.
OUTPUT: CCP-expressions $\alpha, \beta \in \mathsf{CCP}(\{Y\}, \emptyset)$ such that the following holds, where $J = E[\Gamma_1, \ldots, \Gamma_n]$:

(1) For every subexpression of the form $\pi_\Delta(\gamma)$ in α and β there exists $K \subseteq \{1, \ldots, n\}$ with $\Delta = \bigcup_{i \in K} \Gamma_i$.
(2) $[\mathsf{val}_{\mathbb{B}}(\alpha)]_J = [\mathsf{val}_{\mathbb{B}}(Y)]_J \setminus_p [\mathsf{val}_{\mathbb{B}}(Z)]_J$
(3) $[\mathsf{val}_{\mathbb{B}}(\beta)]_J = [\mathsf{val}_{\mathbb{B}}(Y)]_J \sqcap_p [\mathsf{val}_{\mathbb{B}}(Z)]_J$
(4) $|\alpha|, |\beta| \leq \mathcal{O}(\log_2(|\mathsf{val}(\mathbb{B})|))$

An analogous statement can be shown for the operations $\setminus_s$ and $\sqcap_s$ which refer to the suffix order on traces.

In order to apply Proposition 40 to our situation we transform the upper level part $\mathsf{up}(\mathbb{A}_j)$ into an equivalent SLP $\mathbb{C}$ over Lo_j using Lemma 3. This is possible, since $\mathbb{A}_j$ is nicely projecting by (a). The time for this step is polynomially bounded in $|\mathsf{up}(\mathbb{A}_j)|$ and hence in $|\mathbb{A}|$. Note that we have $|\mathsf{val}(\mathbb{C})| = |\mathsf{uval}(\mathbb{A}_j)| \leq |\mathsf{val}(\mathbb{A})| \leq 2^{|\mathbb{A}|}$ by (e). Now we set $\Gamma_i = \mathsf{Lo}_j(i)$ and apply Proposition 40 to Γ_i ($i \in W$) and $\mathbb{C}$ to obtain two CCP-expressions α and β such that $|\alpha|, |\beta| \leq \mathcal{O}(|\mathbb{A}|)$ and (uval denotes $\mathsf{uval}_{\mathbb{A}_j}$)

$$\begin{aligned} [\mathsf{uval}(\alpha)]_{I_j} &= [\mathsf{uval}(Y)]_{I_j} \setminus_s [\mathsf{uval}(Z)]_{I_j}^{-1} \\ [\mathsf{uval}(\beta)]_{I_j} &= [\mathsf{uval}(Z)]_{I_j} \setminus_p [\mathsf{uval}(Y)]_{I_j}^{-1}. \end{aligned}$$

Moreover, for every subexpression of the form $\pi_\Delta(\gamma)$ in α or β there exists $K \subseteq \{1, \ldots, n\}$ with $\Delta = \bigcup_{i \in K} \mathsf{Lo}_j(i)$. Intuitively, α and β represent the parts of $\mathsf{uval}(Y)$ and $\mathsf{uval}(Z)$ that remain after cancellation in the graph group generated by the alphabet Lo_j. Hence, $[\mathsf{uval}(\alpha)]_{I_j}[\mathsf{uval}(\beta)]_{I_j}$ does not contain a factor of the form $[XX^{-1}]_{I_j}$ for $X \in \mathsf{Lo}_j$. Using Lemma 21 and 22 (in order to apply these lemmas, we have to compute, using Lemma 3, SLPs for the strings $\mathsf{uval}(\alpha)$ and $\mathsf{uval}(\beta)$) we can compute $V_{\max} = \max([\mathsf{uval}(\alpha)]_{I_j})$, $V_{\min} = \min([\mathsf{uval}(\beta)]_{I_j})$, and CCP-expressions α', β' such that

$$\begin{aligned} [\mathsf{uval}(\alpha')]_{I_j} &= [\mathsf{uval}(\alpha)]_{I_j} \setminus_s [V_{\max}]_{I_j} \\ [\mathsf{uval}(\beta')]_{I_j} &= [\mathsf{uval}(\beta)]_{I_j} \setminus_p [V_{\min}]_{I_j}. \end{aligned}$$

Recall that $V_{\max}$ and $V_{\min}$ are subsets of Lo_j. The form of the independence relation I_j implies that $|V_{\max}|, |V_{\min}| \leq n = \mathcal{O}(1)$. The length bound in Lemma 22 implies that $|\alpha'|, |\beta'| \leq \mathcal{O}(|\mathbb{A}|)$. Moreover, for every $1 \leq i \leq n$ we must have $|V_{\max} \cap \mathsf{Lo}_j(i)| \leq 1$ and $|V_{\min} \cap \mathsf{Lo}_j(i)| \leq 1$. Let

$$\begin{aligned} V'_{\max} &= \{X \in V_{\max} \mid \mathsf{type}_j(X) \notin \mathsf{type}_j(V_{\min})\} \\ V'_{\min} &= \{X \in V_{\min} \mid \mathsf{type}_j(X) \notin \mathsf{type}_j(V_{\max})\}. \end{aligned}$$

If $(X_1, X_2) \in V_{\max} \times V_{\min}$ is such that $\mathsf{type}_j(X_1) = \mathsf{type}_j(X_2) = i$, then by the definition of $[\mathsf{uval}(\alpha)]_{I_j}$ and $[\mathsf{uval}(\beta)]_{I_j}$, we must have $\mathsf{val}(X_1)\mathsf{val}(X_2) \neq 1$ in $\mathbb{G}_i$. For each such pair we add a new lower level variable X_{X_1,X_2} to Lo_j with right-hand side X_1X_2; let V' be the set of these new variables. Clearly, $|V'| \leq n = \mathcal{O}(1)$. Finally, the right-hand side for X is changed to the CCP-expression

$$\gamma = \alpha' \, v'_{\max} \, v' \, v'_{\min} \, \beta'$$

where $v'_{\max}$ (resp. v', $v'_{\min}$) is an arbitrary string that enumerates all variables from $V'_{\max}$ (resp. V', $V'_{\min}$). We have $|\gamma| = |\alpha'| + |\beta'| + \mathcal{O}(1) \leq \mathcal{O}(|\mathbb{A}|)$. Clearly, γ evaluates in $\mathbb{G}$ to the same group element as $\mathsf{val}_{\mathbb{A}_j}(X)$. By adding at most $|V'| = \mathcal{O}(1)$ many further lower level variables, we obtain a saturated system. The resulting 2-level CCP-system is not necessarily irredundant, but this can be ensured, as explained

above, using oracle calls to the compressed word problems for the vertex groups $\mathbb{G}_i$ (this does not increase the size of the 2-level CCP-system). The resulting system is pure, irredundant, and saturated, but not necessarily nicely projecting, because of the new lower level nonterminals from V'. But note that these variables do not occur in the scope of a projection operator π_Δ. Hence, we may add the new lower level variables to the appropriate sets appearing in projection operators, so that the 2-level CCP-system becomes nicely projecting as well. The resulting 2-level CCP-system is $\mathbb{A}_{j+1}$; it is well-formed. Its size can be bounded by $|\mathbb{A}_j| + \mathcal{O}(|\mathbb{A}|) \leq 2 \cdot |\mathbb{A}| + \mathcal{O}(j \cdot |\mathbb{A}|) + \mathcal{O}(|\mathbb{A}|) \leq 2 \cdot |\mathbb{A}| + \mathcal{O}((j+1) \cdot |\mathbb{A}|)$; hence (a) and (b) above hold for $\mathbb{A}_{j+1}$. Moreover, in the group $\mathbb{G}$ we have $\mathsf{val}(\mathbb{A}_{j+1}) = \mathsf{val}(\mathbb{A}_j) = \mathsf{val}(\mathbb{A})$, hence (c) holds. Lemma 25 implies that X is reduced in $\mathbb{A}_{j+1}$ which implies property (d) for $\mathbb{A}_{j+1}$. Finally, for (e) note that $|\mathsf{uval}(\mathbb{A}_{j+1})| \leq |\mathsf{uval}(\mathbb{A}_j)| \leq |\mathsf{val}(\mathbb{A})|$.

After $r \leq |V_\mathbb{A}|$ steps, our construction yields the well-formed and reduced 2-level CCP-system $\mathbb{A}_r$ with $\mathsf{val}(\mathbb{A}_r) = \mathsf{val}(\mathbb{A})$ in $\mathbb{G}$ which proves Proposition 39. □

Now the proof of Theorem 26 is straightforward: By Proposition 39 we can translate a given SLP $\mathbb{A}$ into an equivalent, reduced and well-formed 2-level CCP-system $\mathbb{B}$ with $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$ in $\mathbb{G}$ in polynomial time using oracle access to $\mathsf{CWP}(\mathbb{G}_i)$ for $i \in W$. By Lemma 38, we have $\mathsf{val}(\mathbb{A}) = 1$ in $\mathbb{G}$ if and only if $\mathsf{uval}(\mathbb{B}) = \varepsilon$. We can use Lemma 3 to translate $\mathsf{up}(\mathbb{B})$ into an equivalent SLP, for which it is trivial to check whether the empty word is produced. This proves Theorem 26.

5. Proof of Proposition 40

We will prove Proposition 40 in this section. Recall that we fixed the finite undirected graph (W, E) with $W = \{1, \ldots, n\}$.

Proposition 40. *Let (W, E) be a fixed independence alphabet with $W = \{1, \ldots, n\}$. The following problem can be solved in polynomial time:*

INPUT: Pairwise disjoint finite alphabets $\Gamma_1, \ldots, \Gamma_n$, an SLP $\mathbb{B}$ over the terminal alphabet $\Gamma = \bigcup_{i=1}^n \Gamma_i$, and two variables Y and Z from $\mathbb{B}$.
OUTPUT: CCP-expressions $\alpha, \beta \in \mathsf{CCP}(\{Y\}, \emptyset)$ such that the following holds, where $I = E[\Gamma_1, \ldots, \Gamma_n]$:

(1) For every subexpression of the form $\pi_\Delta(\gamma)$ in α or β there exists $K \subseteq W$ with $\Delta = \bigcup_{i \in K} \Gamma_i$.

(2) $[\mathsf{val}_\mathbb{B}(\alpha)]_I = [\mathsf{val}_\mathbb{B}(Y)]_I \setminus_p [\mathsf{val}_\mathbb{B}(Z)]_I$

(3) $[\mathsf{val}_\mathbb{B}(\beta)]_I = [\mathsf{val}_\mathbb{B}(Y)]_I \sqcap_p [\mathsf{val}_\mathbb{B}(Z)]_I$

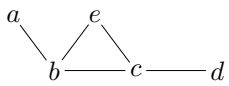
(4) $|\alpha|, |\beta| \leq \mathcal{O}(\log_2(|\mathsf{val}(\mathbb{B})|))$

In the following, we will write $\setminus$ and $\sqcap$ for $\setminus_p$ and $\sqcap_p$, respectively. Moreover, if $\Delta = \bigcup_{i \in K} \Gamma_i$, we will write π_K for the projection morphism $\pi_\Delta : \Gamma^* \to \Gamma^*$. Let us fix $\Gamma = \bigcup_{i=1}^n \Gamma_i$ and let $I = E[\Gamma_1, \ldots, \Gamma_n]$.

Let $s \in \Gamma^*$ be a string and $J \subseteq \{1, \ldots, |s|\}$ a set of positions in s. Below, we will identify the dependence graph D_s with the edge relation of D_s. We are looking for a compact representation for the set of all positions p such that $\exists j \in J : (j,p) \in D_s^*$, i.e., there exists a path in the dependence graph D_s from some position $j \in J$ to position p. For $i \in W$ define

$$\mathsf{pos}(s, J, i) = \min(\{|s|+1\} \cup \{p \mid 1 \leq p \leq |s|, s[p] \in \Gamma_i, \exists j \in J : (j,p) \in D_s^*\}).$$

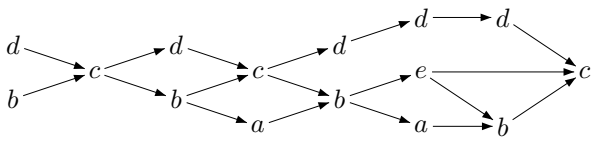
Example 41. To ease the reading we will consider the set $W = \{a, b, c, d, e\}$ instead of $W = \{1, \ldots, 5\}$. The dependence relation D is:



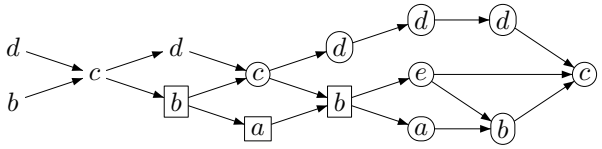
Let $\Gamma_x = \{x\}$ for $x \in W$. We consider the following string:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$s =$	d	b	c	d	b	a	c	d	b	d	e	a	b	d	c

The dependence graph of s look as follows:

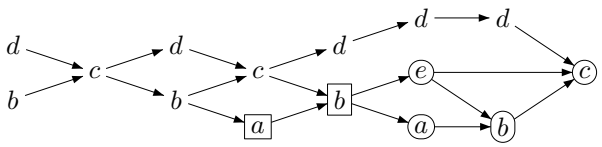


Let $J = \{5, 6, 9\}$. We want to determine $\mathsf{pos}(s, J, 4)$. In the following picture we mark positions from J with boxes and all positions $p \notin J$ with $(j,p) \in D_s^*$ for some $j \in J$ with circles.



The positions with letters from $\Gamma_4 = \{d\}$ which depend from positions from J are $\{8, 10, 14\}$ with the minimum 8, hence $\mathsf{pos}(s, J, 4) = 8$.

For the set $J = \{6, 9\}$ we get the following picture for $\mathsf{pos}(s, J, 4)$:



Since there are no positions with letters from $\Gamma_4 = \{d\}$ which depend from positions from J, it follows that $\mathsf{pos}(s, J, 4) = |s| + 1 = 16$.

Instead of $\mathsf{pos}(s, \{p\}, i)$, we simply write $\mathsf{pos}(s, p, i)$. Note that $\mathsf{pos}(s, \emptyset, i) = |s| + 1$. The definition of $\mathsf{pos}(s, J, i)$ and the fact that symbols from a set Γ_i are pairwise dependent implies:

Lemma 42. *Let $s \in \Gamma^*$ and $J \subseteq \{1, \ldots, |s|\}$. Then for every position $1 \leq p \leq |s|$ the following two properties are equivalent:*

- $\exists j \in J : (j, p) \in D_s^*$
- *If $s[p] \in \Gamma_i$ then $p \geq \mathsf{pos}(s, J, i)$.*

We will also need the following lemma:

Lemma 43. *For a given SLP $\mathbb{A}$, a position $1 \leq p \leq |\mathsf{val}(\mathbb{A})|$ and $i \in W$, we can compute the position $\mathsf{pos}(\mathsf{val}(\mathbb{A}), p, i)$ in polynomial time.*

Proof. We first need a few definitions: Let $D = (W \times W) \setminus E$ be the dependence relation for our fixed independence alphabet (W, E). A path in (W, D) (viewed as an undirected graph) is called *simple*, if it does not visit a node twice. For $j \in W$ let $\mathcal{P}_j$ be the set of all simple paths in the dependence alphabet (W, D) that start in the node j. The path, which only consists of the node j belongs to $\mathcal{P}_j$. Note that $|\mathcal{P}_j| \in \mathcal{O}(1)$ since (W, E) is fixed.

Let us now fix $\mathbb{A}$, p, and i as in the lemma. Assume that $\mathsf{val}(\mathbb{A})[p] \in \Gamma_j$. By Lemma 4(2), the node $j \in W$ can be computed in polynomial time. For a simple path $\rho \in \mathcal{P}_j$ let us define $\mathsf{pos}(p, \rho) \in \{1, \ldots, |\mathsf{val}(\mathbb{A})|+1\}$ inductively. If $\rho = (j)$, then $\mathsf{pos}(p, \rho) = p$. If $\rho = (\rho', k)$, where $\rho' \in \mathcal{P}_j$ and $k \in W$, then $\mathsf{pos}(p, \rho)$ is the smallest position $q > \mathsf{pos}(p, \rho')$ such that $\mathsf{val}(\mathbb{A})[q] \in \Gamma_k$ if such a position exists, otherwise $\mathsf{pos}(p, \rho) = |\mathsf{val}(\mathbb{A})| + 1$. It follows that $\mathsf{pos}(p, \rho)$ can be computed in polynomial time for every simple path $\rho \in \mathcal{P}_j$. Finally, $\mathsf{pos}(\mathsf{val}(\mathbb{A}), p, i)$ is the minimum over all these positions for all simple paths from j to i. □

Let us now come back to the problem of constructing a CCP-expression, which evaluates to $[\mathsf{val}_{\mathbb{B}}(Y)]_I \setminus [\mathsf{val}_{\mathbb{B}}(Z)]_I$ and $[\mathsf{val}_{\mathbb{B}}(Y)]_I \sqcap [\mathsf{val}_{\mathbb{B}}(Z)]_I$. Let us first solve this problem for uncompressed strings. Then we will argue that our algorithm leads to a polynomial time algorithm for compressed input strings.

How can we compute for two given words $s, t \in \Gamma^*$ words $\mathsf{inf}, \mathsf{diff} \in \Gamma^*$ such that $[\mathrm{inf}]_I = [s]_I \sqcap [t]_I$ and $[\mathsf{diff}]_I = [s]_I \setminus [t]_I$? In the algorithm in Figure 1 we accumulate the strings inf and diff by determining for every position from $\{1, \ldots, |s|\}$ (viewed as a node of the dependence graph D_s) whether it belongs to $[\mathrm{inf}]_I$ or $[\mathsf{diff}]_I$. For this, we will store a current position ℓ in the string s, which will increase during the computation. Initially, we set $\ell := 1$ and $\mathrm{inf} := \varepsilon$, $\mathsf{diff} := \varepsilon$. At the end, we have $[\mathrm{inf}]_I = [s]_I \sqcap [t]_I$ and $[\mathsf{diff}]_I = [s]_I \setminus [t]_I$.

For a set of positions $K \subseteq \{1, \ldots, |s|\}$ let us define the string $s{\restriction}K = s[\ell_1] \cdots s[\ell_k]$, where $\ell_1 < \ell_2 < \cdots < \ell_k$ and $K = \{\ell_1, \ldots, \ell_k\}$. Consider a specific iteration

$\ell := 1$; (stores a position in s)
$\mathsf{inf} := \varepsilon$; (stores a string)
$\mathsf{diff} := \varepsilon$; (stores a string)
$\mathsf{pos}(i) := |s| + 1$ for all $i \in W$; (stores positions in s)
while $\ell \leq |s|$ **do**
  $U := \{i \in W \mid \mathsf{pos}(i) < \ell\}$;
  $\mathsf{next} := \min(\{\mathsf{pos}(i) \mid i \in W \setminus U\} \cup \{|s| + 1\})$;
  $j := \max\{i \mid \ell - 1 \leq i \leq \mathsf{next} - 1, [\mathsf{inf}\, \pi_{W\setminus U}(s[\ell : i])]_I \preceq [t]_I\}$; (*)
  $\mathsf{inf} := \mathsf{inf}\, \pi_{W\setminus U}(s[\ell : j])$;
  $\mathsf{diff} := \mathsf{diff}\, \pi_U(s[\ell : j])\, s[j + 1]$; (let us set $s[|s| + 1] = \varepsilon$)
  for all $i \in W$ **do**
    $\mathsf{pos}(i) := \min\{\mathsf{pos}(i), \mathsf{pos}(s, j + 1, i)\}$ (let us set $\mathsf{pos}(s, |s| + 1, i) = |s| + 1$)
  endfor
  $\ell := j + 2$;
endwhile

Fig. 1. An algorithm for computing $[s]_I \sqcap [t]_I$ and $[s]_I \setminus [t]_I$

of the loop body in Figure 1 and let ℓ denote the value of the corresponding program variable at the beginning of the iteration. Assume in the following that $\mathsf{Diff}_\ell \subseteq \{1, \ldots, \ell - 1\}$ is the set of all positions from $\{1, \ldots, \ell - 1\}$, which belong to the difference $[s]_I \setminus [t]_I$, i.e., they do not belong to the common prefix $[s]_I \sqcap [t]_I$. Moreover, let $\mathsf{Inf}_\ell = \{1, \ldots, \ell-1\} \setminus \mathsf{Diff}_\ell$ be the set of all positions from $\{1, \ldots, \ell-1\}$, which belong to the trace prefix $[s]_I \sqcap [t]_I$. Thus, Inf_ℓ is downward-closed in D_s and $[s{\restriction}\mathsf{Inf}_\ell]_I \preceq [s]_I \sqcap [t]_I$. Note that the algorithm does neither store the set Diff_ℓ nor Inf_ℓ. This will be important later, when the input words s and t are represented by SLPs. If $\ell, \mathsf{inf}, \mathsf{diff}$, and $\mathsf{pos}(i)$ $(i \in W)$ denote the values of the corresponding program variables at the beginning of the iteration, then the algorithm will maintain the following two invariants:

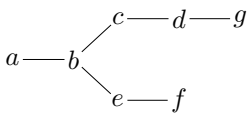
(I1) $\mathsf{inf} = s{\restriction}\mathsf{Inf}_\ell$, $\mathsf{diff} = s{\restriction}\mathsf{Diff}_\ell$,
(I2) $\mathsf{pos}(i) = \mathsf{pos}(s, \mathsf{Diff}_\ell, i)$ for all $i \in W$

In each iteration of the while-loop, we investigate the subword of s from position ℓ to the next position of the form $\mathsf{pos}(i)$, and we determine for each position from some initial segment of this interval, whether it belongs to $[s]_I \sqcap [t]_I$ or $[s]_I \setminus [t]_I$. More precisely, we search for the largest position $j \in \{\ell - 1, \ldots, \mathsf{next} - 1\}$ such that $[\mathsf{inf}\, \pi_{W\setminus U}(s[\ell : j])]_I$ is a prefix of $[t]_I$. Recall that $\mathsf{inf} = s{\restriction}\mathsf{Inf}_\ell$ is the already collected part of the common trace prefix. We update inf and diff by $\mathsf{inf} := \mathsf{inf}\, \pi_{W\setminus U}(s[\ell : j])$ and $\mathsf{diff} := \mathsf{diff}\, \pi_U(s[\ell : j])s[j + 1]$.

Before we prove that the algorithm indeed preserves the invariants (I1) and (I2), let us first consider a detailed example.

Example 44. To ease the reading we will consider the set $W = \{a, b, c, d, e, f, g\}$

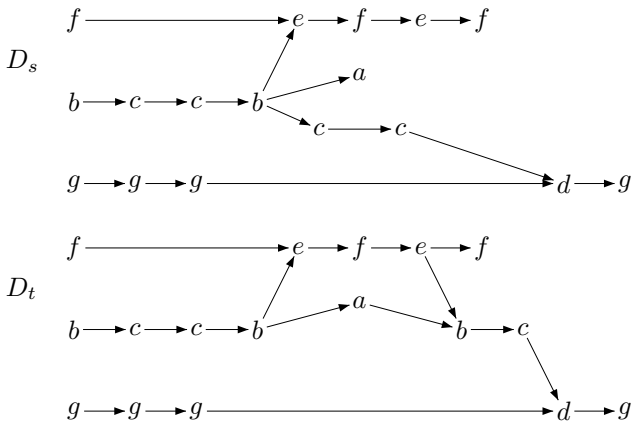
instead of $W = \{1, \ldots, 7\}$ together with the following dependence relation D:



Let $\Gamma_x = \{x\}$ for all $x \in W$. We consider the following strings:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$s =$	f	b	g	c	c	g	b	c	c	e	a	g	f	e	f	d	g
$t =$	b	c	g	c	f	g	b	e	a	g	g	f	e	d	f	b	g

The dependence graphs of $[s]_I$ and $[t]_I$ look as follows:



We want to determine $s \sqcap_p t$ and $s \setminus_p t$ using the algorithm from Figure 1. Initially, we set $\ell = 1$, $\mathsf{inf} = \mathsf{diff} = \varepsilon$, and $\mathsf{pos}(x) = |s| + 1 = 18$ for all $x \in W$. Since $\ell \leq |s|$ the while loop is executed.

First iteration: The algorithm first sets

$$U = \emptyset \quad \text{and} \quad \mathsf{next} = 18.$$

Hence, we have

$$\begin{array}{rcl} \mathsf{inf}\,\pi_{W\setminus U}(s[\ell : \mathsf{next} - 1]) & = & |f\ b\ g\ c\ c\ g\ b\ c\ c\ e\ a\ g\ f\ e\ f\ d\ g \\ & & \ \ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\,11\,12\,13\,14\,15\,16\,17\,. \end{array}$$

Here, we denote with '|' the position between inf and $\pi_{W\setminus U}(s[\ell : \mathsf{next} - 1])$. The algorithm determines the largest number j such that $0 \leq j \leq 17$ such that the trace $[\mathsf{inf}\,\pi_{W\setminus U}(s[\ell : j])]_I$ is a prefix of $[t]_I$. From the dependence graphs above it can be easily seen that $j = 7$. We have

$$\begin{array}{rcl} \mathsf{inf}\,\pi_{W\setminus U}(s[\ell : j]) & = & |f\ b\ g\ c\ c\ g\ b \\ \mathsf{diff}\,\pi_U(s[\ell : j])s[j+1] & = & \qquad\qquad\qquad\ \ c\,, \end{array}$$

which are the new values for inf and diff, respectively. Moreover, the pos-values are reset as follows:

$$\mathsf{pos}(a) = \mathsf{pos}(b) = 18,\ \mathsf{pos}(c) = 8,\ \mathsf{pos}(d) = 16,\ \mathsf{pos}(e) = \mathsf{pos}(f) = 18,\ \mathsf{pos}(g) = 17.$$

Finally, ℓ is set to 9. Since $\ell = 9 \leq |s|$ the while loop is repeated.

Second iteration: The algorithm first sets

$$U = \{c\} \quad \text{and} \quad \mathsf{next} = 16.$$

We have

$$\begin{array}{rcl} \mathsf{inf}\,\pi_{W\setminus\{c\}}(s[8:15]) & = & f\ b\ g\ c\ c\ g\ b\ \ |\ \ e\ a\ g\ f\ e\ f \\ & & 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\,11\,12\,13\,14\,15\,. \end{array}$$

Searching for the largest $\ell - 1 = 8 \leq j \leq 15 = \mathsf{next} - 1$ such that the trace $[\mathsf{inf}\,\pi_{W\setminus U}(s[\ell : j])]_I$ is a prefix of $[t]_I$ gives $j = 15$. We have

$$\begin{array}{rcl} \mathsf{inf}\,\pi_{W\setminus\{c\}}(s[9:15]) & = & f\ b\ g\ c\ c\ g\ b\ \ |\ \ e\ a\ g\ f\ e\ f \\ \mathsf{diff}\,\pi_{\{c\}}(s[9:15])s[16] & = & c\ c\ \ d\,, \end{array}$$

which are the new values for inf and diff. The pos-values do not change in the second iteration, i.e., we still have

$$\mathsf{pos}(a) = \mathsf{pos}(b) = 18,\ \mathsf{pos}(c) = 8,\ \mathsf{pos}(d) = 16,\ \mathsf{pos}(e) = \mathsf{pos}(f) = 18,\ \mathsf{pos}(g) = 17.$$

Finally, ℓ is set to 17. Since $\ell \leq |s|$ the while loop is repeated.

Third iteration: The algorithm first sets

$$U = \{c, d\} \quad \text{and} \quad \mathsf{next} = 17.$$

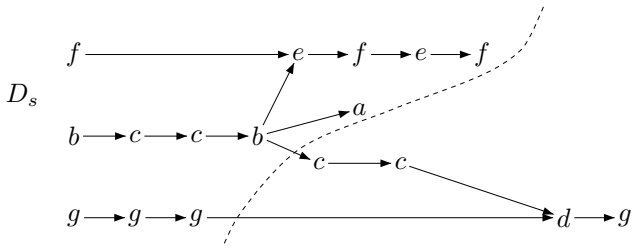
We have

$$\begin{array}{rcl} \mathsf{inf}\,\pi_{W\setminus U}(s[16:16]) & = & f\ b\ g\ c\ c\ g\ b\ \ \ e\ a\ g\ f\ e\ f| \\ & & 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\,11\,12\,13\,14\,15\,16\,. \end{array}$$

We find $j = 16$. Hence, we have

$$\begin{array}{rcl} \mathsf{inf}\,\pi_{W\setminus U}(s[17:16]) & = & f\ b\ g\ c\ c\ g\ b\ \ \ e\ a\ g\ f\ e\ f| \\ \mathsf{diff}\,\pi_{U}(s[17:16])s[17] & = & c\ c\ \ d\ g\,. \end{array}$$

Also in the third iteration, the pos-values do not change. Finally, ℓ is set to 18. Since $\ell > |s|$ the algorithm stops and produces $[\mathsf{inf}]_I = [fbgccgbeagfef]_I$ and $[\mathsf{diff}]_I = [ccdg]_I$. These traces are indeed $[s]_I \sqcap [t]_I$ and $[s]_I \setminus [t]_I$. This is visualized in the next picture with $[s]_I \sqcap [t]_I$ on the left side of the dotted line and $[s]_I \setminus [t]_I$ on the right side.



Let us now prove the correctness of the algorithm. We start with invariant (I1):

Lemma 45. *The algorithm from Figure 1 preserves invariant (I1).*

Proof. Let us take $\ell \in \{1, \ldots, |s|\}$ and assume that invariant (I1) and (I2) hold currently. Hence,

- $\mathsf{inf} = s{\restriction}\mathsf{Inf}_\ell$, $\mathsf{diff} = s{\restriction}\mathsf{Diff}_\ell$, and
- $\mathsf{pos}(i) = \mathsf{pos}(s, \mathsf{Diff}_\ell, i)$ for all $i \in W$.

We have to show that invariant (I1) holds after the next execution of the loop body as well. As in the algorithm, let:

$$U = \{i \in W \mid \mathsf{pos}(s, \mathsf{Diff}_\ell, i) < \ell\} \tag{3}$$

$$\mathsf{next} = \min(\{\mathsf{pos}(s, \mathsf{Diff}_\ell, i) \mid i \in W \setminus U\} \cup \{|s| + 1\}) \tag{4}$$

$$j = \max\{i \mid \ell - 1 \leq i \leq \mathsf{next} - 1, [\mathsf{inf}\, \pi_{W \setminus U}(s[\ell : i])]_I \preceq [t]_I\}. \tag{5}$$

We have to show the following:

- A position $p \in \{\ell, \ldots, j\}$ belongs to the common trace prefix $[s]_I \sqcap [t]_I$ if and only if $s[p] \in \Gamma_i$ for some $i \in W \setminus U$.
- If $j{+}1 \leq |s|$, then $j{+}1$ does not belong to the common trace prefix $[s]_I \sqcap [t]_I$.

For the first point, assume that $s[p] \in \Gamma_i$, where $\ell \leq p \leq j$ and $i \in U$. By definition of U in (3), we have $\mathsf{pos}(s, \mathsf{Diff}_\ell, i) < \ell \leq p$. Lemma 42 implies that there exists a path in D_s from some position in Diff_ℓ to position p. Since positions in Diff_ℓ do not belong to $[s]_I \sqcap [t]_I$, neither does p belong to $[s]_I \sqcap [t]_I$.

For the other direction, consider the set of positions

$$P = \{p \mid \ell \leq p \leq j, s[p] \in \Gamma_i \text{ for some } i \in W \setminus U\}.$$

We claim that $\mathsf{Inf}_\ell \cup P$ is a downward-closed subset of D_s. Since $[s{\restriction}(\mathsf{Inf}_\ell \cup P)]_I = [\mathsf{inf}\, \pi_{W \setminus U}(s[\ell : j])]_I \preceq [t]_I$ by (5), this implies that all positions from P indeed belong to $[s]_I \sqcap [t]_I$. That $\mathsf{Inf}_\ell \cup P$ is downward-closed in D_s follows from the following three points:

- Inf_ℓ is downward-closed.
- There does not exist a path from a node in Diff_ℓ to a node from P: Assume that such a path, ending in $p \in P$, would exist. Let $s[p] \in \Gamma_i$ with $i \in W \setminus U$. Lemma 42 implies $\mathsf{pos}(s, \mathsf{Diff}_\ell, i) \leq p$. Moreover, $i \in W \setminus U$ implies

$\mathsf{pos}(s, \mathsf{Diff}_\ell, i) \geq \ell$ by (3). Hence, $\ell \leq \mathsf{pos}(s, \mathsf{Diff}_\ell, i) \leq p \leq j < \mathsf{next}$, where the last inequality follows from (5). But this contradicts the definition of next in (4).

- There does not exist a path from a node in $\{\ell, \ldots, j\} \setminus P$ to a node of P: By Lemma 42, every node from $\{\ell, \ldots, j\} \setminus P$ can be reached via a path starting in Diff_ℓ. Hence, the existence of a path from $\{\ell, \ldots, j\} \setminus P$ to P contradicts the previous point.

It remains to be shown that position $j+1$ does not belong to the common trace prefix $[s]_I \sqcap [t]_I$ in case $j+1 \leq |s|$. We distinguish several cases: If $j = \mathsf{next}-1$, then $j+1 = \mathsf{pos}(s, \mathsf{Diff}_\ell, i)$ for some $i \in W$. Hence, there exists a path from Diff_ℓ to $j+1$ in D_s; therefore $j+1$ cannot belong to $[s]_I \sqcap [t]_I$. Now, assume that $j < \mathsf{next} - 1$. If $s[j+1] \in \Gamma_i$ for some $i \in U$, then Lemma 42 again yields the existence of a path from Diff_ℓ to $j+1$. Finally, let $s[j+1] \in \Gamma_i$ for some $i \in W \setminus U$. Maximality of j in (5) implies that the trace

$$[\mathsf{inf}\, \pi_{W \setminus U}(s[\ell, j+1])]_I = [\mathsf{inf}\, \pi_{W \setminus U}(s[\ell : j])]_I s[j+1]$$

is not a prefix of $[t]_I$. Since we already know that the trace $[\mathsf{inf}\, \pi_{W \setminus U}(s[\ell : j])]_I$ consists exactly of those positions from $\{1, \ldots, j\}$ that belong to the common trace prefix $[s]_I \sqcap [t]_I$, this implies that $j+1$ does not belong to $[s]_I \sqcap [t]_I$. □

Lemma 46. *The algorithm from Figure 1 preserves invariant (I2).*

Proof. We consider a specific iteration of the while loop and assume that (I1) and (I2) hold at the beginning of the loop, i.e.,

- $\mathsf{inf} = s{\restriction}\mathsf{Inf}_\ell$, $\mathsf{diff} = s{\restriction}\mathsf{Diff}_\ell$ and
- $\mathsf{pos}(i) = \mathsf{pos}(s, \mathsf{Diff}_\ell, i)$ for all $i \in W$.

We infer that (I2) holds after the execution of the loop. Let U, next, and j be defined by (3)–(5). Let $\ell' = j+2 > \ell$ be the new value of ℓ after the execution of the loop body and let $i \in W$. We have to show that $\mathsf{pos}(s, \mathsf{Diff}_{\ell'}, i)$ is the new value of $\mathsf{pos}(i)$ after the execution of the loop body. This means that we have to prove

$$\mathsf{pos}(s, \mathsf{Diff}_{\ell'}, i) = \min\{\mathsf{pos}(s,\ \mathsf{Diff}_\ell, i), \mathsf{pos}(s, \{j+1\}, i)\}. \tag{6}$$

From Lemma 45 and the way diff is updated in the loop body, we get

$$\mathsf{Diff}_{\ell'} = \mathsf{Diff}_\ell \cup \{p \mid \ell \leq p \leq j, \exists k \in U : s[p] \in \Gamma_k\} \cup \{j+1\} \tag{7}$$

(in case $j = |s|$, we omit $\{j+1\}$ from the right-hand side). Hence, $\mathsf{Diff}_\ell \cup \{j+1\} \subseteq \mathsf{Diff}_{\ell'}$, which implies

$$\mathsf{pos}(s, \mathsf{Diff}_{\ell'}, i) \leq \min\{\mathsf{pos}(s, \mathsf{Diff}_\ell, i), \mathsf{pos}(s, j+1, i)\}.$$

It remains to be shown that $\mathsf{pos}(s, \mathsf{Diff}_{\ell'}, i) \geq \min\{\mathsf{pos}(s, \mathsf{Diff}_\ell, i), \mathsf{pos}(s, j+1, i)\}$. The case that $\mathsf{pos}(s, \mathsf{Diff}_{\ell'}, i) = |s|+1$ is trivial. Hence, assume that $\mathsf{pos}(s, \mathsf{Diff}_{\ell'}, i) \leq |s|$ and consider a path in D_s from a position $p \in \mathsf{Diff}_{\ell'}$ to a position $q \leq |s|$ such that

$\ell := 1$;
$\alpha := \varepsilon$;
$\beta := \varepsilon$;
$\mathsf{pos}(i) := |\mathsf{val}(Y)| + 1$ for all $i \in W$;
while $\ell \leq |\mathsf{val}(Y)|$ **do**
  $U := \{i \in W \mid \mathsf{pos}(i) < \ell\}$;
  $\mathsf{next} := \min(\{\mathsf{pos}(i) \mid i \in W \setminus U\} \cup \{|\mathsf{val}(Y)| + 1\})$;
  $j := \max\{i \mid \ell - 1 \leq i \leq \mathsf{next} - 1, [\mathsf{val}(\alpha \circ \pi_{W\setminus U}(Y[\ell : i]))]_I \preceq [\mathsf{val}(Z)]_I\}$; (*)
  $\alpha := \alpha \circ \pi_{W\setminus U}(Y[\ell : j])$;
  $\beta := \beta \circ \pi_U(Y[\ell : j]) \circ Y[j + 1]$; (let us set here $\mathsf{val}(Y)[|\mathsf{val}(Y)| + 1] = \varepsilon$)
  for all $i \in W$ **do**
    $\mathsf{pos}(i) := \min\{\mathsf{pos}(i), \mathsf{pos}(\mathsf{val}(Y), j + 1, i)\}$ (**)
  endfor
  $\ell := j + 2$;
endwhile

Fig. 2. An algorithm for computing $[\mathsf{val}(Y)]_I \sqcap [\mathsf{val}(Z)]_I$ and $[\mathsf{val}(Y)]_I \setminus [\mathsf{val}(Z)]_I$

$s[q] \in \Gamma_i$. It suffices to show that there is a path from a position in $\mathsf{Diff}_\ell \cup \{j + 1\}$ to p (then, there exists a path from $\mathsf{Diff}_\ell \cup \{j + 1\}$ to q too). By (7), we have $p \in \mathsf{Diff}_\ell \cup \{p \mid \ell \leq p \leq j, \exists k \in U : s[p] \in \Gamma_k\} \cup \{j + 1\}$. The case $p \in \mathsf{Diff}_\ell \cup \{j + 1\}$ is trivial. Hence, assume that $\ell \leq p \leq j$ and $s[p] \in \Gamma_k$ for some $k \in U$. From (3), we get $\mathsf{pos}(s, \mathsf{Diff}_\ell, k) < \ell \leq p$. Lemma 42 implies that there exists a path from Diff_ℓ to p. □

Lemma 47. *The number of iterations of the while-loop in Figure 1 is bounded by* $|W| + 1 = n + 1 = \mathcal{O}(1)$.

Proof. We claim that in each execution of the loop body except for the last one, the set $U = \{i \in W \mid \mathsf{pos}(i) < \ell\}$ strictly grows, which proves the lemma. Let us consider an execution of the loop body. Note that the positions $\mathsf{pos}(i)$ cannot increase. There are two cases to distinguish. If $j < \mathsf{next} - 1$, then the symbol $s[j + 1]$ must belong to some alphabet Γ_i with $i \in W \setminus U$ due to the maximality of j in line (*) of the algorithm. Clearly, $\mathsf{pos}(s, \{j + 1\}, i) = j + 1$, hence $\mathsf{pos}(i)$ will be set to a value $\leq j + 1$ in the loop body. Since the new value ℓ will be $j + 2$, the new set U will also contain i, i.e., it strictly grows. If $j = \mathsf{next} - 1 < |s|$, then again, since $j + 1 = \mathsf{next} = \mathsf{pos}(i)$ for some $i \in W \setminus U$ and the new value ℓ will be $j + 2$, the set U strictly grows. Finally, if $j = \mathsf{next} - 1 = |s|$, then ℓ will be set to $|s| + 2$ and the algorithm terminates. □

The algorithm from Figure 1 for computing $[s]_I \setminus [t]_I$ and $[s]_I \sqcap [t]_I$ leads to a polynomial time algorithm, which computes CCP-expressions for (we write val for $\mathsf{val}_\mathbb{B}$ in the following) $[\mathsf{val}(Y)]_I \sqcap [\mathsf{val}(Z)]_I$ and $[\mathsf{val}(Y)]_I \setminus [\mathsf{val}(Z)]_I$, see Figure 2 (where the

concatenation operation in CCP-expressions is denoted by $\circ$ for better readability). The idea is to consider the statements for updating inf and diff in Figure 1 as statements for computing CCP-expressions α and β with $[\mathsf{val}(\alpha)]_I = [\mathsf{val}(Y)]_I \sqcap [\mathsf{val}(Z)]_I$ and $[\mathsf{val}(\beta)]_I = [\mathsf{val}(Y)]_I \setminus [\mathsf{val}(Z)]_I$. So, (2) and (3) from Proposition 40 is satisfied. Moreover, property (1) follows directly from the construction of α and β. For the size estimate in (4), note that by Lemma 47, α and β are concatenations of $\mathcal{O}(1)$ many expressions of the form $\pi_K(Y[p_1, p_2])$. Moreover, each of the positions p_1 and p_2 is bounded by $|\mathsf{val}(Y)| \leq |\mathsf{val}(\mathbb{B})|$ and hence needs only $\mathcal{O}(\log_2(|\mathsf{val}(\mathbb{B})|))$ many bits.

It remains to be argued that the algorithm in Figure 2 is indeed a polynomial time algorithm. By Lemma 47, the number of iterations of the loop body is bounded by $|W| + 1$. Hence, it suffices to show that a single iteration only needs polynomial time. The condition $[\mathsf{val}(\alpha \circ \pi_{W\setminus U}(Y[\ell : j]))]_I \preceq [\mathsf{val}(Z)]_I$ in line (*) of Figure 2 can be checked in polynomial time by Lemma 19; note that by Lemma 3 we can compute in polynomial time an SLP for $\mathsf{val}(\alpha \circ \pi_{W\setminus U}(Y[\ell : j]))$. Hence, the number j in line (*) can be computed in polynomial time via binary search. Finally, the position $\mathsf{pos}(\mathsf{val}(Y), j+1, i)$ in line (**) of Figure 2 can be computed in polynomial time by Lemma 43. This finishes the proof of Proposition 40.

6. More algorithms for compressed traces

Before we proceed to the proof of our second main result Theorem 27, we first prove some further results concerning traces, which are represented by SLPs. This results are more or less direct consequences of Proposition 40.

As in the previous sections we fix the *finite* independence alphabet (W, E) with $W = \{1, \ldots, n\}$. An immediate corollary of Lemma 15, Proposition 40, and Lemma 3 is:

Corollary 48. *Let (W, E) be a fixed independence alphabet with $W = \{1, \ldots, n\}$. For given pairwise disjoint finite alphabets $\Gamma_1, \ldots, \Gamma_n$ and SLPs $\mathbb{A}$ and $\mathbb{B}$ over the alphabet $\Gamma = \bigcup_{i=1}^n \Gamma_i$, we can check in polynomial time, whether the trace supremum $[\mathsf{val}(\mathbb{A})]_{E[\Gamma_1,\ldots,\Gamma_n]} \sqcup_p [\mathsf{val}(\mathbb{B})]_{E[\Gamma_1,\ldots,\Gamma_n]}$ exists, and in case it exists, we can compute in polynomial time an SLP $\mathbb{S}$ such that*

$$[\mathsf{val}(\mathbb{S})]_{E[\Gamma_1,\ldots,\Gamma_n]} = [\mathsf{val}(\mathbb{A})]_{E[\Gamma_1,\ldots,\Gamma_n]} \sqcup_p [\mathsf{val}(\mathbb{B})]_{E[\Gamma_1,\ldots,\Gamma_n]}.$$

Lemma 17 and Corollary 48 imply the following corollary.

Corollary 49. *Let (W, E) be a fixed independence alphabet with $W = \{1, \ldots, n\}$ and $r \in \mathbb{N}$ a constant. For given pairwise disjoint finite alphabets $\Gamma_1, \ldots, \Gamma_n$ and SLPs $\mathbb{A}_1, \ldots, \mathbb{A}_r$ over the alphabet $\Gamma = \bigcup_{i=1}^n \Gamma_i$, we can check in polynomial time, whether the trace supremum $[\mathsf{val}(\mathbb{A}_1)]_{E[\Gamma_1,\ldots,\Gamma_n]} \sqcup_p \cdots \sqcup_p [\mathsf{val}(\mathbb{A}_r)]_{E[\Gamma_1,\ldots,\Gamma_n]}$ exists, and in case it exists, we can compute in polynomial time an SLP $\mathbb{S}$ such that*

$$[\mathsf{val}(\mathbb{S})]_{E[\Gamma_1,\ldots,\Gamma_n]} = [\mathsf{val}(\mathbb{A}_1)]_{E[\Gamma_1,\ldots,\Gamma_n]} \sqcup_p \cdots \sqcup_p [\mathsf{val}(\mathbb{A}_r)]_{E[\Gamma_1,\ldots,\Gamma_n]}.$$

Clearly the Corollaries 48 and 49 also hold for the suffix supremum. It is important that we fix the number r of SLPs in Corollary 49: Each application of Lemma 48 may increase the size of the SLP polynomially. Hence, a non-fixed number of applications might lead to an exponential blow-up.

7. Double cones

For this section let us fix as usual the finite independence alphabet (W, E) with $W = \{1, \dots, n\}$. Moreover, as in Section 2.6, we fix finitely generated groups $\mathbb{G}_i$ for $i \in \{1, \dots n\}$. Let $\mathbb{G}$ denote the graph product $\mathbb{G}(W, E, (\mathbb{G}_i)_{i \in W})$ for the rest of this section. Moreover, let A_i, A, I, and R have the same meaning as in Section 2.6. All identities in this section hold in the trace monoid $\mathbb{M}(A, I)$, unless we add "in $\mathbb{G}$", which of course means that the identity holds in the graph product $\mathbb{G}$.

In this section, we will prove several results for the trace monoid $\mathbb{M}(A, I)$ which will be needed in the next section for deciding $\mathsf{RSCCP}(\mathbb{G}, B)$ for some finite $B \subseteq A$. Let

$$Z = \{i \in W \mid (i, j) \in E \text{ for all } j \neq i\}.$$

For $i \in W$ let

$$\mathsf{Star}(i) = A_i \cup \bigcup_{j \in E(i)} A_j.$$

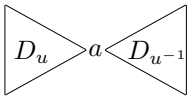
Note that

$$\bigcap_{i \in W} \mathsf{Star}(i) = \bigcup_{i \in Z} A_i. \tag{8}$$

The following statement is straightforward to prove by considering the dependence graph of uau^{-1}.

Lemma 50. *Let $i \in W$, $a \in A_i$, and $u \in \mathsf{IRR}(R)$. The trace uau^{-1} is R-irreducible if and and only if* $\max(u) \cap \mathsf{Star}(i) = \emptyset$.

The dependence graph of an R-irreducible trace uau^{-1} has the following shape:



We call an R-irreducible trace of the form uau^{-1} with $a \in A$ a *double cone*. By the following lemma, each double cone has a unique factorization of the form $u_1 b u_2$ with $|u_1| = |u_2|$.

Lemma 51. *Let $uau^{-1} = u_1 b u_2 \in \mathsf{IRR}(R)$ with $a, b \in A$ and $|u_1| = |u_2|$. Then $a = b$, $u_1 = u$ and $u_2 = u^{-1}$.*

Proof. Let $uau^{-1} = u_1 b u_2 \in \mathsf{IRR}(R)$, where $a, b \in A$ and $|u_1| = |u_2|$. We have $\max(ua) = \{a\}$ and $(a, c) \in D$ for all $c \in \min(u^{-1})$. Moreover $|u_1| = |u_2| = |u|$. By

Levi's Lemma 11, there exist traces x, y_1, y_2 and z with $u_1 b = xy_1$, $u_2 = y_2 z$, $ua = xy_2$, $u^{-1} = y_1 z$ and $y_1 I y_2$. Assume that $y_2 \neq \varepsilon$. Since $\max(y_2) \subseteq \max(ua) = \{a\}$ we get $\max(y_2) = \{a\}$. Since $(a, c) \in D$ for all $c \in \min(y_1) \subseteq \min(u^{-1})$ and $y_1 I y_2$ it follows $y_1 = \varepsilon$. But then $|u| = |u^{-1}| = |z| < |y_2 z| = |u_2|$ leads to a contradiction. Hence, we must have $y_2 = \varepsilon$. Thus $|u| = |u^{-1}| = |y_1 z| = |y_1| + |z| = |y_1| + |u_2| = |y_1| + |u|$ implies $y_1 = \varepsilon$. Therefore we get $ua = u_1 b$ and $u^{-1} = u_2$. Finally, since $\max(ua) = \{a\}$ we must have $a = b$ and $u = u_1$. □

The next lemma gives us a necessary condition for an $\mathsf{RSCCP}(\mathbb{G}, B)$-instance to be solvable. In the lemma we restrict to those (decompressed) equations $w = xax^{-1}$ of an $\mathsf{RSCCP}(\mathbb{G}, B)$-instance where a is from a fixed factor $\mathbb{G}_i$ of our graph product $\mathbb{G}$. Assume that these equations are $w_j = xa_j x^{-1}$ $(1 \leq j \leq m)$ where $w_j \in \mathbb{G}$ (it is given compressed) and $a_j \in A_i$, and assume there exists a solution x in $\mathbb{G}$. Then the lemma tells us that there exists a single trace v and a single element $c \in \mathbb{G}_i$ such that each left-hand side w_j can be represented by an R-irreducible trace of the form $vb_j v^{-1}$ with $b_j \in A_i$ (i.e., it is a double cone). Moreover, for all $(1 \leq j \leq m)$ we have $a_j = cb_j c^{-1}$ in the group $\mathbb{G}_i$. The latter means that a certain $\mathsf{RSCCP}(\mathbb{G}_i, \{a_1, \ldots, a_m\})$ is solvable (in our later application of the lemma, we will compute SLPs for the group elements $b_j \in \mathbb{G}_i$). In this way, we will reduce our $\mathsf{RSCCP}(\mathbb{G}, B)$-instance to RSCCP-instances for the factor groups $\mathbb{G}_i$.

Lemma 52. *Let $i \in W$, $a_1, \ldots, a_m \in A_i$, $w_1, \ldots, w_m \in \mathsf{IRR}(R)$, and $x \in \mathbb{M}(A, I)$ such that $w_j = xa_j x^{-1}$ in $\mathbb{G}$ for all $1 \leq j \leq m$. Then there exist $v \in \mathsf{IRR}(R)$, $b_1, \ldots, b_m \in A_i$, $c \in \mathbb{G}_i$ such that for all $1 \leq j \leq m$:*

- *$w_j = vb_j v^{-1}$ in $\mathbb{M}(A, I)$ and*
- *$a_j = cb_j c^{-1}$ in the group $\mathbb{G}_i$.*

Proof. We prove the lemma by induction on the length of the trace x. W.l.o.g. we can assume that $x \in \mathsf{IRR}(R)$, since if $w_j = xa_j x^{-1}$ holds in $\mathbb{G}$ then also $w_j = \mathsf{NF}_R(x) a_j \mathsf{NF}_R(x)^{-1}$ in $\mathbb{G}$.

If $|x| = 0$ then $w_j = a_j$ for all $1 \leq j \leq m$. Hence, we can set $v = \varepsilon$, $b_j = a_j$, and $c = 1_{\mathbb{G}_i}$. For the induction step we distinguish three cases. First assume that $\max(x) \cap A_i \neq \emptyset$. Let $x = yd$ in $\mathbb{M}(A, I)$ with $d \in A_i$. In $\mathbb{G}$ we have

$$w_j = xa_j x^{-1} = yda_j d^{-1} y^{-1} = ya'_j y^{-1}$$

with $a'_j = da_j d^{-1} \in A_i$ for $1 \leq j \leq m$. Since $|y| < |x|$ it follows by induction that there are $v \in \mathsf{IRR}(R)$, $b_1, \ldots, b_m \in A_i$, $c \in \mathbb{G}_i$ such that for all $1 \leq j \leq m$:

- $w_j = vb_j v^{-1}$ in $\mathbb{M}(A, I)$ and
- $a'_j = cb_j c^{-1}$ in the group $\mathbb{G}_i$.

The last identity implies $a_j = (d^{-1}c) b_j (c^{-1}d)$.

Next, assume that $\max(x) \cap A_j \neq \emptyset$ for some j with $(i,j) \in E$. Let $x = yd$ in $\mathbb{M}(A,I)$ with $d \in A_j$ and $(i,j) \in E$. We get

$$w_j = xa_jx^{-1} = yca_jc^{-1}y^{-1} = ya_jy^{-1}$$

in $\mathbb{G}$. Again, we can directly apply the induction hypothesis.

Finally assume that $\max(x) \cap \mathsf{Star}(i) = \emptyset$. Then, by Lemma 50 $xa_jx^{-1} \in \mathsf{IRR}(R)$ (since $x \in \mathsf{IRR}(R)$) for all $1 \leq j \leq m$. Hence, $w_j = xa_jx^{-1}$ in $\mathbb{M}(A,I)$. We can set $v = x$, $b_j = a_j$, and $c = 1_{\mathbb{G}_i}$. □

For $a \in A$ we denote in the sequel with $A(a)$ the unique set A_i ($i \in W$) such that $a \in A_i$. Similarly, we denote with $\mathbb{G}(a)$ the unique group $\mathbb{G}_i$ such that $a \in A_i$.

Note that the two premises (a) and (b) in the following lemma are precisely the conclusions (for all $i \in W$) in Lemma 52. The lemma gives us a criterion (namely (2) below) for solvability of an $\mathsf{RSCCP}(\mathbb{G}, B)$-instance (which is (1) below). Using the results from Section 6, we are able to check this criterion in polynomial time (using oracle access to the problems for the factor groups $\mathbb{G}_i$ listed in Theorem 27).

Lemma 53. *Let $B \subseteq A$ be a finite set with $B_i = B \cap A_i \neq \emptyset$ for $i \in W$. Let $w_a \in \mathsf{IRR}(R)$ ($a \in B$), $v_i \in \mathsf{IRR}(R)$ ($i \in W$), $b_a \in A(a)$ ($a \in B$), and $c_i \in \mathbb{G}_i$ ($i \in W$) such that for all $i \in W$ and all $a \in B_i$ we have:*

(a) $w_a = v_ib_av_i^{-1}$ in $\mathbb{M}(A,I)$ and
(b) $a = c_ib_ac_i^{-1}$ in $\mathbb{G}_i$.

If $s = \bigsqcup_{i \in W} v_i$ exists, then $\mathsf{alph}(s) \cap A_i = \emptyset$ for all $i \in Z$. Moreover, the following two conditions are equivalent:

(1) There exists $x \in \mathbb{M}(A,I)$ with $w_a = xax^{-1}$ in $\mathbb{G}$ for all $a \in B$.
(2) The trace supremum $s = \bigsqcup_{i \in W} v_i$ exists and $w_a = sas^{-1}$ holds in $\mathbb{G}$ for all $a \in \bigcup_{i \in W \setminus Z} B_i$.

Proof. We first show that $\mathsf{alph}(s) \cap A_j = \emptyset$ for all $j \in Z$ in case $s = \bigsqcup_{i \in W} v_i$ exists. For this it suffices to show that every v_i ($i \in W$) does not contain symbols from $\bigcup_{j \in Z} A_j$. Recall that for $a \in B_i$ the trace $w_a = v_ib_av_i^{-1}$ is R-irreducible. But if v_i would contain a symbol from $\bigcup_{j \in Z} A_j$, then $v_ib_av_i^{-1}$ would not be R-irreducible.

Let us now prove the equivalence of (1) and (2). For the direction (2) ⇒ (1) we set

$$x = s \prod_{j \in Z} c_j^{-1}$$

Then, for all $a \in \bigcup_{i \in W \setminus Z} B_i$ we get

$$w_a = sas^{-1} = xax^{-1},$$

since $\prod_{j \in Z} c_j^{-1}$ commutes with a. On the other hand, if $a \in B_i$ for some $i \in Z$, then we must have $v_i = \varepsilon$ (otherwise $w_a = v_ib_av_i^{-1}$ would not belong to $\mathsf{IRR}(R)$).

Hence, in $\mathbb{G}$ we get

$$w_a = b_a = c_i^{-1} a c_i = (s \prod_{j \in Z} c_j^{-1}) a (s \prod_{j \in Z} c_j^{-1})^{-1} = xax^{-1}.$$

Here, it is important to note that that s commutes with $\prod_{j \in Z} c_j^{-1}$ and a (since s does not contain symbols from $\bigcup_{j \in Z} A_j$) and also $\prod_{j \in Z \setminus \{i\}} c_j^{-1}$ commutes with $a \in B_i$.

For the direction (1) $\Rightarrow$ (2) let $w_a = xax^{-1}$ in $\mathbb{G}$ for all $a \in B$, where we assume w.l.o.g. that $x \in \mathsf{IRR}(R)$. For all $i \in W$, choose $d_i \in A_i \cup \{\varepsilon\}$ and $x_i \in \mathsf{IRR}(R)$ such that $x = x_i d_i$ and $\max(x_i) \cap A_i = \emptyset$. Furthermore let $x_i = t_i u_i$ with $u_i I A_i$ such that $|u_i|$ is maximal. By Lemma 12, this factorization is unique. Then, for all $a \in B_i \neq \emptyset$,

$$v_i b_a v_i^{-1} = w_a = xax^{-1} = t_i u_i d_i a d_i^{-1} u_i^{-1} t_i^{-1} = t_i c_a t_i^{-1}$$

in $\mathbb{G}$ where we set $c_a = d_i a d_i^{-1} \in A_i$. Moreover, the definition of t_i implies that $\max(t_i) \cap \mathsf{Star}(i) = \emptyset$. By Lemma 50, $t_i c_a t_i^{-1} \in \mathsf{IRR}(R)$. Hence, $t_i c_a t_i^{-1} = v_i b_a v_i^{-1}$. Lemma 51 implies $t_i = v_i$ and $c_a = b_a$. Hence, we have $x = t_i u_i d_i = v_i u_i d_i$ which implies $v_i \preceq_p x$ for all $i \in W$. Therefore $s = \bigsqcup_{i \in W} v_i$ exists and $s \preceq_p x$. But then $x = sy$ for some R-irreducible $y \in \mathbb{M}(A, I)$. Moreover, we can find for all $i \in W$ some $r_i \in \mathbb{M}(A, I)$ such that $s = v_i r_i$. We have $x = v_i r_i y$ and set $z_i = r_i y$. Then, for all $a \in B_i$, we get $v_i b_a v_i^{-1} = w_a = xax^{-1} = v_i z_i a z_i^{-1} v_i^{-1}$ in $\mathbb{G}$. We can cancel v_i and v_i^{-1} to infer $b_a = z_i a z_i^{-1}$ in $\mathbb{G}$. Hence, we must have $z_i a z_i^{-1} \xrightarrow{*}_R b_a$. It follows that $\mathsf{alph}(z_i) \subseteq \mathsf{Star}(i)$ for all $i \in W$. Since y is a suffix of z_i for all $i \in W$ it follows that $\mathsf{alph}(y) \subseteq \mathsf{Star}(i)$ for all $i \in W$ and therefore $\mathsf{alph}(y) \subseteq \bigcup_{j \in Z} A_j$ by (8). Hence, y commutes with all $a \in \bigcup_{i \in W \setminus Z} B_i$. Thus, for all $a \in \bigcup_{i \in W \setminus Z} B_i$ we get

$$w_a = xax^{-1} = syay^{-1}s^{-1} = sas^{-1}$$

in $\mathbb{G}$. This concludes the proof of the lemma. □

8. Restricted simultaneous compressed conjugacy

Based on our results on double cones from the previous section, we will prove Theorem 27 in this section. We use the notations for (W, E), $\mathbb{G}_i$ $(i \in W)$, $\mathbb{G}$, R, A, A_i $(i \in W)$ and $\mathbb{M}(A, I)$ from the previous section. Let furthermore Σ_i be a finite generating set for $\mathbb{G}_i$ for $i \in \{1, \ldots, n\}$. W.l.o.g. we can assume that $\Sigma_i \cap \Sigma_j = \emptyset$ for $i \neq j$. We define $\Sigma = \bigcup_{i=1}^n \Sigma_i$; it is a finite generating set of the graph product $\mathbb{G}$.

Let $\emptyset \neq B_i \subseteq \mathbb{G}_i$ be finite for $i \in W$ and $B = \bigcup_{i \in W} B_i$. Let us fix an instance $(\mathbb{A}_a)_{a \in B}$ of the problem $\mathsf{RSCCP}(\mathbb{G}, B)$. We can assume w.l.o.g. that for all $i \in W$ and all $a \in B_i$ we have $a \neq 1_{\mathbb{G}_i}$ (and hence $B_i \subseteq A_i$), since otherwise we only have to test in polynomial time (using oracle access to $\mathsf{CWP}(\mathbb{G}_i)$) whether $\mathsf{val}(\mathbb{A}_a) = 1_{\mathbb{G}_i}$. If this is true, then we can remove a and $\mathbb{A}_a$ from the $\mathsf{RSCCP}(\mathbb{G}, B)$ instance.

To the given SLPs $\mathbb{A}_a$ $(a \in B)$ we first apply Proposition 39 to construct reduced well-formed 2-level CCP-systems $\mathbb{B}_a$ $(a \in B)$ with $\mathsf{val}(\mathbb{A}_a) = \mathsf{val}(\mathbb{B}_a)$ in $\mathbb{G}$ in

polynomial time using oracle access to the decision problems $\mathsf{CWP}(\mathbb{G}_i)$ for $i \in W$. As explained in Section 4, we can interpret lower level nonterminals from a 2-level CCP-system $\mathbb{B}_a$ as elements from A and hence $\mathsf{uval}_{\mathbb{B}_a}$ maps every upper level nonterminal of $\mathbb{B}_a$ to a word over A, which can be interpreted as an element of $\mathbb{M}(A, I)$ or $\mathbb{G}$. We need the following lemma:

Lemma 54. *For a given reduced and well-formed 2-level CCP-system $\mathbb{A}$ over $\Sigma^{\pm 1}$ we can check in polynomial time, whether the trace $[\mathsf{uval}(\mathbb{A})]_I$ is a double cone. In case $[\mathsf{uval}(\mathbb{A})]_I$ is a double cone, we can compute in polynomial time the following data:*

- *some $i \in W$,*
- *a reduced and well-formed 2-level CCP-system $\mathbb{V}_a$ over $\Sigma^{\pm 1}$ and*
- *an SLP $\mathbb{C}$ over the alphabet $\Sigma_i^{\pm 1}$ such that $\mathsf{val}(\mathbb{C})$ represents the element $a \in A_i$ and $[\mathsf{uval}(\mathbb{A})]_I = [\mathsf{uval}(\mathbb{V}_a)]_I a [\mathsf{uval}(\mathbb{V}_a)]_I^{-1}$ in $\mathbb{M}(A, I)$.*

Proof. First we check whether $|\mathsf{uval}(\mathbb{A})|$ is odd. If not, then $\mathsf{uval}(\mathbb{A})$ cannot be a double cone. Assume that $|\mathsf{uval}(\mathbb{A})| = 2k+1$ for some $k \geq 0$ and let $\mathsf{uval}(\mathbb{A}) = u_1 a u_2$ with $|u_1| = |u_2| = k$. By Lemma 51, $[\mathsf{uval}(\mathbb{A})]_I$ is a double cone if and only if $[u_1]_I = [u_2]_I^{-1}$ (for this, it is important that $\mathbb{A}$ is reduced). We can easily compute reduced and well-formed 2-level CCP-systems $\mathbb{V}_a$ and $\mathbb{V}'_a$ with $\mathsf{uval}(\mathbb{V}_a) = u_1$ and $\mathsf{uval}(\mathbb{V}'_a) = u_2^{-1}$. By Lemma 3 and 20, we can check in polynomial time whether $[\mathsf{uval}(\mathbb{V}_a)]_I = [\mathsf{uval}(\mathbb{V}'_a)]_I$. Moreover, we can compute in polynomial time $i \in W$ such that $a \in A_i$ and an SLP $\mathbb{C}$, which generates the group element a (this SLP is part of $\mathsf{lo}(\mathbb{A})$). This concludes the proof. □

Now we can present an algorithm that solves $\mathsf{RSCCP}(\mathbb{G}, B)$ in polynomial time with oracle access to the problems $\mathsf{CWP}(\mathbb{G}_i)$ and $\mathsf{RSCCP}(\mathbb{G}_i, B_i)$ for $i \in W$:

Proof of Theorem 27. Let $\mathbb{A}_a$ ($a \in B$) be the input SLPs. We have to check whether there exists x such that $\mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}$ for all $a \in B$. By Proposition 39 we can assume that $\mathbb{A}_a$ is a reduced well-formed 2-level CCP-system for all $a \in B$. Note that we need oracle access to the problems $\mathsf{CWP}(\mathbb{G}_i)$ in order to apply Proposition 39.

In a first step we check, whether there exist $v_i \in \mathbb{M}(A, I)$, $c_i \in \mathbb{G}_i$ ($i \in W$) and $b_a \in A(a)$ ($a \in B$) such that for all $i \in W$ and all $a \in B_i$:

- $[\mathsf{uval}(\mathbb{A}_a)]_I = v_i b_a v_i^{-1}$ (in $\mathbb{M}(A, I)$) and
- $a = c_i b_a c_i^{-1}$ in the group $\mathbb{G}_i$.

Two points are important here:

- If these elements v_i, c_i, b_a do not exist, then by Lemma 52 there cannot exist x such that $\mathsf{val}(\mathbb{A}_a) = xax^{-1}$ in $\mathbb{G}$ for all $a \in B$.
- One can check in polynomial time whether v_i, c_i, b_a with the above properties exist using oracle access to the problems $\mathsf{RSCCP}(\mathbb{G}_i, B_i)$ ($i \in W$): Using

Lemma 54, we check in polynomial time, whether every trace $[\mathsf{uval}(\mathbb{A}_a)]_I$ is a double cone (we can reject, if this does not hold) and compute reduced and well-formed CCP-systems $\mathbb{V}_a$ and SLPs for elements $b_a \in A$ with $[\mathsf{uval}(\mathbb{A}_a)]_I = [\mathsf{uval}(\mathbb{V}_a)]_I b_a [\mathsf{uval}(\mathbb{V}_a)]_I^{-1}$. Next, we check whether $b_a \in A(a)$ (this is easy; just look at the terminal alphabet of the SLP for b_a) and whether $[\mathsf{uval}(\mathbb{V}_a)]_I = [\mathsf{uval}(\mathbb{V}_b)]_I$ if $a, b \in B$ belong to the same set B_i. The latter is possible in polynomial time by Lemma 3 and 20. Finally, we have to check whether for every $i \in W$ there exists $c_i \in \mathbb{G}_i$ such that for all $a \in B_i$, $a = c_i b_a c_i^{-1}$ holds in the group $\mathbb{G}_i$. But this is an instance of the problem $\mathsf{RSCCP}(\mathbb{G}_i, B_i)$ (recall that b_a is given by an SLP over the alphabet $\Sigma_i^{\pm 1}$).

Let us now assume that the elements v_i, c_i, b_a with the above properties exist and that we have computed the reduced and well-formed 2-level CCP-systems $\mathbb{V}_i$ ($i \in W$) with $[\mathsf{uval}(\mathbb{V}_i)]_I = v_i$. Then, by Lemma 53 it suffices to check whether the supremum $s = \bigsqcup_{i \in W} v_i$ exists and whether

$$[\mathsf{uval}(\mathbb{A}_a)]_I = sas^{-1} \tag{9}$$

holds in $\mathbb{G}$ for all $a \in \bigcup_{i \in W \setminus Z} B_i$.

First we check in polynomial time using Corollary 49 whether the trace supremum $s = \bigsqcup_{i \in W} [\mathsf{uval}(\mathbb{V}_i)]_I$ exists (recall that $|W| = n$ is a constant in our consideration). Assume that s exist. Corollary 49 allows us to compute in polynomial time a well-formed 2-level CCP-systems $\mathbb{S}$ such that $[\mathsf{uval}(\mathbb{S})]_I = s$. Hence (9) becomes an instance of $\mathsf{CWP}(\mathbb{G})$, which by Theorem 26 can be solved in polynomial time with oracle access to the problems $\mathsf{CWP}(\mathbb{G}_i)$ ($i \in W$). This concludes the proof of Theorem 27.

9. Computing the core of a compressed trace

The next three sections are concerned with the proof of Theorem 31, stating that the compressed conjugacy problem of a graph group can be solved in polynomial time. In this section, we introduce the concept of the core of a trace [46] and show that the core of a compressed trace can be computed in polynomial time.

We will fix a finite independence alphabet (Σ, I) and the corresponding graph group $\mathbb{G}(\Sigma, I)$ for the rest of this section. With R we denote the trace rewriting system from Section 2.5.

The following results are direct corollaries of Lemma 3, Proposition 39, and Proposition 40 (which can be applies since (Σ, I) is a fixed independence alphabet):

Corollary 55. *For a given SLP $\mathbb{A}$ over the alphabet $\Sigma^{\pm 1}$ we can compute in polynomial time an SLP $\mathbb{B}$ over the alphabet $\Sigma^{\pm 1}$ with $[\mathsf{val}(\mathbb{B})]_I = \mathsf{NF}_R([\mathsf{val}(\mathbb{A})]_I)$.*

Corollary 56. *For given SLPs $\mathbb{A}$ and $\mathbb{B}$ over $\Sigma^{\pm 1}$, we can compute in polynomial time SLPs $\mathbb{P}$ and $\mathbb{D}$ such that:*

- $[\mathsf{val}(\mathbb{P})]_I = [\mathsf{val}(\mathbb{A})]_I \sqcap_p [\mathsf{val}(\mathbb{B})]_I$ *and*
- $[\mathsf{val}(\mathbb{D})]_I = [\mathsf{val}(\mathbb{A})]_I \setminus_p [\mathsf{val}(\mathbb{B})]_I$.

Definition 57. *A trace v is called* cyclically R-irreducible *if $v \in \mathsf{IRR}(R)$ and $\min(v) \cap \min(v^{-1}) = \emptyset$. If for a trace w we have $\mathsf{NF}_R(w) = uvu^{-1}$ in $\mathbb{M}(\Sigma^{\pm 1}, I)$ for traces u, v with v cyclically R-irreducible, then we call v the* core *of w, $\mathsf{core}(w)$ for short.*

The trace v in the last definition is uniquely defined [46]. Moreover, note that a trace t is a double cone if and only if $t \in \mathsf{IRR}(R)$ and $\mathsf{core}(t)$ has length 1.

In this section, we will present a polynomial time algorithm for computing an SLP that represents $\mathsf{core}([\mathsf{val}(\mathbb{A})]_I)$ for a given SLP $\mathbb{A}$. For this, we need the following lemmas.

Lemma 58. *Let $u, w \in \mathbb{M}(\Sigma^{\pm 1}, I)$. If $u \preceq_p w$, $u^{-1} \preceq_p w$ and $w \in \mathsf{IRR}(R)$, then $u = \varepsilon$.*

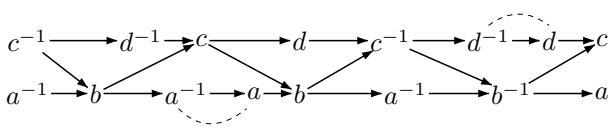
Proof. Suppose for contradiction that

$$T = \{w \in \mathsf{IRR}(R) \mid \exists u \in \mathbb{M}(\Sigma^{\pm 1}, I) \setminus \{\varepsilon\} : u, u^{-1} \preceq_p w\} \neq \emptyset.$$

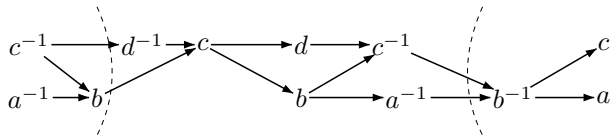
Let $w \in T$ with $|w|$ minimal and $u \in \mathbb{M}(\Sigma^{\pm 1}, I)$ such that $u \neq \varepsilon$ and $u, u^{-1} \preceq_p w$. If $|u| = 1$ then $u = a$ for some $a \in \Sigma^{\pm 1}$ and hence $a, a^{-1} \preceq_p w$, a contradiction since aDa^{-1}. If $|u| = 2$ then $u = a_1 a_2$ for some $a_1, a_2 \in \Sigma^{\pm 1}$. Since w, and therefore u is R-irreducible, we have $a_1 \neq a_2^{-1}$. Since $a_1 \in \min(w)$ and $a_2^{-1} \in \min(w)$ we have $a_1 I a_2^{-1}$, i.e., $a_1 I a_2$. Hence, also $a_2 \in \min(w)$, which contradicts $a_2^{-1} \in \min(w)$. So assume that $|u| > 2$. Let $a \in \min(u)$. Then $a \in \min(w)$, and there exist traces t, w' with $w = aw' = u^{-1}t$. If $a \notin \min(u^{-1})$, then $a \in \min(t)$ and aIu^{-1}. But the latter independence contradicts $a^{-1} \in \mathsf{alph}(u^{-1})$. Hence $a \in \min(u^{-1})$, i.e., $a^{-1} \in \max(u)$. Thus, we can write $u = ava^{-1}$ and $u^{-1} = av^{-1}a^{-1}$ with $v \neq \varepsilon$ (since $|u| > 2$). Since $ava^{-1} = u \preceq_p aw'$ and $av^{-1}a^{-1} = u^{-1} \preceq_p aw'$ and $\mathbb{M}(\Sigma^{\pm 1}, I)$ is cancellative, we have $v \preceq_p w'$, $v^{-1} \preceq_p w'$. Since $v \neq \varepsilon$, we have a contradiction to the fact that $|w|$ is minimal. □

Note that Lemma 58 also holds for the suffix order $\preceq_s$.

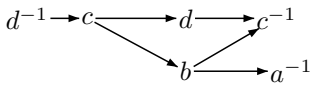
Example 59. We take the independence alphabet from Example 13 and consider the trace $w = [c^{-1}d^{-1}a^{-1}ba^{-1}cabdc^{-1}d^{-1}a^{-1}b^{-1}dca]_I \in \mathbb{M}(\Sigma^{\pm 1}, I)$, whose dependence graph looks as follows:



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



Hence, the core of w is $\mathsf{core}(w) = [d^{-1}cbdc^{-1}a^{-1}]_I$ and looks as follows:



Note that we have $\mathsf{NF}_R(w) \sqcap_p \mathsf{NF}_R(w^{-1}) = c^{-1}a^{-1}b$ and hence

$$\begin{aligned}\mathsf{NF}_R\Big(\big(\mathsf{NF}_R(w)\sqcap_p\mathsf{NF}_R(w^{-1})\big)^{-1}\mathsf{NF}_R(w)\big(\mathsf{NF}_R(w)\sqcap_p\mathsf{NF}_R(w^{-1})\big)\Big)\\ = \mathsf{NF}_R\Big(\big(c^{-1}a^{-1}b\big)^{-1}\big(c^{-1}d^{-1}a^{-1}bcbdc^{-1}a^{-1}b^{-1}ca\big)\big(c^{-1}a^{-1}b\big)\Big)\\ = d^{-1}cbdc^{-1}a^{-1} = \mathsf{core}(w).\end{aligned}$$

This fact holds for every trace, and will be proven next.

Lemma 60. *Let $w \in \mathsf{IRR}(R)$ and $d = w \sqcap_p w^{-1}$. Then $\mathsf{NF}_R(d^{-1}wd) = \mathsf{core}(w)$.*

Proof. Let $d = w \sqcap_p w^{-1}$. Thus, there are traces u, v such that $du = w = v^{-1}d^{-1}$ and $\min(u) \cap \min(v) = \emptyset$. By Levi's Lemma (Lemma 11) it follows that there are traces x, y_1, y_2, z such that $xy_1 = d$, $y_2z = u$, $xy_2 = v^{-1}$, $y_1z = d^{-1}$, and y_1Iy_2. Hence we have $y_1 \preceq_p d^{-1}$ and $y_1 \preceq_s d$, which is equivalent to $y_1^{-1} \preceq_p d^{-1}$. Moreover, since w is R-irreducible, so is d^{-1}. We can apply Lemma 58 to infer that $y_1 = \varepsilon$.

It follows that $x = d$, $z = d^{-1}$, and thus $w = du = dy_2z = dy_2d^{-1}$. Moreover, since $\min(y_2z) \cap \min(y_2^{-1}x^{-1}) = \min(u) \cap \min(v) = \emptyset$, we have $\min(y_2) \cap \min(y_2^{-1}) = \emptyset$. Hence, y_2 is the core of w. Moreover since w (and therefore y_2) is R-irreducible, we have $\mathsf{NF}_R(d^{-1}wd) = \mathsf{NF}_R(d^{-1}dy_2d^{-1}d) = y_2$. □

We now easily obtain:

Corollary 61. *For the fixed independence alphabet (Σ, I) the following problem can be solved in polynomial time:*

INPUT: An SLP $\mathbb{A}$ over $\Sigma^{\pm 1}$.
OUTPUT: An SLP $\mathbb{B}$ with $[\mathsf{val}(\mathbb{B})]_I = \mathsf{core}([\mathsf{val}(\mathbb{A})]_I)$

Proof. By Corollary 55 we can assume that $[\mathsf{val}(\mathbb{A})]_I$ is R-irreducible. By Corollary 56 we can compute in polynomial time an SLP $\mathbb{P}$ with $[\mathsf{val}(\mathbb{P})]_I = [\mathsf{val}(\mathbb{A})]_I \sqcap_p [\mathsf{val}(\mathbb{A})^{-1}]_I$. Lemma 60 implies

$$\mathsf{core}([\mathsf{val}(\mathbb{A})]_I) = \mathsf{NF}_R([\mathsf{val}(\mathbb{P})^{-1}\mathsf{val}(\mathbb{A})\mathsf{val}(\mathbb{P})]_I).$$

By Corollary 55 we can compute in polynomial time an SLP $\mathbb{B}$ with

$$[\mathsf{val}(\mathbb{B})]_I = \mathsf{NF}_R([\mathsf{val}(\mathbb{P})^{-1}\mathsf{val}(\mathbb{A})\mathsf{val}(\mathbb{P})]_I),$$

which concludes the proof. □

10. A pattern matching algorithm for connected patterns

Our second tool for proving Theorem 31 is a pattern matching algorithm for compressed traces. For two traces v and w we say that v is a factor of w if there exist traces x, y with $w = xvy$. This is equivalent to saying that the dependence graph D_w contains a convex subset such that the induced subgraph is isomorphic to D_v. We consider the following problem and show that it can be solved in polynomial time if the independence alphabet (Σ, I) satisfies certain conditions.

INPUT: An independence alphabet (Σ, I) and two SLPs $\mathbb{T}$ and $\mathbb{P}$ over Σ.
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 may assume that $\Sigma = \mathsf{alph}(\mathbb{T})$ and that Σ is connected. Otherwise we simply solve several instances of the latter problem separately. Also, 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 can view the projection morphism $\pi_\Gamma : \Sigma^* \to \Gamma^*$ also as a morphism $\pi_\Gamma : \mathbb{M}(\Sigma, I) \to \mathbb{M}(\Gamma, I \cap (\Gamma \times \Gamma))$ in the natural way, i.e., we define $\pi_\Gamma([u]_I) = [\pi_\Gamma(u)]_I$ (this is indeed well-defined). Since Σ is a constant size alphabet, we can compute in polynomial time an SLP (without initial variable) that contains for every variable $X \in V$ and every $\Gamma \subseteq \Sigma$ a variable X_Γ such that $\mathsf{val}(X_\Gamma) = \pi_\Gamma(\mathsf{val}_\mathbb{T}(X))$; see the proof of Lemma 3. If $\mathsf{rhs}_\mathbb{T}(X) = YZ$, then we simply set the right-hand side of X_Γ to $Y_\Gamma Z_\Gamma$.

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 $[\mathsf{val}(\mathbb{T})]_I$ if and only if there is a prefix $u \preceq_p [\mathsf{val}(\mathbb{T})]_I$ such that $u[\mathsf{val}(\mathbb{P})]_I \preceq_p [\mathsf{val}(\mathbb{T})]_I$, 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 if and only if there is a prefix $u \preceq_p w$ such that $uv \preceq_p w$, and $O = (|u|_a)_{a\in\Sigma}$. 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 $[\mathsf{val}(\mathbb{T})]_I$. 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.

Let X be a nonterminal of $\mathbb{T}$ with $\mathsf{rhs}_\mathbb{T}(X) = YZ$ and let O be an occurrence of $[\mathsf{val}(\mathbb{P})]_I$ in $[\mathsf{val}(X)]_I$. If there are $a, b \in \mathsf{alph}(\mathbb{P})$ such that $O_a < |\mathsf{val}(Y)|_a$ 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. We assume w.l.o.g. that $|\mathsf{val}(\mathbb{P})| \geq 2$, otherwise the problem reduces simply to checking whether there occurs a certain letter in $\mathsf{val}(\mathbb{T})$. This assumption implies that $[\mathsf{val}(\mathbb{P})]_I$ is a factor of $[\mathsf{val}(\mathbb{T})]_I$ if and only if there is a nonterminal X of $\mathbb{T}$ for

which there is an occurrence of $\mathbb{P}$ at the cut of X.

Example 62. We take the independence alphabet from Example 13 again. Let X be a nonterminal with $\mathsf{val}(X) = acbc\ ad\ cbc\ acbc\ acbc\ acbc\ acb|c\ acbc\ acbc\ acbc\ acb\ dc$ where '|' denotes the cut of X and $\mathsf{val}(\mathbb{P}) = acbc\ acbc\ acbc\ acbc\ acbc$. Then the occurrences of $\mathsf{val}(\mathbb{P})$ at the cut of X are $(1,1,2,1)$, $(2,2,4,1)$, $(3,3,6,1)$, and $(4,4,8,1)$ where the positions in a tuple correspond to the letters in our alphabet in the order a, b, c, d. We will see later how to construct them.

Lemma 63 ([32]). *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 if and only if (n_a, n_b) is an occurrence of $\pi_{\{a,b\}}(v)$ in $\pi_{\{a,b\}}(w)$ for all $(a,b) \in D$.*

Let $\Gamma, \Delta \subseteq \Sigma$. We call occurrences $(O_a)_{a\in\Gamma}$ and $(O'_a)_{a\in\Delta}$ *matching* if $(O_a)_{a\in\Gamma\cap\Delta} = (O'_a)_{a\in\Gamma\cap\Delta}$. An *arithmetic progression* is a subset of $\mathbb{N}^\Sigma$ of the form

$$\{(i_a)_{a\in\Sigma} + k \cdot (d_a)_{a\in\Sigma} \mid 0 \le k \le \ell\}.$$

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 this arithmetic progression is $\lceil \log_2(\ell) \rceil + \sum_{a\in\Sigma} (\lceil \log_2(i_a) \rceil + \lceil \log_2(d_a) \rceil)$. In Example 62, the occurrences of $\mathsf{val}(\mathbb{P})$ at the cut of X form the arithmetic progression $\big((1,1,2,1), (1,1,2,0), 3\big)$.

We will use Lemma 63 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 [18]. In particular Lemma 64 below was inspired by [18, Proposition 1].

Throughout the rest of this section we make the following assumption:

$$\mathsf{alph}(\mathbb{P}) \text{ is connected and } \{a,b\} \cap \mathsf{alph}(\mathbb{P}) \neq \emptyset \text{ for all } (a,b) \in D \text{ with } a \neq b. \quad (10)$$

Let X be a nonterminal of $\mathbb{T}$ and let O be an occurrence of $\mathbb{P}$ at the cut of X. Since the alphabet of the pattern is connected there must be some $a, b \in \Sigma$ with $(a,b) \in D$ 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 (see [29]) 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 $n \ge \ell$ and strings $u, v \in \{a,b\}^*$ with $v \preceq_p u$, $|u|_a = d_a$ and $|u|_b = d_b$. Moreover, by [29] the arithmetic progression $((i_a, i_b), (d_a, d_b), \ell)$ can be computed in time $\mathcal{O}(|\mathbb{T}|^2|\mathbb{P}|)$.[d] 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.

[d] In fact, in [29] it was shown that the arithmetic progression $(i_a + i_b, d_a + d_b, \ell)$ can be computed in time $\mathcal{O}(|\mathbb{T}|^2|\mathbb{P}|)$. Observe that from this the arithmetic progression $((i_a, i_b), (d_a, d_b), \ell)$ can be computed in time $|\mathbb{T}| + |\mathbb{P}|$.

The following lemma states that for each occurrence O at the cut of a nonterminal either there is a pair $(a,b) \in D$ such that the projection of O onto $\{a,b\}$ is the first or the last element of an arithmetic progression, or all projections of O lie at the cut of the same nonterminal.

Lemma 64. *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 $a,b \in \mathsf{alph}(\mathbb{P})$ with $(a,b) \in D$ 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 $X_{\{a,b\}}$.

Proof. Let $\mathsf{rhs}_{\mathbb{T}}(X) = YZ$. Clearly, by our general assumption (10) it suffices to show that either (ii) holds, or $O_a < |\mathsf{val}(Y)|_a < O_a + |\mathsf{val}(\mathbb{P})|_a$ for all $a \in \mathsf{alph}(\mathbb{P})$. We show this assertion by induction on $|\mathsf{alph}(\mathbb{P})|$. If $\mathsf{alph}(\mathbb{P})$ is a singleton, then it is trivially true.

Next, we consider the case $|\mathsf{alph}(\mathbb{P})| = 2$. So let $\{a,b\} = \mathsf{alph}(\mathbb{P})$ and hence $(a,b) \in D$ by (10). Assume that (ii) does not hold. Consider the arithmetic progression $((i_a,i_b),(d_a,d_b),\ell)$ of occurrences of $\mathsf{val}(\mathbb{P})$ at the cut of $X_{\{a,b\}}$. Then $\mathsf{val}(\mathbb{P})$ is of the form $u^n v$ for some $n \geq \ell$ and strings $u,v \in \{a,b\}^*$ with $v \preceq_p u$, $|u|_a = d_a$ and $|u|_b = d_b$. We conclude that $d_a, d_b > 0$ as otherwise $|\mathsf{alph}(\mathbb{P})| \leq 1$. Suppose for contradiction that $i_a + \ell d_a > |\mathsf{val}(Y)|_a$. Since no prefix w of $\pi_{\{a,b\}}(\mathsf{val}(X))$ can satisfy $|w|_a > |\mathsf{val}(Y)|_a$ and $|w|_b < |\mathsf{val}(Y)|_b$ we conclude $i_b + \ell d_b \geq |\mathsf{val}(Y)|_b$. But then the occurrence $(i_a + \ell d_a, i_b + \ell d_b)$ is not at the cut of $X_{\{a,b\}}$, which is a contradiction. Hence $i_a + \ell d_a \leq |\mathsf{val}(Y)|_a$ and by symmetry $i_b + \ell d_b \leq |\mathsf{val}(Y)|_b$. Similarly, since (i_a,i_b) is an occurrences of $\mathsf{val}(\mathbb{P})$ at the cut of $X_{\{a,b\}}$, we get $|\mathsf{val}(Y)|_a \leq i_a + |\mathsf{val}(\mathbb{P})|_a$ and $|\mathsf{val}(Y)|_b \leq i_b + |\mathsf{val}(\mathbb{P})|_b$. As $\pi_{\{a,b\}}(O)$ is neither the first nor the last element of the arithmetic progression (we assume that (ii) does not hold), we have $O_a = i_a + k d_a$ and $O_b = i_b + k d_b$ for some $0 < k < \ell$ and hence $O_a < |\mathsf{val}(Y)|_a < O_a + |\mathsf{val}(\mathbb{P})|_a$ and $O_b < |\mathsf{val}(Y)|_b < O_b + |\mathsf{val}(\mathbb{P})|_b$ as required.

Now, suppose that $|\mathsf{alph}(\mathbb{P})| \geq 3$. Since O is an occurrence at the cut of X, there are $a,b \in \mathsf{alph}(\mathbb{P})$ such that $O_a < |\mathsf{val}(Y)|_a$ and $O_b + |\mathsf{val}(\mathbb{P})|_b > |\mathsf{val}(Y)|_b$. We may assume that $(a,b) \in D$. Indeed, if $O_a + |\mathsf{val}(\mathbb{P})|_a > |\mathsf{val}(Y)|_a$ choose $a = b$. Otherwise, since $\mathsf{alph}(\mathbb{P})$ is connected there is a dependence path between a and b. Since $O_a + |\mathsf{val}(\mathbb{P})|_a \leq |\mathsf{val}(Y)|_a$, there must be an edge $(a',b') \in D$ on this path such that $a',b' \in \mathsf{alph}(\mathbb{P})$, $O_{a'} + |\mathsf{val}(\mathbb{P})|_{a'} \leq |\mathsf{val}(Y)|_{a'}$ (and hence $O_{a'} < |\mathsf{val}(Y)|_{a'}$), and $O_{b'} + |\mathsf{val}(\mathbb{P})|_{b'} > |\mathsf{val}(Y)|_{b'}$.

Next consider a spanning tree of $(\mathsf{alph}(\mathbb{P}), D \cap \mathsf{alph}(\mathbb{P}) \times \mathsf{alph}(\mathbb{P}))$ which contains the edge (a,b) (in case $a \neq b$). Let $c \notin \{a,b\}$ be a leaf of this spanning tree (it exists since $|\mathsf{alph}(\mathbb{P})| \geq 3$). Obviously, $\Delta = \mathsf{alph}(\mathbb{P}) \setminus \{c\}$ is connected and $\pi_\Delta(O)$ is at the cut of X_Δ. Thus we can apply the induction hypothesis to $\pi_\Delta(\mathsf{val}(\mathbb{P}))$ and X_Δ. We get either (ii) (in which case we are done) or $O_a < |\mathsf{val}(Y)|_a < O_a + |\mathsf{val}(\mathbb{P})|_a$ for all $a \in \Delta$. Assume the latter. In particular, $O_d < |\mathsf{val}(Y)|_d < O_d + |\mathsf{val}(\mathbb{P})|_d$

for some $d \in \Delta$ with $(c,d) \in D$. Hence, $\pi_{\{d,c\}}(O)$ is at the cut of $X_{\{d,c\}}$. Thus, applying the induction hypothesis also to $\pi_{\{d,c\}}(\mathsf{val}(\mathbb{P}))$ and $X_{\{d,c\}}$ we get either (ii) or $O_c < |\mathsf{val}(Y)|_c < O_c + |\mathsf{val}(\mathbb{P})|_c$. □

The last lemma 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 are $a, b \in \mathsf{alph}(\mathbb{P})$ with $(a,b) \in D$ 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 64, if O is periodic, then $\pi_{\{a,b\}}(O)$ is an element of the arithmetic progression of occurrences of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X_{\{a,b\}}$ for all $(a,b) \in D$ (but neither the first nor the last element, if $a, b \in \mathsf{alph}(\mathbb{P})$). The next proposition shows that we can decide in polynomial time whether there are single occurrences of $\mathsf{val}(\mathbb{P})$ in $\mathbb{T}$.

Proposition 65. *For given SLPs $\mathbb{T}$ and $\mathbb{P}$ we can decide in time $(|\mathbb{T}| + |\mathbb{P}|)^{\mathcal{O}(1)}$ whether there is a single occurrence at the cut of some nonterminal of $\mathbb{T}$.*

Proof. We do the following for all $a, b \in \mathsf{alph}(\mathbb{P})$ with $(a,b) \in D$ and all nonterminals X of $\mathbb{T}$: First we check using [29] whether an occurrence of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X_{\{a,b\}}$ exists. If such an occurrence exists, then we can compute (using again [29]) the first occurrence (O_a^f, O_b^f) and the last occurrence (O_a^l, O_b^l) in the arithmetic progression of occurrences of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X_{\{a,b\}}$. For all $(O_a, O_b) \in \{(O_a^f, O_b^f), (O_a^l, O_b^l)\}$ we check, whether (O_a, O_b) is a projection of an occurrence of $\mathsf{val}(\mathbb{P})$ in $\mathsf{val}(X)$ as follows.

Let $a_1, \ldots, a_n$ be an enumeration of Σ such that $a = a_1$, $b = a_2$ and $D(a_i) \cap \{a_1, \ldots, a_{i-1}\} \neq \emptyset$ for all $2 \leq i \leq n$. Moreover, we require that the elements of $\mathsf{alph}(\mathbb{P})$ appear at the beginning of our enumeration, i.e., are the elements $a_1, \ldots, a_j$ for some $j \leq n$. This can be assumed since Σ and $\mathsf{alph}(\mathbb{P})$ are connected. We iterate over $3 \leq i \leq n$ and compute, if possible, an integer O_{a_i} such that $(O_{a_1}, \ldots, O_{a_i})$ is an occurrence of $\pi_{\{a_1,\ldots,a_i\}}(\mathsf{val}(\mathbb{P}))$ in $\pi_{\{a_1,\ldots,a_i\}}(\mathsf{val}(X))$.

So let $i \geq 3$, $d = a_i$, and $\Delta = \{a_1, \ldots, a_{i-1}\}$. By our general assumption (10) we can choose some $c \in \Delta \cap \mathsf{alph}(\mathbb{P})$ such that $(c,d) \in D$. Let us further assume that we have already constructed an occurrence $(O_{a_1}, \ldots, O_{a_{i-1}})$ of $\pi_\Delta(\mathsf{val}(\mathbb{P}))$ in $\pi_\Delta(\mathsf{val}(X))$. First, we compute the unique number $k \geq 0$ such that $d^k c$ is a prefix of $\pi_{\{c,d\}}(\mathsf{val}(\mathbb{P}))$. Then, we compute the word $w \in \{c,d\}^*$ such that wd^kc is a prefix of $\pi_{c,d}(\mathsf{val}(X))$ and $|w|_c = O_c$. If such a prefix w does not exist, then there is no occurrence $(O_{a_1}, \ldots, O_{a_{i-1}}, O_d)$ of $\pi_{\Delta\cup\{d\}}(\mathsf{val}(\mathbb{P}))$ in $\pi_{\Delta\cup\{d\}}(\mathsf{val}(X))$. On the other hand, if such a prefix w exists, it exists uniquely. Note that if there is an occurrence $(O_{a_1}, \ldots, O_{a_{i-1}}, O_d)$ of $\pi_{\Delta\cup\{d\}}(\mathsf{val}(\mathbb{P}))$ in $\pi_{\Delta\cup\{d\}}(\mathsf{val}(X))$, then we must have $O_d = |w|_d$. Hence, we set $|w|_d = O_d$. Last, using [29] we check in polynomial time for all $e \in D(d) \cap \Delta$ whether (O_e, O_d) is an occurrence of $\pi_{\{d,e\}}(\mathsf{val}(\mathbb{P}))$ in $\pi_{\{d,e\}}\mathsf{val}(X)$. By Lemma 63, the latter holds if and only if $(O_{a_1}, \ldots, O_{a_{i-1}}, O_d)$ is an occurrence of $\pi_{\Delta\cup\{d\}}(\mathsf{val}(\mathbb{P}))$ in $\pi_{\Delta\cup\{d\}}(\mathsf{val}(X))$. □

It remains to be shown that for every nonterminal X of $\mathbb{T}$ we can compute the periodic occurrences of $\mathsf{val}(\mathbb{P})$ at the cut of X. To this aim we define the amalgamation of arithmetic progressions. Let $\Gamma, \Gamma' \subseteq \Sigma$ such that $\Gamma \cap \Gamma' \neq \emptyset$. Consider two arithmetic progressions

$$p = ((i_a)_{a\in\Gamma}, (d_a)_{a\in\Gamma}, \ell), \qquad p' = ((i'_a)_{a\in\Gamma'}, (d'_a)_{a\in\Gamma'}, \ell').$$

The *amalgamation* of p and p' is

$$p \otimes p' = \{v = (v_a)_{a\in\Gamma\cup\Gamma'} \mid \pi_\Gamma(v) \in p \text{ and } \pi_{\Gamma'}(v) \in p'\}.$$

Example 66. We continue Example 62 and show how to compute occurrences at the cut. First we consider the projections of $\mathsf{val}(\mathbb{P})$ and $\mathsf{val}(X)$:

$$\begin{aligned} \pi_{\{a,b\}}(\mathsf{val}(\mathbb{P})) &= (ab)^5 & \mathsf{val}(X_{\{a,b\}}) &= (ab)^6|(ab)^4 \\ \pi_{\{b,c\}}(\mathsf{val}(\mathbb{P})) &= (cbc)^5 & \mathsf{val}(X_{\{b,c\}}) &= (cbc)^5cb|c(cbc)^4 \\ \pi_{\{c,d\}}(\mathsf{val}(\mathbb{P})) &= c^{10} & \mathsf{val}(X_{\{c,d\}}) &= c^2dc^9|c^8dc \end{aligned}$$

For the projections we find the arithmetic progressions p_{ab}, p_{bc}, p_{cd} of occurrences at the cut:

$$\begin{aligned} &\text{occurrences of } \pi_{\{a,b\}}(\mathsf{val}(\mathbb{P})) \text{ at the cut of } X_{\{a,b\}} : p_{ab} = \big((2,2),(1,1),3\big) \\ &\text{occurrences of } \pi_{\{b,c\}}(\mathsf{val}(\mathbb{P})) \text{ at the cut of } X_{\{b,c\}} : p_{bc} = \big((1,2),(1,2),4\big) \\ &\text{occurrences of } \pi_{\{c,d\}}(\mathsf{val}(\mathbb{P})) \text{ at the cut of } X_{\{c,d\}} : p_{cd} = \big((2,1),(1,0),7\big). \end{aligned}$$

Note that in p_{ab} the first component corresponds to a and the second to b whereas in p_{bc} the first component corresponds to b and the second to c. We amalgamate the arithmetic progressions and obtain $p_{abc} = p_{ab} \otimes p_{bc} = \big((2,2,4),(1,1,2),3\big)$. If we again amalgamate we obtain $p_{abcd} = p_{abc} \otimes p_{cd} = \big((2,2,4,1),(1,1,2,0),2\big)$. This way we found occurrences $(2,2,4,1)$, $(3,3,6,1)$ and $(4,4,8,1)$ of $\mathbb{P}$ at the cut of X. Observe that there is a fourth occurrence $(1,1,2,1)$ that we did not find this way which is single.

Lemma 67. *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 $(|p| + |p'|)^{\mathcal{O}(1)}$.*

Proof. We need to solve the system of linear equations

$$[\ i_b + d_b \cdot x = i'_b + d'_b \cdot y\ ]_{b\in\Gamma\cap\Gamma'} \tag{11}$$

for integers x and y under the constraint

$$0 \leq x \leq \ell \text{ and } 0 \leq y \leq \ell'. \tag{12}$$

Let us fix an $a \in \Gamma \cap \Gamma'$. First we solve the single equation

$$i_a + d_a \cdot x = i'_a + d'_a \cdot y. \tag{13}$$

for non-negative integers x and y. The solutions are given by the least solution plus a multiple of the least common multiple of d_a and d'_a. We start by computing $g = \gcd(d_a, d'_a)$. If $i_a \neq i'_a \bmod g$, then there is no solution for equation (13) and hence $p \otimes p' = \emptyset$. In this case we stop. Otherwise, we compute the least solution $s_a \geq \max(i_a, i'_a)$ of the simultaneous congruences

$$z = i_a \bmod d_a,$$
$$z = i'_a \bmod d'_a.$$

This can be accomplished with $(\log(d_a)+\log(d'_a))^2$ many bit operations; see e.g. [1]. Let $k = (s_a - i_a)/d_a \geq 0$ and $k' = (s_a - i'_a)/d'_a \geq 0$. Now, the non-negative solutions of equation (13) are given by

$$(x, y) = (k + \frac{d'_a}{g} \cdot t, k' + \frac{d_a}{g} \cdot t) \text{ for all } t \geq 0. \tag{14}$$

If $|\Gamma \cap \Gamma'| = 1$ we adapt the range for t such that the constraint (12) is satisfied and we are done.

Otherwise, (11) is a system of at least 2 linear equations in 2 variables. Hence (11) has at least 2 (and then infinitely many) solutions if and only if any two equations are linearly dependent over $\mathbb{Q}$, i.e. for all $b \in \Gamma \cap \Gamma'$ the following holds:

$$\exists k_b \in \mathbb{Q} : d_a = k_b \cdot d_b, \quad d'_b = k_b \cdot d'_a \text{ and } i'_a - i_a = k_b \cdot (i'_b - i_b) \tag{15}$$

In this case all solutions of equation (13) are solutions of system (11). Thus we can test condition (15) for all $b \in \Gamma \cap \Gamma'$ and in case it holds it only remains to adapt the range for t such that the constraint (12) is satisfied. Otherwise there is at most one solution and we can fix $b \in \Gamma \cap \Gamma'$ such that (15) does not hold. We plug the solution (14) into $i_b + d_b \cdot x = i'_b + d'_b \cdot y$ and obtain

$$i_b + (k + \frac{d'_a}{g} \cdot t) \cdot d_b = i'_b + (k' + \frac{d_a}{g} \cdot t) \cdot d'_b.$$

We can solve this for t (if possible) and test whether this gives rise to a solution for (11) under the constraint (12). □

Proposition 68. *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 $(|\mathbb{T}| + |\mathbb{P}|)^{\mathcal{O}(1)}$.*

Proof. As in the proof of Proposition 65 let $a_1, \ldots, a_n$ be an enumeration of Σ such that $\{a_1, \ldots, a_{i-1}\} \cap D(a_i) \neq \emptyset$ for all $2 \leq i \leq n$ and the elements of $\mathsf{alph}(\mathbb{P})$ appear at the beginning of the enumeration. We iterate over $1 \leq i \leq n$ and compute the arithmetic progressions of the periodic occurrences of $\pi_{\{a_1,\ldots,a_i\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X_{\{a_1,\ldots,a_i\}}$. For $i = 1$ this is easy.

So let $i \geq 2$, let $a = a_i$ and let $\Delta = \{a_1, \ldots, a_{i-1}\}$. Assume that the periodic occurrences of $\pi_\Delta(\mathsf{val}(\mathbb{P}))$ at the cut of X_Δ are given by the arithmetic progression $p = ((i_c)_{c\in\Delta}, (d_c)_{c\in\Delta}, \ell)$. For all $b \in D(a) \cap \Delta$ let $p_{\{a,b\}}$ be the arithmetic progression

of all occurrences of $\pi_{\{a,b\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X_{\{a,b\}}$ (without the first and the last occurrence if $a, b \in \mathsf{alph}(\mathbb{P})$). Recall that we assume that $\{c, d\} \cap \mathsf{alph}(\mathbb{P}) \neq \emptyset$ for all $c, d \in \Sigma$ with $(c, d) \in D$ and $c \neq d$. Hence, by Lemma 63, O is a periodic occurrence of $\pi_{\{a_1,\ldots,a_i\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X_{\{a_1,\ldots,a_i\}}$ if and only if $\pi_\Delta(O) \in p$ and $(O_a, O_b) \in p_{\{a,b\}}$ for all $b \in D(a) \cap \Delta$. Hence the periodic occurrences of $\pi_{\{a_1,\ldots,a_i\}}(\mathsf{val}(\mathbb{P}))$ at the cut of $X_{\{a_1,\ldots,a_i\}}$ are given by the arithmetic progression

$$\bigotimes_{b \in D(a) \cap \Delta} p_{\{a,b\}} \otimes p.$$

The result follows now from Lemma 67. □

Summarizing the last section we get the following theorem.

Theorem 69. *Given an independence alphabet (Σ, I) and two SLPs $\mathbb{P}$ and $\mathbb{T}$ over Σ 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. Note that our assumption (10) is satisfied if $\mathsf{alph}(\mathbb{P}) = \mathsf{alph}(\mathbb{T})$. Recall that we may assume that $\mathsf{alph}(\mathbb{T})$ is connected and that $|\mathsf{val}(\mathbb{P})| \geq 2$. Using Proposition 65 we can decide in polynomial time whether a single occurrence of $\mathbb{P}$ at the cut of some nonterminal of $\mathbb{T}$ exists. By Proposition 68 we can compute the periodic occurrences of $\mathbb{P}$ at the cuts of all nonterminals from $\mathbb{T}$ in polynomial time. The result follows, since by definition $[\mathsf{val}(\mathbb{P})]_I$ is a factor of $[\mathsf{val}(\mathbb{T})]_I$ if and only if there is a nonterminal X of $\mathbb{T}$ such that there is either a single occurrence of $\mathbb{P}$ at the cut of X or a periodic occurrence of $\mathbb{P}$ at the cut of X. □

Remark 70. In the last section we actually proved the above theorem under weaker assumptions: We only need for each connected component Σ_i of $\mathsf{alph}(\mathbb{T})$ that $\Sigma_i \cap \mathsf{alph}(\mathbb{P})$ is connected and that $\{a, b\} \cap \mathsf{alph}(\mathbb{P}) \neq \emptyset$ for all $(a, b) \in D \cap (\Sigma_i \times \Sigma_i)$ with $a \neq b$.

11. Compressed conjugacy

In this section we will prove Theorem 31. For this, we will follow the approach from [32,47] for non-compressed traces. We will fix the graph group $\mathbb{G}(\Sigma, I)$ for the rest of this section. The following result allows us to transfer the conjugacy problem to a problem on (compressed) traces:

Theorem 71 ([32,47]). *Let $u, v \in \mathbb{M}(\Sigma^{\pm 1}, I)$. Then the following are equivalent:*

(1) u is conjugated to v in $\mathbb{G}(\Sigma, I)$.
(2) There exists $x \in \mathbb{M}(\Sigma^{\pm 1}, I)$ such that $x\,\mathsf{core}(u) = \mathsf{core}(v)\,x$ in $\mathbb{M}(\Sigma^{\pm 1}, I)$ (it is said that $\mathsf{core}(u)$ and $\mathsf{core}(v)$ are conjugated in $\mathbb{M}(\Sigma^{\pm 1}, I)$).
(3) $|\mathsf{core}(u)|_a = |\mathsf{core}(v)|_a$ for all $a \in \Sigma^{\pm 1}$ and there exists $k \leq |\Sigma^{\pm 1}|$ such that $\mathsf{core}(u)$ is a factor of $\mathsf{core}(v)^k$.

The equivalence of (1) and (2) can be found in [47], the equivalence of (2) and (3) is shown in [32]. We can now infer Theorem 31:

Proof of Theorem 31. 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 Corollary 61, 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 Theorem 71, it suffices to check the following two conditions:

(a) $|\mathsf{core}([\mathsf{val}(\mathbb{C})]_I)|_a = |\mathsf{core}([\mathsf{val}(\mathbb{D})]_I)|_a$ for all $a \in \Sigma^{\pm 1}$
(b) There exists $k \leq |\Sigma^{\pm 1}|$ such that $\mathsf{core}([\mathsf{val}(\mathbb{C})]_I)$ is a factor of $\mathsf{core}([\mathsf{val}(\mathbb{D})]_I)^k$.

Condition (a) can be easily checked in polynomial time, since the number of occurrences of a symbol in a compressed strings can be computed in polynomial time. Moreover, condition (b) can be checked in polynomial time by Theorem 69, since (by condition (a)) we can assume that $\mathsf{alph}(\mathsf{val}(\mathbb{C})) = \mathsf{alph}(\mathsf{val}(\mathbb{D}))$.

12. Open problems

We have shown that the restricted simultaneous compressed conjugacy problem for a graph product of finitely generated groups (see Section 3) can be reduced to the compressed word problems and the restricted simultaneous compressed conjugacy problems for the vertex groups in polynomial time (Theorem 27). It remains unclear whether this holds also for the general simultaneous compressed conjugacy problem as well. It is even unclear, whether the simultaneous compressed conjugacy problem for a graph group can be solved in polynomial time. It is also unclear, whether the compressed conjugacy problem for a graph product of finitely generated groups can be reduced to the compressed word problems and the compressed conjugacy problem for the vertex groups in polynomial time. Here, we have at least a polynomial time algorithm for graph groups (Theorem 31).

Additionally we do not know whether the general compressed pattern matching problem for traces, where we drop restriction (10) on page 44, can be decided in polynomial time.

For graph groups, we do not know whether our compressed decision problems (compressed word problem, compressed conjugacy problem, and restricted simultaneous compressed conjugacy problem) can be solved in polynomial time, if the independence alphabet is part of the input. Finally, we would like to know, whether the graph product of finitely generated groups with finitely generated automorphism groups has a finitely generated automorphism group.

References

[1] E. Bach and J. Shallit. *Algorithmic Number Theory*, volume I: Efficient Algorithms. MIT Press, 1996.

[2] A. Bertoni, C. Choffrut, and R. Radicioni. Literal shuffle of compressed words. In *Proceeding of the 5th IFIP International Conference on Theoretical Computer Science (IFIP TCS 2008), Milano (Italy)*, pages 87–100. Springer, 2008.
[3] R. V. Book and F. Otto. *String–Rewriting Systems*. Springer, 1993.
[4] W. W. Boone. The word problem. *Annals of Mathematics (2)*, 70:207–265, 1959.
[5] R. Charney. An introduction to right-angled Artin groups. *Geometriae Dedicata*, 125:141–158, 2007.
[6] R. Charney, J. Crisp, and K. Vogtmann. Automorphisms of 2-dimensional right-angled Artin groups. *Geometry & Topology*, 11:2227–2264, 2007.
[7] R. Charney and K. Vogtmann. Finiteness properties of automorphism groups of right-angled Artin groups. *Bulletin of the London Mathematical Society*, 41(1):94–102, 2009.
[8] R. Cori, Y. Métivier, and W. Zielonka. Asynchronous mappings and asynchronous cellular automata. *Information and Computation*, 106(2):159–202, 1993.
[9] L. Corredor and M. Gutierrez. A generating set for the automorphism group of a graph product of abelian groups. Technical report, arXiv.org, 2009. `http://arxiv.org/abs/0911.0576`.
[10] J. Crisp, E. Godelle, and B. Wiest. The conjugacy problem in right-angled Artin groups and their subgroups. *Journal of Topology*, 2(3), 2009.
[11] M. B. Day. Peak reduction and finite presentations for automorphism groups of right-angled artin groups. *Geometry & Topology*, 13(2):817–855, 2009.
[12] V. Diekert. *Combinatorics on Traces*. Number 454 in Lecture Notes in Computer Science. Springer, 1990.
[13] V. Diekert and M. Lohrey. Word equations over graph products. *International Journal of Algebra and Computation*, 18(3):493–533, 2008.
[14] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.
[15] C. Droms. A complex for right-angled coxeter groups. *Proceedings of the American Mathematical Society*, 131(8):2305–2311, 2003.
[16] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston. *Word processing in groups*. Jones and Bartlett, Boston, 1992.
[17] L. Gasieniec, M. Karpinski, W. Plandowski, and W. Rytter. Efficient algorithms for Lempel-Ziv encoding (extended abstract). In R. G. Karlsson and A. Lingas, editors, *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory (SWAT 1996), Reykjavík (Iceland)*, number 1097 in Lecture Notes in Computer Science, pages 392–403. Springer, 1996.
[18] B. Genest and A. Muscholl. Pattern matching and membership for hierarchical message sequence charts. *Theory of Computing Systems*, 42(4):536–567, 2008.
[19] S. M. Gersten, D. F. Holt, and T. R. Riley. Isoperimetric inequalities for nilpotent groups. *Geometric and Functional Analysis*, 13(4):795–814, 2003.
[20] E. R. Green. *Graph Products of Groups*. PhD thesis, The University of Leeds, 1990.
[21] C. Hagenah. *Gleichungen mit regulären Randbedingungen über freien Gruppen*. PhD thesis, University of Stuttgart, Institut für Informatik, 2000.
[22] N. Haubold and M. Lohrey. Compressed word problems in HNN-extensions and amalgamated products. In *Proceedings of Computer Science in Russia (CSR 2009)*, number 5675 in Lecture Notes in Computer Science, pages 237–249. Springer, 2009. long version to appear in Theory of Computing Systems.
[23] N. Haubold, M. Lohrey, and C. Mathissen. Compressed conjugacy and the word problem for outer automorphism groups of graph groups. In *Proceedings of Developments in Language Theory (DLT 2010)*, number 6224 in Lecture Notes in Computer Science, pages 218–230. Springer, 2010.

[24] S. Hermiller and J. Meier. Algorithms and geometry for graph products of groups. *Journal of Algebra*, 171:230–257, 1995.
[25] I. Kapovich, A. Myasnikov, P. Schupp, and V. Shpilrain. Generic-case complexity, decision problems in group theory, and random walks. *Journal of Algebra*, 264(2):665–694, 2003.
[26] M. I. Kargapolov and J. I. Merzljakov. *Fundamentals of the theory of groups*, volume 62 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1979.
[27] D. Kuske and M. Lohrey. Logical aspects of Cayley-graphs: the monoid case. *International Journal of Algebra and Computation*, 16(2):307–340, 2006.
[28] 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.
[29] Y. Lifshits. Processing compressed texts: A tractability border. In B. Ma and K. Zhang, editors, *Proceedings of the 18th Annual Symposium on Combinatorial Pattern Matching (CPM 2007), London (Canada)*, number 4580 in Lecture Notes in Computer Science, pages 228–240. Springer, 2007.
[30] Y. Lifshits and M. Lohrey. Querying and embedding compressed texts. In R. Královic and P. Urzyczyn, editors, *Proceedings of the 31th International Symposium on Mathematical Foundations of Computer Science (MFCS 2006), Stará Lesná (Slovakia)*, number 4162 in Lecture Notes in Computer Science, pages 681–692. Springer, 2006.
[31] R. J. Lipton and Y. Zalcstein. Word problems solvable in logspace. *Journal of the Association for Computing Machinery*, 24(3):522–526, 1977.
[32] 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.
[33] M. Lohrey. Word problems and membership problems on compressed words. *SIAM Journal on Computing*, 35(5):1210 – 1240, 2006.
[34] M. Lohrey and S. Schleimer. Efficient computation in groups via compression. In *Proceedings of Computer Science in Russia (CSR 2007), Ekatarinburg (Russia)*, number 4649 in Lecture Notes in Computer Science, pages 249–258. Springer, 2007.
[35] R. C. Lyndon and P. E. Schupp. *Combinatorial Group Theory*. Springer, 1977.
[36] J. Macdonald. Compressed words and automorphisms in fully residually free groups. *International Journal of Algebra and Computation*, 20(3):343–355, 2010.
[37] M. Miyazaki, A. Shinohara, and M. Takeda. An improved pattern matching algorithm for strings in terms of straight-line programs. In A. Apostolico and J. Hein, editors, *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching (CPM 97), Aarhus (Denmark)*, number 1264 in Lecture Notes in Computer Science, pages 1–11. Springer, 1997.
[38] A. Myasnikov, V. Shpilrain, and A. Ushakov. *Group-based Cryptography*. Birkhäuser, 2008.
[39] P. S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *American Mathematical Society, Translations, II. Series*, 9:1–122, 1958.
[40] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.
[41] W. Plandowski. Testing equivalence of morphisms on context-free languages. In J. van Leeuwen, editor, *Second Annual European Symposium on Algorithms (ESA'94), Utrecht (The Netherlands)*, number 855 in Lecture Notes in Computer Science, pages 460–470. Springer, 1994.
[42] W. Plandowski and W. Rytter. Application of Lempel-Ziv encodings to the solution of word equations. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP 1998)*, number 1443 in Lecture Notes in Computer Science, pages 731–742. Springer, 1998.
[43] W. Plandowski and W. Rytter. Complexity of language recognition problems for

compressed words. In J. Karhumäki, H. A. Maurer, G. Paun, and G. Rozenberg, editors, *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 262–272. Springer, 1999.

[44] S. Schleimer. Polynomial-time word problems. *Commentarii Mathematici Helvetici*, 83(4):741–765, 2008.

[45] H. Servatius. Automorphisms of graph groups. *Journal of Algebra*, 126(1):34–60, 1989.

[46] C. Wrathall. The word problem for free partially commutative groups. *Journal of Symbolic Computation*, 6(1):99–104, 1988.

[47] C. Wrathall. Free partially commutative groups. In *Combinatorics, computing and complexity*, pages 195–216. Kluwer Academic Press, 1989.