Knapsack in Graph Groups, HNN-Extensions and Amalgamated Products

Markus Lohrey

Universität Siegen, Germany
lohrey@eti.uni-siegen.de

Georg Zetzsche*

LSV, CNRS & ENS Cachan, Université Paris-Saclay, France
zetzsche@lsv.fr

Abstract

It is shown that the knapsack problem, which was introduced by Myasnikov et al. for arbitrary finitely generated groups, can be solved in NP for graph groups. This result even holds if the group elements are represented in a compressed form by SLPs, which generalizes the classical NP-completeness result of the integer knapsack problem. We also prove general transfer results: NP-membership of the knapsack problem is passed on to finite extensions, HNN-extensions over finite associated subgroups, and amalgamated products with finite identified subgroups.

1 Introduction

In their paper [41], Myasnikov, Nikolaev, and Ushakov started the investigation of classical discrete optimization problems, which are classically formulated over the integers, for arbitrary in general non-commutative groups. Among other problems, they introduced for a finitely generated group G the *knapsack problem* and the *subset sum problem*. The input for the knapsack

*This author is supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD).

problem is a sequence of group elements $g_1, \ldots, g_k, g \in G$ (specified by finite words over the generators of G) and it is asked whether there exists a solution $(x_1, \ldots, x_k) \in \mathbb{N}^k$ of the equation $g_1^{x_1} \cdots g_k^{x_k} = g$. For the subset sum problem one restricts the solution to $\{0,1\}^k$. For the particular case $G = \mathbb{Z}$ (where the additive notation $x_1 \cdot g_1 + \cdots + x_k \cdot g_k = g$ is usually preferred) these problems are NP-complete if the numbers $g_1, \ldots, g_k, g$ are encoded in binary representation. For subset sum, this is a classical result from Karp's seminal paper [26] on NP-completeness. Knapsack for integers is usually formulated in a more general form in the literature; NP-completeness of the above form (for binary encoded integers) was shown in [19], where the problem was called MULTISUBSET SUM.[1] Interestingly, if we consider subset sum for the group $G = \mathbb{Z}$, but encode the input numbers $g_1, \ldots, g_k, g$ in unary notation, then the problem is in DLOGTIME-uniform TC^0 (a small subclass of polynomial time and even of logarithmic space that captures the complexity of multiplication of binary encoded numbers) [16], and the same holds for knapsack, since the instance $x_1 \cdot g_1 + \cdots + x_k \cdot g_k = g$ has a solution if and only if it has a solution with $x_i \leq k \cdot (\max\{g_1, \ldots, g_k, g\})^3$ [42]. This allows to reduce unary knapsack to unary subset sum. See [23] for related results.

In [41] the authors encode elements of the finitely generated group G by words over the group generators and their inverses. For $G = \mathbb{Z}$ this representation corresponds to the unary encoding of integers. Among others, the following results were shown in [41]:

- Subset sum and knapsack can be solved in polynomial time for every hyperbolic group.
- Subset sum for a virtually nilpotent group (a finite extension of a nilpotent group) can be solved in polynomial time.
- For the following groups, subset sum is NP-complete (whereas the word problem can be solved in polynomial time): free metabelian non-abelian groups of finite rank, the wreath product $\mathbb{Z} \wr \mathbb{Z}$, Thompson's group F, and the Baumslag-Solitar group $\mathrm{BS}(1,2)$.

Further results on knapsack and subset sum have been recently obtained in [29]:

[1] Note that if we ask for a solution $(x_1, \ldots, x_k)$ in $\mathbb{Z}^k$, then knapsack can be solved in polynomial time (even for binary encoded integers) by checking whether $\gcd(g_1, \ldots, g_k)$ divides g.

- For a virtually nilpotent group, subset sum belongs to NL (nondeterministic logspace).
- There is a nilpotent group of class 2 (in fact, a direct product of sufficiently many copies of the discrete Heisenberg group $H_3(\mathbb{Z})$), for which knapsack is undecidable.
- The knapsack problem for the discrete Heisenberg group $H_3(\mathbb{Z})$ is decidable. In particular, together with the previous point it follows that decidability of knapsack is not preserved under direct products.
- There is a polycyclic group with an NP-complete subset sum problem.
- The knapsack problem is decidable for every co-context-free group.

The focus of this paper will be on the knapsack problem. We will prove that this problem can be solved in NP for every *graph group*. Graph groups are also known as right-angled Artin groups or free partially commutative groups. A graph group is specified by a finite simple graph. The vertices are the generators of the group, and two generators a and b are allowed to commute if and only if a and b are adjacent. Graph groups somehow interpolate between free groups and free abelian groups and can be seen as a group counterpart of trace monoids (free partially commutative monoids), which have been used for the specification of concurrent behavior. In combinatorial group theory, graph groups are currently an active area of research, mainly because of their rich subgroup structure (see e.g. [5, 9, 18]). To prove that knapsack belongs to NP for a graph group, we proceed in two steps:

- We show that if an instance $g_1^{x_1} \cdots g_k^{x_k} = g$ has a solution in a graph group, then it has a solution where every x_i is bounded exponentially in the input length (the total length of all words representing the group elements $g_1, \ldots, g_k, g$).
- We then guess the binary encodings of numbers $n_1, \ldots, n_k$ that are bounded by the exponential bound from the previous point and verify in polynomial time the identity $g_1^{n_1} \cdots g_k^{n_k} = g$. The latter problem is an instance of the so-called *compressed word problem* for a graph group. This is the classical word problem, where the input group element is given succinctly by a so-called *straight-line program* (SLP), which is a context-free grammar that produces a single word (here, a word over the group generators and their inverses). An SLP with n

productions in Chomsky normal form can produce a string of length 2^n. Nevertheless, the compressed word problem for each fixed graph group can be solved in polynomial time [33] (see also [32] for more details).

In fact, our proof yields a stronger result: First, it yields an NP procedure for solving knapsack-like equations $h_0 g_1^{x_1} h_1 \cdots h_{k-1} g_k^{x_k} h_k = 1$ where some of the variables $x_1, \ldots, x_k$ are allowed to be identical. We call such an equation an *exponent equation*. Hence, we prove that solvability of exponent equations over a graph group belongs to NP.

Second, we show that the latter result even holds, when the group elements $g_1, \ldots, g_k, h_0, \ldots, h_k$ are given succinctly by SLPs; we speak of *solvability of compressed exponent equations*. This is interesting, since the SLP-encoding of group elements corresponds in the case $G = \mathbb{Z}$ to the binary encoding of integers. Hence, membership in NP for solvability of compressed exponent equations over a graph group generalizes the classical NP-membership for knapsack (over $\mathbb{Z}$) to a much wider class of groups.

Furthermore, we extend the class of groups for which solvability of knapsack (resp. compressed exponent equations) can be checked in NP by proving general transfer results. Our first transfer result states that if H is a finite extension of G and solvability of compressed exponent equations (or knapsack) can be checked in NP for G, then the same holds for H. This provides such algorithms for the large class of *virtually special groups*. These are finite extensions of subgroups of graph groups. Virtually special groups recently played a major role in a spectacular breakthrough in three-dimensional topology, namely the solution of the virtual Haken conjecture [1]. In the course of this development it turned out that the class of virtually special groups is extremely rich: It contains Coxeter groups [20], one-relator groups with torsion [46], fully residually free groups [46], and fundamental groups of hyperbolic 3-manifolds [1].

We also prove transfer results for HNN-extensions and amalgamated products with finite associated (resp. identified) subgroups in the case of the knapsack problem. Such HNN-extensions and amalgamated products play a fundamental role in combinatorial group theory [37]. For example, they appear in Stallings' decomposition of groups with more than one end [43] and in the construction of virtually free groups [10]. Furthermore, they are known to preserve a wide variety of structural and algorithmic properties (see Section 9).

A side product of our proof is that the set of all solutions $(x_1, \ldots, x_k) \in \mathbb{N}^k$ of an exponent equation $g_1^{x_1} \cdots g_k^{x_k} = g$ over a graph group is semilinear,

and a semilinear representation can be produced effectively. This seems to be true for many groups, e.g., for all co-context-free groups [29]. On the other hand, the discrete Heisenberg group $H_3(\mathbb{Z})$ is an example of a group for which solvability of exponent equations is decidable, but the set of all solutions of an exponent equation is not semilinear; it is defined by a single quadratic Diophantine equation [29].

Finally, we complement our upper bounds with a new lower bound: Knapsack and subset sum are both NP-complete for a direct product of two free groups of rank two ($F_2 \times F_2$). This group is the graph group corresponding to a cycle of length four. NP-hardness already holds for the case that the input group elements are specified by words over the generators (for SLP-compressed words, NP-hardness already holds for $\mathbb{Z}$) and the exponent variables are allowed to take values in $\mathbb{Z}$ (instead of $\mathbb{N}$). NP-completeness of subset sum for $F_2 \times F_2$ solves an open problem from [17].

Related work. The knapsack problem is a special case of the more general *rational subset membership problem*. A rational subset of a finitely generated monoid M is the homomorphic image in M of a regular language over the generators of M. In the rational subset membership problem for M the input consists of a rational subset $L \subseteq M$ (specified by a finite automaton) and an element $m \in M$ and it is asked whether $m \in L$. It was shown in [36] that the rational subset membership problem for a graph group G is decidable if and only if the corresponding graph has (i) no induced cycle on four nodes (C4) and (ii) no induced path on four nodes (P4). For the decidable cases, the precise complexity is open.

Knapsack for G can be also viewed as the question, whether a word equation $z_1 z_2 \cdots z_n = 1$, where $z_1, \ldots, z_n$ are variables, together with constraints of the form $\{g^n \mid n \geq 0\}$ for the variables has a solution in G. Such a solution is a mapping $\varphi\colon \{z_1, \ldots, z_n\} \to G$ such that $\varphi(z_1 z_2 \cdots z_n)$ evaluates to 1 in G and all constraints are satisfied. For another class of constraints (so-called normalized rational constraints, which do not cover constraints of the form $\{g^n \mid n \geq 0\}$), solvability of general word equations was shown to be decidable (PSPACE-complete) for graph groups by Diekert and Muscholl [14]. This result was extended in [13] to a transfer theorem for graph products. A graph product is specified by a finite simple graph where every node is labeled with a group. The associated group is obtained from the free product of all vertex groups by allowing elements from adjacent groups to commute. Note that decidability of knapsack is not preserved under graph products: It is not even preserved under direct products (see

the above mentioned results from [29]).

2 Words and Straight-Line Programs

For a word w we denote with $\mathrm{alph}(w)$ the set of symbols occurring in w. The length of the word w is $|w|$.

A *straight-line program*, briefly *SLP*, is basically a context-free grammar that produces exactly one string. To ensure this, the grammar has to be acyclic and deterministic (every variable has a unique production where it occurs on the left-hand side). Formally, an SLP is a tuple $\mathcal{G} = (V, \Sigma, \mathrm{rhs}, S)$, where V is a finite set of *variables* (or *nonterminals*), Σ is the *terminal alphabet*, $S \in V$ is the *start variable*, and rhs maps every variable to a *right-hand side* $\mathrm{rhs}(A) \in (V \cup \Sigma)^*$. We require that there is a linear order $<$ on V such that $B < A$ whenever $B \in N \cap \mathrm{alph}(\mathrm{rhs}(A))$. Every variable $A \in V$ derives to a unique string $\mathrm{val}_{\mathcal{G}}(A)$ by iteratively replacing variables by the corresponding right-hand sides, starting with A. Finally, the string *derived by* $\mathcal{G}$ is $\mathrm{val}(\mathcal{G}) = \mathrm{val}_{\mathcal{G}}(S)$.

Let $\mathcal{G} = (V, \Sigma, \mathrm{rhs}, S)$ be an SLP. The *size* of $\mathcal{G}$ is $|\mathcal{G}| = \sum_{A \in V} |\mathrm{rhs}(A)|$, i.e., the total length of all right-hand sides. A simple induction shows that for every SLP $\mathcal{G}$ of size m one has $|\mathrm{val}(\mathcal{G})| \leq \mathcal{O}(3^{m/3}) \subseteq 2^{\mathcal{O}(m)}$ [8, proof of Lemma 1]. On the other hand, it is straightforward to define an SLP $\mathcal{H}$ of size $2n$ such that $|\mathrm{val}(\mathcal{H})| \geq 2^n$. This justifies to see an SLP $\mathcal{G}$ as a compressed representation of the string $\mathrm{val}(\mathcal{G})$, and exponential compression rates can be achieved in this way. More details on SLPs can be found in the survey [31].

3 Knapsack and Exponent Equations

We assume that the reader has some basic knowledge concerning (finitely generated) groups (see e.g. [37] for further details). Let G be a finitely generated group, and let A be a finite generating set for G. Then, elements of G can be represented by finite words over the alphabet $A^{\pm 1} = A \cup A^{-1}$.

An *exponent equation* over G is an equation of the form

$$v_0 u_1^{x_1} v_1 u_2^{x_2} v_2 \cdots u_n^{x_n} v_n = 1$$

where $u_1, u_2, \ldots, u_n, v_0, v_1, \ldots, v_n \in G$ are group elements that are given by finite words over the alphabet $A^{\pm 1}$ and $x_1, x_2, \ldots, x_n$ are not necessarily distinct variables. Such an exponent equation is *solvable* if there exists a mapping $\sigma \colon \{x_1, \ldots, x_n\} \to \mathbb{N}$ such that $v_0 u_1^{\sigma(x_1)} v_1 u_1^{\sigma(x_2)} v_2 \cdots u_n^{\sigma(x_n)} v_n = 1$

in the group G. *Solvability of exponent equations over G* is the following computational problem:

Input: An exponent equation E over G (where elements of G are specified by words over the group generators and their inverses).

Question: Is E solvable?

The *knapsack problem* for the group G is the restriction of solvability of exponent equations over G to exponent equations of the form $u_1^{x_1} u_2^{x_2} \cdots u_n^{x_n} u^{-1} = 1$ or, equivalently, $u_1^{x_1} u_2^{x_2} \cdots u_n^{x_n} = u$ where the exponent variables $x_1, \ldots, x_n$ have to be pairwise different.

We will also study a compressed version of exponent equations over G, where elements of G are given by SLPs over $A^{\pm 1}$. A *compressed exponent equation* is an exponent equation $v_0 u_1^{x_1} v_1 u_2^{x_2} v_2 \cdots u_n^{x_n} v_n = 1$ where the group elements $u_1, u_2, \ldots, u_n, v_0, v_1, \ldots, v_n \in G$ are given by SLPs over the terminal alphabet $A^{\pm 1}$. The sum of the sizes of these SLPs is the size of the compressed exponent equation.

Let us define *solvability of compressed exponent equations over G* as the following computational problem:

Input: A compressed exponent equation E over G.

Question: Is E solvable?

The *compressed knapsack problem* for G is defined analogously. Note that with this terminology, the classical knapsack problem for binary encoded integers is the compressed knapsack problem for the group $\mathbb{Z}$. The binary encoding of an integer can be easily transformed into an SLP over the alphabet $\{a, a^{-1}\}$ (where a is a generator of $\mathbb{Z}$) and vice versa. Here, the number of bits in the binary encoding and the size of the SLP are linearly related.

It is a simple observation that the decidability and complexity of solvability of (compressed) exponent equations over G as well as the (compressed) knapsack problem for G does not depend on the chosen finite generating set for the group G. Therefore, we do not have to mention the generating set explicitly in these problems.

Remark 1. *Since we are dealing with a group, one might also allow solution mappings $\sigma\colon \{x_1, \ldots, x_n\} \to \mathbb{Z}$ to the integers. But this variant of solvability of (compressed) exponent equations (knapsack, respectively) can be reduced to the above version, where σ maps to $\mathbb{N}$, by simply replacing a power $u_i^{x_i}$ by $u_i^{x_i}(u_i^{-1})^{y_i}$, where y_i is a fresh variable.*

The goal of this paper is to prove the decidability of solvability of exponent equations for so-called graph groups. We actually prove that solvability

of compressed exponent equations for a graph group belongs to NP. Graph groups will be introduced in the next section.

4 Traces and Graph Groups

Let (A, I) be a finite simple graph. In other words, the edge relation $I \subseteq A \times A$ is irreflexive and symmetric. It is also called the *independence relation*, and (A, I) is called an *independence alphabet*. We consider the monoid $\mathbb{M}(A, I) = A^*/\equiv_I$, where $\equiv_I$ is the smallest congruence relation on the free monoid A^* that contains all pairs (ab, ba) with $a, b \in A$ and $(a, b) \in I$. This monoid is called a *trace monoid* or *partially commutative free monoid*; it is cancellative, i.e., $xy = xz$ or $yx = zx$ implies $y = z$. Elements of $\mathbb{M}(A, I)$ are called *Mazurkiewicz traces* or simply *traces*. The trace represented by the word u is denoted by $[u]_I$, or simply u if no confusion can arise. For a language $L \subseteq A^*$ we denote with $[L]_I = \{u \in A^* \mid \exists v \in L : u \equiv_I v\}$ its *partially commutative closure*. The length of the trace $[u]_I$ is $|[u]_I| = |u|$ and its alphabet is $\mathrm{alph}([u]_I) = \mathrm{alph}(u)$. It is easy to see that these definitions do not depend on the concrete word that represents the trace $[u]_I$. For subsets $B, C \subseteq A$ we write BIC for $B \times C \subseteq I$. If $B = \{a\}$ we simply write aIC. For traces s, t we write sIt for $\mathrm{alph}(s)I\mathrm{alph}(t)$. The empty trace $[\varepsilon]_I$ is the identity element of the monoid $\mathbb{M}(A, I)$ and is denoted by 1. A trace t is *connected* if we cannot factorize t as $t = uv$ with $u \neq 1 \neq v$ and uIv.

A trace $t \in \mathbb{M}(A, I)$ can be visualized by its *dependence graph* D_t. To define D_t, choose an arbitrary word $w = a_1 a_2 \cdots a_n$, $a_i \in A$, with $t = [w]_I$ and define $D_t = (\{1, \ldots, n\}, E, \lambda)$, where $E = \{(i, j) \mid i < j, (a_i, a_j) \in D\}$ and $\lambda(i) = a_i$. If we identify isomorphic dependence graphs, then this definition is independent of the chosen word representing t. Moreover, the mapping $t \mapsto D_t$ is injective. As a consequence of the representation of traces by dependence graphs, one obtains Levi's lemma for traces (see e.g. [15, p. 74]), which is one of the fundamental facts in trace theory. The formal statement is as follows.

Lemma 2. *Let $u_1, \ldots, u_m, v_1, \ldots, v_n \in \mathbb{M}(A, I)$. Then*

$$u_1 u_2 \cdots u_m = v_1 v_2 \cdots v_n$$

if and only if there exist $w_{i,j} \in \mathbb{M}(A, I)$ $(1 \leq i \leq m,\ 1 \leq j \leq n)$ such that

- *$u_i = w_{i,1} w_{i,2} \cdots w_{i,n}$ for every $1 \leq i \leq m$,*
- *$v_j = w_{1,j} w_{2,j} \cdots w_{m,j}$ for every $1 \leq j \leq n$, and*

- $(w_{i,j}, w_{k,\ell}) \in I$ *if* $1 \leq i < k \leq m$ *and* $n \geq j > \ell \geq 1$.

The situation in the lemma will be visualized by a diagram of the following kind. The i–th column corresponds to u_i, the j–th row corresponds to v_j, and the intersection of the i–th column and the j–th row represents $w_{i,j}$. Furthermore $w_{i,j}$ and $w_{k,\ell}$ are independent if one of them is left-above the other one.

v_n	$w_{1,n}$	$w_{2,n}$	$w_{3,n}$	$\ldots$	$w_{m,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_3	$w_{1,3}$	$w_{2,3}$	$w_{3,3}$	$\ldots$	$w_{m,3}$
v_2	$w_{1,2}$	$w_{2,2}$	$w_{3,2}$	$\ldots$	$w_{m,2}$
v_1	$w_{1,1}$	$w_{2,1}$	$w_{3,1}$	$\ldots$	$w_{m,1}$
	u_1	u_2	u_3	$\ldots$	u_m

A consequence of Levi's Lemma is that trace monoids are cancellative, i.e., $usv = utv$ implies $s = t$ for all traces $s, t, u, v \in \mathbb{M}(A, I)$.

For a trace $u \in \mathbb{M}(A, I)$ let $\rho(u)$ be the number of prefixes of u. We will use the following statement from [4].

Lemma 3. *Let $t \in \mathbb{M}(A, I)$ be a trace of length n. Then $\rho(t) \in \mathcal{O}(n^\alpha)$, where α is the size of a largest clique of the complementary graph $(A, I)^c = (A, (A \times A) \setminus I)$.*

With an independence alphabet (A, I) we associate the group

$$\mathbb{G}(A, I) = \langle A \mid ab = ba \ ((a, b) \in I)\rangle.$$

Such a group is called a *graph group*, or *right-angled Artin group*[2], or *free partially commutative group*. Here, we use the term graph group. Graph groups received a lot of attention in group theory during the last few years, mainly due to their rich subgroup structure [5, 9, 18], and their relationship to low dimensional topology (via so-called virtually special groups) [1, 20, 46]. We represent elements of $\mathbb{G}(A, I)$ by traces over an extended independence alphabet. For this, let $A^{-1} = \{a^{-1} \mid a \in A\}$ be a disjoint copy of the alphabet A, and let $A^{\pm 1} = A \cup A^{-1}$. We define $(a^{-1})^{-1} = a$ and for a word $w = a_1 a_2 \cdots a_n$ with $a_i \in A^{\pm 1}$ we define $w^{-1} = a_n^{-1} \cdots a_2^{-1} a_1^{-1}$. This defines an involution (without fixed points) on $(A^{\pm 1})^*$. We extend the independence relation I to $A^{\pm 1}$ by $(a^x, b^y) \in I$ for all $(a, b) \in I$ and $x, y \in \{-1, 1\}$. Then,

[2]This term comes from the fact that right-angled Artin groups are exactly the Artin groups corresponding to right-angled Coxeter groups.

there is a canonical surjective morphism $h\colon \mathbb{M}(A^{\pm 1}, I) \to \mathbb{G}(A, I)$ that maps every symbol $a \in A^{\pm 1}$ to the corresponding group element. Of course, h is not injective, but we can easily define a subset $\mathrm{IRR}(A^{\pm 1}, I) \subseteq \mathbb{M}(A^{\pm 1}, I)$ of *irreducible traces* such that h restricted to $\mathrm{IRR}(A^{\pm 1}, I)$ is bijective. The set $\mathrm{IRR}(A^{\pm 1}, I)$ consists of all traces $t \in \mathbb{M}(A^{\pm 1}, I)$ such that t does not contain a factor $[aa^{-1}]_I$ with $a \in A^{\pm 1}$, i.e., there do not exist $u, v \in \mathbb{M}(A^{\pm 1}, I)$ and $a \in A^{\pm 1}$ such that in $\mathbb{M}(A^{\pm 1}, I)$ we have a factorization $t = u[aa^{-1}]_I v$. For every trace t there exists a corresponding *irreducible normal form* that is obtained by removing from t factors $[aa^{-1}]_I$ with $a \in A^{\pm 1}$ as long as possible. It can be shown that this reduction process is terminating (which is trivial since it reduces the length) and confluent (in [30] a more general confluence lemma for graph products of monoids is shown). Hence, the irreducible normal form of t does not depend on the concrete order of reduction steps. For a group element $g \in \mathbb{G}(A, I)$ we denote with $|g|$ the length of the unique trace $t \in \mathrm{IRR}(A^{\pm 1}, I)$ such that $h(t) = g$.

For a trace $t = [u]_I$ ($u \in (A^{\pm 1})^*$) we can define $t^{-1} = [u^{-1}]_I$. This is well-defined, since $u \equiv_I v$ implies $u^{-1} \equiv_I v^{-1}$. The following lemma will be important. See [14, Lemma 23].

Lemma 4. *Let $s, t \in \mathrm{IRR}(A^{\pm 1}, I)$. Then there exist unique factorizations $s = up$, $t = p^{-1}v$ such that $uv \in \mathrm{IRR}(A^{\pm 1}, I)$. Hence, uv is the irreducible normal form of st.*

5 Factorizations of Powers

Based on Levi's lemma we prove in this section a factorization result for powers of a connected trace. We start with the case that we factorize such a power into two factors.

Lemma 5. *Let $u \in \mathbb{M}(A, I) \setminus \{1\}$ be a connected trace. Then, for all $x \in \mathbb{N}$ and all traces y_1, y_2 the following two statements are equivalent:*

(i) $u^x = y_1 y_2$

(ii) There exist $l, k, c \in \mathbb{N}$ and traces s, p such that: $y_1 = u^l s$, $y_2 = p u^k$, $sp = u^c$, $l + k + c = x$, and $c \leq |A|$.

Proof. That (ii) implies (i) is clear. It remains to prove that (i) implies (ii). Assume that $u^x = y_1 y_2$ holds. The case that $x \leq |A|$ is trivial. Hence, assume that $x \geq |A| + 1$. We apply Levi's lemma (Lemma 2) to the identity $u^x = y_1 y_2$:

y_2	$u_{1,2}$	$u_{2,2}$	$u_{3,2}$	$u_{4,2}$	$\cdots$	$u_{x-1,2}$	$u_{x,2}$
y_1	$u_{1,1}$	$u_{2,1}$	$u_{3,1}$	$u_{4,1}$	$\cdots$	$u_{x-1,1}$	$u_{x,1}$
	u	u	u	u	$\cdots$	u	u

Let $A_i = \mathrm{alph}(u_{1,2} \cdots u_{i,2})$. Then $A_i \subseteq A_{i+1}$. If $A_1 = \emptyset$ then $u_{1,2} = 1$ and we can go to Case 2 below. Otherwise, assume that $A_1 \neq \emptyset$. In that case there must exist $1 \leq i \leq |A|$ such that $A_i = A_{i+1}$, which implies $\mathrm{alph}(u_{i+1,2}) \subseteq A_i$. Since $u_{i+1,1} I (u_{1,2} \cdots u_{i,2})$ we also have $u_{i+1,1} I u_{i+1,2}$. Since u is connected, we have $u_{i+1,1} = 1$ or $u_{i+1,2} = 1$. We can therefore distinguish the following two cases:

Case 1. There exists $1 \leq i \leq |A| + 1$ such that $u_{i,1} = 1$. Then $u_{i,2} = u$, which implies $u_{j,1} = 1$ for all $j > i$ (since $u_{i,2} I u_{j,1}$):

y_2	$u_{1,2}$	$u_{2,2}$	$\cdots$	$u_{i-1,2}$	u	u	$\cdots$	u	u
y_1	$u_{1,1}$	$u_{2,1}$	$\cdots$	$u_{i-1,1}$	1	1	$\cdots$	1	1
	u	u	$\cdots$	u	u	u	$\cdots$	u	u

Let $s = u_{1,1} u_{2,1} \cdots u_{i-1,1}$ and $p = u_{1,2} u_{2,2} \cdots u_{i-1,2}$. Thus, $y_1 = u^0 s$, $y_2 = p u^{x-i+1}$ and $sp = u^{i-1}$ with $i - 1 \leq |A|$, and the conclusion of the lemma holds.

Case 2. There exists $1 \leq i \leq |A| + 1$ such that $u_{i,2} = 1$. Then, $u_{j,2} = 1$ for all $j < i$ (since $u_{i,1} = u$ and $u_{j,2} I u_{i,1}$):

y_2	1	1	$\cdots$	1	1	$u_{i+1,2}$	$\cdots$	$u_{x-1,1}$	$u_{x,2}$
y_1	u	u	$\cdots$	u	u	$u_{i+1,1}$	$\cdots$	$u_{x-1,1}$	$u_{x,1}$
	u	u	$\cdots$	u	u	u	$\cdots$	u	u

Let $y'_1 = u_{i+1,1} \cdots u_{x,1}$. Hence, $u^{x-i} = y'_1 y_2$. We can use induction to get factorizations $y'_1 = u^l s$, $y_2 = p u^k$, and $sp = u^c$ with $c \leq |A|$ and $k + l + c = x - i$. Finally, we have $y_1 = u^i y'_1 = u^{i+l} s$, which shows the conclusion of the lemma. □

Now we lift Lemma 5 to an arbitrary number of factors.

Lemma 6. *Let $u \in \mathbb{M}(A, I) \setminus \{1\}$ be a connected trace and $m \in \mathbb{N}$, $m \geq 2$. Then, for all $x \in \mathbb{N}$ and traces $y_1, \ldots, y_m$ the following two statements are equivalent:*

(i) $u^x = y_1 y_2 \cdots y_m$.

(ii) There exist traces $p_{i,j}$ $(1 \leq j < i \leq m)$, s_i $(1 \leq i \leq m)$ and numbers $x_i, c_j \in \mathbb{N}$ $(1 \leq i \leq m,\ 1 \leq j \leq m-1)$ such that:

- $y_i = (\prod_{j=1}^{i-1} p_{i,j})u^{x_i}s_i$ *for all* $1 \leq i \leq m$,
- $p_{i,j} I p_{k,l}$ *if* $j < l < k < i$ *and* $p_{i,j} I (u^{x_k} s_k)$ *if* $j < k < i$
- $s_m = 1$ *and for all* $1 \leq j < m$, $s_j \prod_{i=j+1}^{m} p_{i,j} = u^{c_j}$
- $c_j \leq |A|$ *for all* $1 \leq j \leq m-1$,
- $x = \sum_{i=1}^{m} x_i + \sum_{i=1}^{m-1} c_i$.

Proof. Let us first show that (ii) implies (i). Assume that (ii) holds. Then we get

$$y_1 y_2 \cdots y_m = \prod_{i=1}^{m} \Big((\prod_{j=1}^{i-1} p_{i,j}) u^{x_i} s_i \Big).$$

The independencies $p_{i,j} I p_{k,l}$ for $j < l < k < i$ and $p_{i,j} I (u^{x_k} s_k)$ for $j < k < i$ yield

$$\begin{aligned} & \prod_{i=1}^{m} \Big((\prod_{j=1}^{i-1} p_{i,j}) u^{x_i} s_i \Big) \\ = \;& u^{x_1} s_1 p_{2,1} \cdots p_{m,1} u^{x_2} s_2 p_{3,2} \cdots p_{m,2} u^{x_3} s_3 \cdots u^{x_{m-1}} s_{m-1} p_{m,m-1} u^{x_m} s_m \\ = \;& u^{x_1} u^{c_1} u^{x_2} u^{c_2} u^{x_3} \cdots u^{c_{m-1}} u^{x_m} = u^x. \end{aligned}$$

We now prove that (i) implies (ii) by induction on m. So, assume that $u^x = y_1 y_2 \cdots y_m$. The case $m = 2$ follows directly from Lemma 5. Now assume that $m \geq 3$. By Lemma 5 there exist factorizations $y_1 = u^{x_1} s_1$, $y_2 \cdots y_m = p_1 u^{x'}$, and $s_1 p_1 = u^{c_1}$ with $c_1 \leq |A|$ and $x_1 + x' + c_1 = x$. Levi's lemma applied to $y_2 \cdots y_m = p_1 u^{x'}$ gives the following diagram:

y_m	$p_{m,1}$	y'_m					
$\vdots$	$\vdots$	$\vdots$					
y_3	$p_{3,1}$	y'_3					
y_2	$p_{2,1}$	y'_2					
	p_1	u	u	u	$\ldots$	u	u

There exist y'_i with $y_i = p_{i,1} y'_i$ $(2 \leq i \leq m)$, $y'_2 \cdots y'_m = u^{x'}$, and $y'_j I p_{i,1}$ for $j < i$. By induction on m we get factorizations

$$y'_i = \prod_{j=2}^{i-1} p_{i,j} u^{x_i} s_i$$

for $2 \leq i \leq m$ such that for all $2 \leq j < i \leq m$:

- $p_{i,j} I p_{k,l}$ if $j < l < k < i$ and $p_{i,j} I (u^{x_k} s_k)$ if $j < k < i$,
- $s_m = 1$ and for all $2 \leq j < m$, $s_j \prod_{i=j+1}^{m} p_{i,j} = u^{c_j}$ for some $c_j \leq |A|$,
- $x' = \sum_{i=2}^{m} x_i + \sum_{i=2}^{m-1} c_i$.

Since $y'_j I p_{i,1}$ for $j < i$ we get $p_{i,1} I p_{j,k}$ for $1 < k < j < i$ and $p_{i,1} I u^{x_j} s_j$ for $1 < j < i$. Finally, we have

$$s_1 \prod_{i=2}^{m} p_{i,1} = s_1 p_1 = u^{c_1}$$

and

$$x = x_1 + c_1 + x' = x_1 + c_1 + \sum_{i=2}^{m} x_i + \sum_{i=2}^{m-1} c_i = \sum_{i=1}^{m} x_i + \sum_{i=1}^{m-1} c_i.$$

This proves the lemma. □

Remark 7. *In Section 8 we will apply Lemma 6 in order to replace an equation $u^x = y_1 y_2 \cdots y_m$ (where $x, y_1, \ldots, y_m$ are variables and u is a concrete connected trace) by an equivalent disjunction. Note that the length of all factors $p_{i,j}$ and s_i above is bounded by $|A| \cdot |u|$. Hence, one can guess these traces as well as the numbers $c_j \leq |A|$ (the guess results in a big disjunction). We can also guess which of the numbers x_i are zero and which are greater than zero. After these guesses we can verify the independences $p_{i,j} I p_{k,l}$ $(j < l < k < i)$ and $p_{i,j} I (u^{x_k} s_k)$ $(j < k < i)$, and the identities $s_m = 1$, $s_j \prod_{i=j+1}^{m} p_{i,j} = u^{c_j}$ $(1 \leq j < m)$. If one of them does not hold, the specific guess does not contribute to the disjunction. In this way, we can replace the equation $u^x = y_1 y_2 \cdots y_m$ by a big disjunction of formulas of the form*

$$\exists x_i > 0 \ (i \in K) : x = \sum_{i \in K}^{m} x_i + c \wedge \bigwedge_{i \in K} y_i = p_i u^{x_i} s_i \wedge \bigwedge_{i \in [1,m] \setminus K} y_i = p_i s_i,$$

where $K \subseteq [1, m]$, $c \leq |A| \cdot (m-1)$ and the p_i, s_i are concrete traces of length at most $|A| \cdot (m-1) \cdot |u|$. The number of disjuncts in the disjunction will not be important for our purpose.

6 Automata for Partially Commutative Closures

In this section, we present several automata constructions that are well-known from the theory of recognizable trace languages [11, Chapter 2]. For our purpose we need upper bounds on the size (the size of an automaton is its number of states) of the constructed automata. In our specific situation we can obtain better bounds than those obtained from the known constructions. Therefore, we present the constructions in detail.

Let us fix an independence alphabet (A, I) and let $\mathcal{A} = (Q, A, \Delta, q_0, F)$ be a nondeterministic finite automaton (NFA) over the alphabet A, where $\Delta \subseteq Q \times A \times Q$ is the transition relation, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final states. Then, $\mathcal{A}$ is an I-diamond NFA if for all $(a, b) \in I$ and all transitions $(p, a, q), (q, b, r) \in \Delta$ there exists a state q' such that $(p, b, q'), (q', a, r) \in \Delta$. For an I-diamond automaton we have $L(\mathcal{A}) = [L(\mathcal{A})]_I$. The NFA $\mathcal{A}$ is *memorizing* if (i) every state is accessible from the initial state q_0 and (ii) there is a mapping $\alpha\colon Q \to 2^A$ such that for every word $w \in A^*$, if $q_0 \xrightarrow{w}_{\mathcal{A}} q$, then $\alpha(q) = \mathrm{alph}(w)$.

Lemma 8. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be I-diamond NFA and let n_i be the number of states of $\mathcal{A}_i$. Assume that $\mathcal{A}_2$ is memorizing. Then there exists an I-diamond NFA for $[L(\mathcal{A}_1)L(\mathcal{A}_2)]_I$ with $n_1 \cdot n_2$ many states.*

Proof. Let $\mathcal{A}_i = (Q_i, A, \Delta_i, q_{0,i}, F_i)$ for $i \in \{1, 2\}$. Let $\alpha_2\colon Q_2 \to 2^A$ be the map witnessing the fact that $\mathcal{A}_2$ is memorizing. Then, let

$$\mathcal{A} = (Q_1 \times Q_2, A, \Delta, \langle q_{0,1}, q_{0,2}\rangle, F_1 \times F_2),$$

where

$$\begin{aligned}\Delta \;=\; & \{(\langle p_1, p_2\rangle, a, \langle q_1, p_2\rangle) \mid (p_1, a, q_1) \in \Delta_1, aI\alpha_2(p_2)\} \cup \\ & \{(\langle p_1, p_2\rangle, a, \langle p_1, q_2\rangle) \mid (p_2, a, q_2) \in \Delta_2\}.\end{aligned}$$

This indeed defines an I-diamond NFA.

We show that the following two statements are equivalent for all $w \in A^*$, $p_1 \in Q_1$, and $p_2 \in Q_2$:

(i) $\langle q_{0,1}, q_{0,2}\rangle \xrightarrow{w}_{\mathcal{A}} \langle p_1, p_2\rangle$

(ii) There are $w_1, w_2 \in A^*$ such that $w \equiv_I w_1 w_2$, $q_{0,1} \xrightarrow{w_1}_{\mathcal{A}_1} p_1$, and $q_{0,2} \xrightarrow{w_2}_{\mathcal{A}_2} p_2$.

This clearly implies that $L(\mathcal{A}) = [L(\mathcal{A}_1)L(\mathcal{A}_2)]_I$.

Let us first prove that (i) implies (ii). The case $w = \varepsilon$ is clear. Hence, let $w = w'a$. Then there exist $p_1' \in Q_1$, $p_2' \in Q_2$ such that

$$\langle q_{0,1}, q_{0,2}\rangle \xrightarrow{w'}_{\mathcal{A}} \langle p_1', p_2'\rangle \xrightarrow{a}_{\mathcal{A}} \langle p_1, p_2\rangle.$$

By induction, there exists a factorization $w' \equiv_I w_1'w_2'$ such that $q_{0,1} \xrightarrow{w_1'}_{\mathcal{A}_1} p_1'$ and $q_{0,2} \xrightarrow{w_2'}_{\mathcal{A}_2} p_2'$. Note that $\mathrm{alph}(w_2') = \alpha_2(p_2')$. There are two cases:

Case 1. $p_1' \xrightarrow{a}_{\mathcal{A}_1} p_1$, $p_2 = p_2'$, and $aI\alpha_2(p_2')$. Thus, aIw_2'. We get $w = w'a \equiv_I w_1'w_2'a \equiv_I (w_1'a)w_2'$. Let $w_1 = w_1'a$ and $w_2 = w_2'$. We get $q_{0,1} \xrightarrow{w_1}_{\mathcal{A}_1} p_1$ and $q_{0,2} \xrightarrow{w_2}_{\mathcal{A}_2} p_2$.

Case 2. $p_2' \xrightarrow{a}_{\mathcal{A}_2} p_2$ and $p_1 = p_1'$. Let $w_1 = w_1'$ and $w_2 = w_2'a$. Thus, $w = w'a \equiv_I w_1'w_2'a = w_1w_2$. Moreover, we have $q_{0,1} \xrightarrow{w_1}_{\mathcal{A}_1} p_1$ and $q_{0,2} \xrightarrow{w_2}_{\mathcal{A}_2} p_2$.

Let us now prove that (ii) implies (i). Assume that $w \equiv_I w_1w_2$, $q_{0,1} \xrightarrow{w_1}_{\mathcal{A}_1} p_1$, and $q_{0,2} \xrightarrow{w_2}_{\mathcal{A}_2} p_2$. We have to show that $\langle q_{0,1}, q_{0,2}\rangle \xrightarrow{w}_{\mathcal{A}} \langle p_1, p_2\rangle$. But since $\mathcal{A}$ is an I-diamond NFA, it suffices to show that $\langle q_{0,1}, q_{0,2}\rangle \xrightarrow{w_1w_2}_{\mathcal{A}} \langle p_1, p_2\rangle$, which follows directly from the assumption and the definition of $\mathcal{A}$ (note that $\alpha_2(q_{0,2}) = \emptyset$). This concludes the proof of the lemma. □

In general, for a regular language $L \subseteq A^*$, the partially commutative closure $[L]_I$ is not regular. For instance, if $A = \{a, b\}$ and aIb, then $[(ab)^*]_I$ consists of all words with the same number of a's as b's. On the other hand, it is well known that if u is a connected trace, then $[u^*]_I$ is regular (in fact, there is a more general result, known as Ochmanski's theorem [11, Section 2.3]). For our purpose we need an upper on the size of an I-diamond NFA for $[u^*]_I$ (with u connected). Recall that $\rho(u)$ is the number of different prefixes of the trace u.

Lemma 9. *Let $u \in \mathbb{M}(A, I) \setminus \{1\}$ be connected. There is a memorizing I-diamond NFA for $[u^*]_I$ of size $2 \cdot \rho(u)^{|A|}$.*

Proof. The following construction can be found in [40, Proposition 5] for the more general case of the partially commutative closure of a so-called loop-connected automaton. We present the construction in our simplified situation, since the NFA gets slightly smaller.

We first define a non-memorizing I-diamond NFA $\mathcal{A}$ for $[u^*]_I$ of size $\rho(u)^{|A|}$. Then, we show that by adding an additional bit to all states, we can get a memorizing I-diamond NFA $\mathcal{A}$ for $[u^*]_I$ of size $2 \cdot \rho(u)^{|A|}$. The idea for the construction of $\mathcal{A}$ is implicitly contained in the proof of Lemma 5:

Assume that the automaton wants to read a word of the form u^x and a prefix y_1 is already read. Then y_1 must be of the form $u^k s$, where s is a prefix of u^c for some $c \leq |A|$. The prefix s must be of the form $u_1 u_2 \cdots u_c$ such that if $u = u_i v_i$, then $v_i I u_j$ if $i < j$. The state of the NFA stores the tuple $(u_1, u_2, \ldots, u_c)$.

Define $\mathcal{A} = (Q, A, \Delta, q_0, F)$, where Q is the set of all tuples $(u_1, u_2, \ldots, u_c)$ of traces such that there exist $v_1, \ldots, v_c \in \mathbb{M}(A, I)$ with $u = u_i v_i$ (since $\mathbb{M}(A, I)$ is cancellative, the v_i are uniquely determined by the u_i), $u_i \neq 1$, $v_i \neq 1$, and $v_i I u_j$ if $i < j$. Note that we must have $c \leq |A|$: If $c > |A|$, then there exist $i \leq |A|$ such that $\mathrm{alph}(v_1 \cdots v_i) = \mathrm{alph}(v_1 \cdots v_{i+1})$. Hence, $\mathrm{alph}(v_{i+1}) \subseteq \mathrm{alph}(v_1 \cdots v_i)$. Since $(v_1 \cdots v_i) I u_{i+1}$ we get $u_{i+1} I v_{i+1}$ which contradicts $u_{i+1} \neq 1 \neq v_{i+1}$ and the fact that u is connected.

Since $u_i \neq 1$ for all i, we can encode a state $(u_1, u_2, \ldots, u_c) \in Q$ by the tuple $(u_1, u_2, \ldots, u_c, 1, \ldots, 1)$ of length $|A|$. This implies that the number of states of $\mathcal{A}$ is bounded by $\rho(u)^{|A|}$. Note that if $|u| = 1$, then the empty tuple () is the only state.

The transitions of $\mathcal{A}$ are defined as follows, where $(u_1, u_2, \ldots, u_c) \in Q$:

(a) $() \xrightarrow{a}_{\mathcal{A}} ()$ if $u = a \in A$,

(b) $(u_1, u_2, \ldots, u_c) \xrightarrow{a}_{\mathcal{A}} (u_2, \ldots, u_c)$ if $c > 0$, $u_1 a = u$ and $aI(u_2 \cdots u_c)$,

(c) $(u_1, u_2, \ldots, u_c) \xrightarrow{a}_{\mathcal{A}} (u_1, \ldots, u_i, a, u_{i+1}, \ldots, u_c)$ if $aI(u_{i+1} \cdots u_c)$ and $(u_1, \ldots, u_i, a, u_{i+1}, \ldots, u_c) \in Q$,

(d) $(u_1, u_2, \ldots, u_c) \xrightarrow{a}_{\mathcal{A}} (u_1, \ldots, u_{i-1}, u_i a, u_{i+1}, \ldots, u_c)$ if $aI(u_{i+1} \cdots u_c)$ and $(u_1, \ldots, u_{i-1}, u_i a, u_{i+1}, \ldots, u_c) \in Q$.

The initial state as well as the final state is the empty tuple (). It is easy to check that this is indeed an I-diamond NFA.

We claim that for every state $(u_1, \ldots, u_c) \in Q$ and every $w \in A^*$ the following two statements are equivalent (which shows that $L(\mathcal{A}) = [u^*]_I$):

(i) $() \xrightarrow{w}_{\mathcal{A}} (u_1, \ldots, u_c)$

(ii) $w \equiv_I u^k u_1 \cdots u_c$ for some $k \geq 0$

Let us first show by induction on $|w|$ that (i) implies (ii). The case $w = \varepsilon$ is clear. So, assume that $w = w'a$. There must exist a state $(u'_1, \ldots, u'_d) \in Q$ such that

$$() \xrightarrow{w'}_{\mathcal{A}} (u'_1, \ldots, u'_d) \xrightarrow{a}_{\mathcal{A}} (u_1, \ldots, u_c).$$

By induction, we get $w' \equiv_I u^\ell u'_1 \cdots u'_d$ for some $\ell \geq 0$. The definition of the transitions of $\mathcal{A}$ implies that $w = w'a \equiv_I u^\ell u'_1 \cdots u'_d a \equiv_I u^k u_1 \cdots u_c$, where $k \in \{\ell, \ell+1\}$.

For the direction from (ii) to (i) assume that $w \equiv_I u^k u_1 \cdots u_c$ for some $k \geq 0$. We have to show that $() \xrightarrow{w}_{\mathcal{A}} (u_1, \ldots, u_c)$. Since $\mathcal{A}$ is an I-diamond NFA, it suffices to show that $() \xrightarrow{u^k u_1 \cdots u_c}_{\mathcal{A}} (u_1, \ldots, u_c)$. But this follows directly from the definition of $\mathcal{A}$.

To make $\mathcal{A}$ memorizing, we first keep only those states that are accessible from the initial state $()$. Then, we add an extra bit to every state that indicates whether we have already seen a completed occurrence of u. Thus, the new set of states is $Q \times \{0,1\}$, the initial state is the pair $((),0)$, and the final states are $((),0)$ and $((),1)$. The transitions operate on the Q-component as for $\mathcal{A}$. The $\{0,1\}$-component is copied except for a transition $(q,a,p) \in \Delta$ of type (b). This transition gives us the transitions $((q,0),a,(p,1))$ and $((q,1),a,(p,1))$. Then, we can define the α-mapping by

$$\alpha((u_1,\ldots,u_c),i) = \bigcup_{j=1}^{c} \mathrm{alph}(u_j) \cup \mathrm{alph}(u^i).$$

The resulting NFA is still an I-diamond NFA. □

A direct consequence of Lemma 8 and 9 is:

Lemma 10. *Let $p,u,s \in \mathbb{M}(A,I)$ with $u \neq 1$ connected. There is an NFA for $[pu^*s]_I$ of size $2 \cdot \rho(p) \cdot \rho(s) \cdot \rho(u)^{|A|}$.*

Proof. We first construct an I-diamond NFA for p (which is identified here with the set of words $\{u \in A^* \mid p = [u]_I\}$) with $\rho(p)$ many states by taking the set of all prefixes of p as states. Then, we construct a memorizing I-diamond NFA for $[u^*]_I$ with $2 \cdot \rho(u)^{|A|}$ states using Lemma 9. By Lemma 8 we get an I-diamond automaton for $[pu^*]_I$ with $2 \cdot \rho(p) \cdot \rho(u)^{|A|}$ many states. Finally, we construct an I-diamond NFA for s with $\rho(s)$ many states by taking the set of all prefixes of s as states. This NFA is also memorizing. Hence, we can apply Lemma 8 to get an NFA for $[pu^*s]_I$ with $2 \cdot \rho(p) \cdot \rho(s) \cdot \rho(u)^{|A|}$ many states. □

The main lemma from this section that will be needed later is:

Lemma 11. *Let $p,q,u,v,s,t \in \mathbb{M}(A,I)$ with $u \neq 1$ and $v \neq 1$ connected. Let $m = \max\{\rho(p),\rho(q),\rho(s),\rho(t)\}$ and $n = \max\{\rho(u),\rho(v)\}$. Then the set*

$$L(p,u,s,q,v,t) := \{(x,y) \in \mathbb{N} \times \mathbb{N} \mid pu^x s = qv^y t\}$$

is semilinear and is a union of $\mathcal{O}(m^8 \cdot n^{4|A|})$ *many linear sets of the form* $\{(a + bz, c + dz) \mid z \in \mathbb{N}\}$ *with* $a, b, c, d \in \mathcal{O}(m^8 \cdot n^{4|A|})$.

Proof. By Lemma 9 there exists an NFA for $[pu^*s]_I$ of size

$$k = 2 \cdot \rho(p) \cdot \rho(s) \cdot \rho(u)^{|A|} \leq 2 \cdot m^2 \cdot n^{|A|}$$

and an NFA for $[qv^*t]_I$ of size

$$\ell = 2 \cdot \rho(q) \cdot \rho(t) \cdot \rho(v)^{|A|} \leq 2 \cdot m^2 \cdot n^{|A|}.$$

Then, we obtain an NFA $\mathcal{A}$ for $L = [pu^*s]_I \cap [qv^*t]_I$ with $k \cdot \ell$ states. We are only interested in the length of words from L. Hence, we replace in $\mathcal{A}$ every transition label by the symbol a. The resulting NFA $\mathcal{B}$ is defined over a unary alphabet. Let $P = \{n \mid a^n \in L(\mathcal{B})\}$. By [44, Theorem 1], the set P can be written as a union

$$P = \bigcup_{i=1}^{r} \{b_i + c_i \cdot z \mid z \in \mathbb{N}\}$$

with $r \in \mathcal{O}(k^2\ell^2) \subseteq \mathcal{O}(m^8 \cdot n^{4|A|})$ and $b_i, c_i \in \mathcal{O}(k^2\ell^2) \subseteq \mathcal{O}(m^8 \cdot n^{4|A|})$. For every $1 \leq i \leq r$ and $z \in \mathbb{N}$ there must exist a pair $(x, y) \in \mathbb{N} \times \mathbb{N}$ such that

$$b_i + c_i \cdot z = |ps| + |u| \cdot x = |qt| + |v| \cdot y.$$

In particular, $b_i \geq |ps|$, $b_i \geq |qt|$, $|u|$ divides $b_i - |ps|$ and c_i, and $|v|$ divides $b_i - |qt|$ and c_i. We get:

$$L(p, u, s, q, v, t) = \bigcup_{i=1}^{r} \left\{ \left(\frac{b_i - |ps|}{|u|} + \frac{c_i}{|u|} \cdot z, \frac{b_i - |qt|}{|v|} + \frac{c_i}{|v|} \cdot z \right) \;\middle|\; z \in \mathbb{N} \right\}$$

This shows the lemma. □

7 Linear Diophantine Equations

We will also need a bound on the norm of a smallest vector in a certain kind of semilinear sets. We will easily obtain this bound from a result by zur Gathen and Sieveking [45].

Lemma 12. *Let* $A \in \mathbb{Z}^{n \times m}$, $\overline{a} \in \mathbb{Z}^n$, $C \in \mathbb{N}^{k \times m}$, $\overline{c} \in \mathbb{N}^k$. *Let* β *be an upper bound for the absolute value of all entries in* A, $\overline{a}$, C, $\overline{c}$. *The set*

$$L = \{C\overline{z} + \overline{c} \mid \overline{z} \in \mathbb{N}^m, A\overline{z} = \overline{a}\} \subseteq \mathbb{N}^k \tag{1}$$

is semilinear. Moreover, if $L \neq \emptyset$ *then* L *contains a vector with all entries bounded by* $\beta + n! \cdot m \cdot (m + 1) \cdot \beta^{n+1}$.

Proof. Semilinearity of L is clear since the set is Presburger-definable. For the size bound, we use a result by zur Gathen and Sieveking [45] to bound the size of a smallest positive solution of the system $A\overline{z} = \overline{a}$. Let $A \in \mathbb{Z}^{n\times m}$, $B \in \mathbb{Z}^{p\times m}$, $\overline{a} \in \mathbb{Z}^{n\times 1}$, $\overline{b} \in \mathbb{Z}^{p\times 1}$. Let $r = \text{rank}(A)$, and $s = \text{rank}\begin{pmatrix} A \\ B \end{pmatrix}$. Let M be an upper bound on the absolute values of all $(s-1)\times(s-1)$- or $(s\times s)$-subdeterminants of the $(n+p)\times(m+1)$-matrix $\begin{pmatrix} A & \overline{a} \\ B & \overline{b} \end{pmatrix}$, which are formed with at least r rows from the matrix $(A\ \overline{a})$. Then by the main result of [45], the system $A\overline{z} = \overline{a}$, $B\overline{z} \geq \overline{b}$ has an integer solution if and only if it has an integer solution $\overline{z}$ such that the absolute value of every entry of $\overline{z}$ is bounded by $(m+1)M$.

In our situation, we set $p = m$, B is the m-dimensional identity matrix, and $\overline{b}$ is the vector with all entries equal to zero (then $B\overline{z} \geq \overline{b}$ expresses that all entries of z are positive). Since $\begin{pmatrix} A \\ B \end{pmatrix}$ is an $(n+m)\times m$-matrix we get $s = \text{rank}\begin{pmatrix} A \\ B \end{pmatrix} \leq m$. We claim that the absolute values of all $(s\times s)$-subdeterminants (and also all $(s-1)\times(s-1)$-subdeterminants) of the matrix $\begin{pmatrix} A & \overline{a} \\ B & \overline{b} \end{pmatrix}$ are bounded by $n!\cdot\beta^n$. To see this, select s rows and s columns from $\begin{pmatrix} A & \overline{a} \\ B & \overline{b} \end{pmatrix}$ and consider the resulting submatrix D. Recall Leibniz' formula for the determinant (where S_s is the set of all permutations of $\{1,\ldots,s\}$):

$$\det(D) = \sum_{\sigma\in S_s} \text{sgn}(\sigma) \prod_{i=1}^{s} D[i,\sigma(i)].$$

Assume that the rows $1,\ldots,s_1$ ($s_1 \leq n$) of D are from the $n\times(m+1)$-submatrix $(A,\overline{a})$. The remaining ($s_2 := s - s_1$ many) rows $s_1+1,\ldots,s$ of D are from $(B,\overline{b})$. If one of the rows $s_1+1,\ldots,s$ of D only contains zeros, then $\det(D) = 0$. Otherwise, since B is the identity matrix and $\overline{b}$ is the zero vector, each of the rows $s_1+1,\ldots,s$ contains a unique 1; all other entries are zero. That means that every permutation $\sigma \in S_s$ that gives a non-zero contribution to $\det(D)$ must take fixed values on $s_1+1,\ldots,s$. For the values of σ on the rows $1,\ldots,s_1$, only $s_1 \leq n$ many values remain. Hence, at most $s_1! \leq n!$ many permutations contribute a non-zero value to $\det(D)$. Moreover, every such contribution is bounded by $\beta^{s_1} \leq \beta^n$, which gives the bound $n!\cdot\beta^n$ on $\det(D)$. It follows that if $A\overline{z} = \overline{a}$ has a positive solution, then it has a positive solution where every entry is bounded by

$(m+1) \cdot n! \cdot \beta^n$.

By substituting every entry of $\overline{z}$ by $(m+1)n! \cdot \beta^n$ in $C\overline{z} + \overline{c}$, it follows that if the set L in (1) is non-empty, then it contains a vector with all entries bounded by $\beta + n! \cdot m \cdot (m+1) \cdot \beta^{n+1}$. □

8 Exponent Equations in Graph Groups

The aim of this section is to prove the following two statements, where G is a fixed graph group:

- The set of solutions of an exponent equation over G is (effectively) semilinear.
- Solvability of compressed exponent equations over G belongs to NP.

We start with some definitions. As usual, we fix an independence alphabet (A, I). In the following we will consider reduction rules on sequences of traces. For better readability we separate the consecutive traces in such a sequence by commas. Let $u_1, u_2, \ldots, u_n \in \mathrm{IRR}(A^{\pm 1}, I)$ be irreducible traces. The sequence $u_1, u_2, \ldots, u_n$ is *I-freely reducible* if the sequence $u_1, u_2, \ldots, u_n$ can be reduced to the empty sequence ε by the following rules:

- $u_i, u_j \to u_j, u_i$ if $u_i I u_j$
- $u_i, u_j \to \varepsilon$ if $u_i = u_j^{-1}$ in $\mathbb{G}(A, I)$
- $u_i \to \varepsilon$ if $u_i = \varepsilon$.

A concrete sequence of these rewrite steps leading to the empty sequence is a *reduction* of the sequence $u_1, u_2, \ldots, u_n$. Such a reduction can be seen as a witness for the fact that $u_1 u_2 \cdots u_n = 1$ in $\mathbb{G}(A, I)$. On the other hand, $u_1 u_2 \cdots u_n = 1$ does not necessarily imply that $u_1, u_2, \ldots, u_n$ has a reduction. For instance, the sequence a^{-1}, ab, b^{-1} has no reduction. But we can show that every sequence which multiplies to 1 in G can be refined (by factorizing the elements of the sequence) such that the resulting refined sequence has a reduction. For getting an NP-algorithm, it is important to bound the length of the refined sequence exponentially in the length of the initial sequence.

Lemma 13. *Let $n \geq 2$ and $u_1, u_2, \ldots, u_n \in \mathrm{IRR}(A^{\pm 1}, I)$. If $u_1 u_2 \cdots u_n = 1$ in $\mathbb{G}(A, I)$, then there exist factorizations $u_i = u_{i,1} \cdots u_{i,k_i}$ such that the sequence*

$$u_{1,1}, \ldots, u_{1,k_1},\ u_{2,1}, \ldots, u_{2,k_2},\ \ldots, u_{n,1}, \ldots, u_{n,k_n}$$

is I-freely reducible. Moreover, $\sum_{i=1}^{n} k_i \leq 2^n - 2$.

Proof. We prove the lemma by induction on n. The case $n = 2$ is trivial (we must have $u_2 = u_1^{-1}$). If $n \geq 3$ then by Lemma 4 we can factorize u_1 and u_2 as $u_1 = ps$ and $u_2 = s^{-1}t$ such that $v := pt$ is irreducible. Hence, $vu_3 \cdots u_n = 1$ in $\mathbb{G}(A, I)$. By induction, we obtain factorizations $pt = v = v_1 \cdots v_k$ and $u_i = v_{i,1} \cdots v_{i,k_i}$ $(3 \leq i \leq n)$ such that the sequence

$$v_1, \ldots, v_k,\ v_{3,1}, \ldots, v_{3,k_3}, \ldots, v_{n,1}, \ldots, v_{n,k_n} \tag{2}$$

is I-freely reducible. Moreover,

$$k + \sum_{i=3}^{n} k_i \leq 2^{n-1} - 2.$$

By applying Levi's lemma to the identity $pt = v_1 v_2 \cdots v_k$, we obtain factorizations $v_i = u_{i,1}u_{i,2}$ such that $p = u_{1,1} \cdots u_{k,1}$, $t = u_{1,2} \cdots u_{k,2}$, and $u_{i,2} I u_{j,1}$ for $1 \leq i < j \leq k$.

Fix a concrete reduction of the sequence (2). We now consider the following sequence

$$u_{1,1}, \ldots, u_{k,1}, s,\ s^{-1}, u_{1,2}, \ldots, u_{k,2},\ \tilde{v}_{3,1}, \ldots, \tilde{v}_{3,k_3}, \ldots, \tilde{v}_{n,1}, \ldots, \tilde{v}_{n,k_n}, \tag{3}$$

where the subsequence $\tilde{v}_{i,j}$ is $u_{l,2}^{-1}, u_{l,1}^{-1}$ if $v_{i,j}$ cancels against v_l in our fixed reduction of (2) (which, in particular implies that $v_{i,j} = v_l^{-1} = u_{l,2}^{-1}u_{l,1}^{-1}$). Otherwise (i.e., if $v_{i,j}$ does not cancel against any v_l in our fixed reduction), we set $\tilde{v}_{i,j} = v_{i,j}$.

Note that $u_{1,1} \cdots u_{k,1}s = ps = u_1$, $s^{-1}u_{1,2} \cdots u_{k,2} = s^{-1}t = u_2$ and the concatenation of all traces in $\tilde{v}_{i,1}, \ldots, \tilde{v}_{i,k_i}$ is u_i for $3 \leq i \leq n$. Hence, it remains to show that the sequence (3) is I-freely reducible. First of all, $u_{1,1}, \ldots, u_{k,1}, s, s^{-1}, u_{1,2}, \ldots, u_{k,2}$ reduces to $u_{1,1}, \ldots, u_{k,1}, u_{1,2}, \ldots, u_{k,2}$, which can be rearranged to $u_{1,1}, u_{1,2}, u_{2,1}, u_{2,2}, \ldots, u_{k,1}, u_{k,2}$ using the fact that $u_{i,2} I u_{j,1}$ for $1 \leq i < j \leq k$. Finally, the sequence

$$u_{1,1}u_{1,2}, u_{2,1}u_{2,2}, \ldots, u_{k,1}u_{k,2}, \tilde{v}_{3,1}, \ldots, \tilde{v}_{3,k_3}, \ldots, \tilde{v}_{n,1}, \ldots, \tilde{v}_{n,k_n}$$

is I-freely reducible. The definition of $\tilde{v}_{i,j}$ allows to basically apply the fixed reduction of (2) to this sequence.

The number of traces in the sequence (3) can be estimated as

$$2k + 2 + 2 \cdot \sum_{i=3}^{n} k_i \leq 2 \cdot (2^{n-1} - 2) + 2 = 2^n - 2.$$

This concludes the proof of the lemma. □

We now come to the main technical result of this paper. Let $\alpha \leq |A|$ be the size of a largest clique of the complementary graph $(A, I)^c = (A, (A \times A) \setminus I)$.

Theorem 14. *Let $u_1, u_2, \ldots, u_n \in \mathbb{G}(A, I) \setminus \{1\}$, $v_0, v_1, \ldots, v_n \in \mathbb{G}(A, I)$ and let $x_1, \ldots, x_n$ be variables (we may have $x_i = x_j$ for $i \neq j$) ranging over $\mathbb{N}$. Then, the set of solutions of the exponent equation*

$$v_0 u_1^{x_1} v_1 u_2^{x_2} v_2 \cdots u_n^{x_n} v_n = 1$$

is semilinear. Moreover, if there is a solution, then there is a solution with $x_i \in \mathcal{O}((\alpha n)! \cdot 2^{2\alpha^2 n(n+3)} \cdot \mu^{8\alpha(n+1)} \cdot \nu^{8\alpha|A|(n+1)})$, where

- *$\mu \in \mathcal{O}(|A|^\alpha \cdot 2^{2\alpha^2 n} \cdot \lambda^\alpha)$,*
- *$\nu \in \mathcal{O}(\lambda^\alpha)$, and*
- *$\lambda = \max\{|u_1|, |u_2|, \ldots, |u_n|, |v_0|, |v_1|, \ldots, |v_n|\}$.*

Proof. Let us choose irreducible traces for $u_1, u_2, \ldots, u_n, v_0, v_1, \ldots, v_n$; we denote these traces with the same letters as the group elements. A trace u is called *cyclically reduced* if there do not exist $a \in A^{\pm 1}$ and v such that $u = ava^{-1}$. For every trace u there exist unique traces p, w such that $u = pwp^{-1}$ and w is cyclically reduced (since the reduction relation $a^{-1}xa \to x$ is terminating and confluent [12, Lemma 16]). These traces p and w can be computed in polynomial time. Note that for a cyclically reduced irreducible trace w, every power w^n is irreducible. Let $u_i = p_i w_i p_i^{-1}$ with w_i cyclically reduced. By replacing every $u_i^{x_i}$ by $p_i w_i^{x_i} p_i^{-1}$, we can assume that all u_i are cyclically reduced and irreducible. In case one of the traces u_i is not connected, we can write u_i as $u_i = u_{i,1} u_{i,2}$ with $u_{i,1} I u_{i,2}$ and $u_{i,1} \neq 1 \neq u_{i,2}$. Thus, we can replace the power $u_i^{x_i}$ by $u_{i,1}^{x_i} u_{i,2}^{x_i}$. Note that $u_{i,1}$ and $u_{i,2}$ are still irreducible and cyclically reduced. By doing this, the number n from the theorem multiplies by at most α (which is the maximal number of pairwise independent letters). In order to keep the notation simple we still use the letter n for the number of u_i, but at the end of the proof we have to multiply n by α in the derived bound. Hence, for the further proof we can assume that all u_i are connected, irreducible and cyclically reduced. Let λ be the maximal length of one of the traces $u_1, u_2, \ldots, u_n, v_0, v_1, \ldots, v_n$, which does not increase by the above preprocessing.

We now apply Lemma 13 to the equation

$$v_0 u_1^{x_1} v_1 u_2^{x_2} v_2 \cdots u_n^{x_n} v_n = 1, \tag{4}$$

where every $u_i^{x_i}$ is viewed as a single factor. Note that by our preprocessing, all factors $u_1^{x_1}, u_2^{x_2}, \ldots, u_n^{x_n}, v_0, \ldots, v_n$ are irreducible (for all choices of the

x_i). By taking a big disjunction over (i) all possible factorizations of the $2n+1$ factors $u_1^{x_1}, u_2^{x_2}, \ldots, u_n^{x_n}, v_0, \ldots, v_n$ into totally at most $2^{2n+1}-2$ factors and (ii) all possible reductions of the resulting refined factorization of $v_0 u_1^{x_1} v_1 u_2^{x_2} v_2 \cdots u_n^{x_n} v_n$, it follows that (4) is equivalent to a disjunction of statements of the following form: There exist traces $y_{i,1}, \ldots, y_{i,k_i}$ $(1 \leq i \leq n)$ and $z_{i,1}, \ldots, z_{i,l_i}$ $(0 \leq i \leq n)$ such that

(a) $u_i^{x_i} = y_{i,1} \cdots y_{i,k_i}$ $(1 \leq i \leq n)$

(b) $v_i = z_{i,1} \cdots z_{i,l_i}$ $(0 \leq i \leq n)$

(c) $y_{i,j} I y_{k,l}$ for all $(i,j,k,l) \in J_1$

(d) $y_{i,j} I z_{k,l}$ for all $(i,j,k,l) \in J_2$

(e) $z_{i,j} I z_{k,l}$ for all $(i,j,k,l) \in J_3$

(f) $y_{i,j} = y_{k,l}^{-1}$ for all $(i,j,k,l) \in M_1$

(g) $y_{i,j} = z_{k,l}^{-1}$ for all $(i,j,k,l) \in M_2$

(h) $z_{i,j} = z_{k,l}^{-1}$ for all $(i,j,k,l) \in M_3$

Here, the numbers k_i and l_i sum up to at most $2^{2n+1}-2$ (hence, some k_i can be exponentially large, whereas l_i can be bound by the length of v_i, which is at most λ). The tuple sets J_1, J_2, J_3 collect all independences between the factors $y_{i,j}$, $z_{k,l}$ that are necessary to carry out the chosen reduction of the refined left-hand side in (4). Similarly, the tuple sets M_1, M_2, M_3 tell us which of the factors $y_{i,j}$, $z_{k,l}$ cancels against which of the factors $y_{i,j}$, $z_{k,l}$ in our chosen reduction of the refined left-hand side in (4). Note that every factor $y_{i,j}$ (resp., $z_{k,l}$) appears in exactly one of the identities (f), (g), (h) (since in the reduction every factor cancels against another unique factor).

Next, we simplify our statements. Since the v_i are concrete traces (of length at most λ), we can take a disjunction over all possible factorizations $v_i = v_{i,1} \cdots v_{i,l_i}$ $(1 \leq i \leq n+1)$. This allows to replace every variable $z_{i,j}$ by a concrete trace $v_{i,j}$. Statements of the form $v_{i,j} I v_{k,l}$ and $v_{i,j} = v_{k,l}^{-1}$ can, of course, be eliminated. Moreover, if there is an identity $y_{i,j} = v_{k,l}^{-1}$ then we can replace the variable $y_{i,j}$ by the concrete trace $v_{k,l}^{-1}$ (of length at most λ).

In the next step, we replace statements of the form $u_i^{x_i} = y_{i,1} \cdots y_{i,k_i}$ $(1 \leq i \leq n)$. Note that some of the variables $y_{i,j}$ might have been replaced by concrete traces of length at most λ. We apply to each of these equations Lemma 6, or better Remark 7. This allows us to replace every equation

$u_i^{x_i} = y_{i,1} \cdots y_{i,k_i}$ $(1 \leq i \leq n)$ by a disjunction of statements of the following form: There exist numbers $x_{i,j} > 0$ $(1 \leq i \leq n,\ j \in K_i)$ such that

- $x_i = c_i + \sum_{j \in K_i} x_{i,j}$ for all $1 \leq i \leq n$,
- $y_{i,j} = p_{i,j} u_i^{x_{i,j}} s_{i,j}$ for all $1 \leq i \leq n$, $j \in K_i$,
- $y_{i,j} = p_{i,j} s_{i,j}$ for all $1 \leq i \leq n$, $j \in [1, k_i] \setminus K_i$.

Here, $K_i \subseteq [1, k_i]$, the c_i are concrete numbers with $c_i \leq |A| \cdot (k_i - 1)$, and the $p_{i,j}, s_{i,j}$ are concrete traces of length at most $|A| \cdot (k_i - 1) \cdot |u_i| \leq |A| \cdot (2^{2n+1} - 3) \cdot \lambda$. Hence, the lengths of these traces can be exponential in n.

Note that since $x_i > 0$, we know the alphabet of $y_{i,j} = p_{i,j} u_i^{x_{i,j}} s_{i,j}$ (resp., $y_{i,j} = p_{i,j} s_{i,j}$). This allows us to replace all independences of the form $y_{i,j} I y_{k,l}$ for $(i,j,k,l) \in J_1$ (see (c)) and $y_{i,j} I z_{k,l}$ for $(i,j,k,l) \in J_2$ (see (d)) by concrete truth values. Note that all variables $z_{k,l}$ have already been replaced by concrete traces. If $y_{i,j}$ was already replaced by a concrete trace, then we can determine from an equation $y_{i,j} = p_{i,j} u_i^{x_{i,j}} s_{i,j}$ the exponent $x_{i,j}$. Since $y_{i,j}$ was replaced by a trace of length at most λ (a small number), we get $x_{i,j} \leq \lambda$, and we can replace $x_{i,j}$ in $x_i = \sum_{j \in K_i} x_{i,j} + c_i$ by a concrete number of size at most λ. Finally, if $y_{i,j}$ was replaced by a concrete trace, and we have an equation of the form $y_{i,j} = p_{i,j} s_{i,j}$, then the resulting identity is either true or false and can be eliminated.

After this step, we obtain a big disjunction of statements of the following form: There exist numbers $x_{i,j} > 0$ $(1 \leq i \leq n,\ j \in K_i')$ such that

(a') $x_i = c_i + \sum_{j \in K_i'} x_{i,j}$ for all $1 \leq i \leq n$, and

(b') $p_{i,j} u_i^{x_{i,j}} s_{i,j} = s_{k,l}^{-1} (u_k^{-1})^{x_{k,l}} p_{k,l}^{-1}$ for all $(i,j,k,l) \in M$.

Here, $K_i' \subseteq K_i$ is a set of size at most $k_i \leq 2^{2n+1} - 2$, $c_i \leq |A| \cdot (k_i - 1) + \lambda \cdot k_i < (|A| + \lambda) \cdot (2^{2n+1} - 2)$, and the $p_{i,j}, s_{i,j}$ are concrete traces of length at most $|A| \cdot (2^{2n+1} - 3) \cdot \lambda$. The set M specifies a matching in the sense that for every exponent $x_{a,b}$ $(1 \leq a \leq n,\ b \in K_i')$ there is a unique $(i,j,k,l) \in M$ such that $(i,j) = (a,b)$ or $(k,l) = (a,b)$.

We now apply Lemma 11 to the identities $p_{i,j} u_i^{x_{i,j}} s_{i,j} = s_{k,l}^{-1} (u_k^{-1})^{x_{k,l}} p_{k,l}^{-1}$. Each such identity can be replaced by a disjunction of constraints

$$(x_{i,j}, x_{k,l}) \in \{(a_{i,j,k,l} + b_{i,j,k,l} \cdot z_{i,j,k,l}, c_{i,j,k,l} + d_{i,j,k,l} \cdot z_{i,j,k,l}) \mid z_{i,j,k,l} \in \mathbb{N}\}.$$

For the numbers $a_{i,j,k,l}, b_{i,j,k,l}, c_{i,j,k,l}, d_{i,j,k,l}$ we obtain the bound

$$a_{i,j,k,l}, b_{i,j,k,l}, c_{i,j,k,l}, d_{i,j,k,l} \in \mathcal{O}(\mu^8 \cdot \nu^{8|A|})$$

(the alphabet of the traces is $A^{\pm 1}$ which has size $2|A|$, therefore, we have to multiply in Lemma 11 $|A|$ by 2), where, by Lemma 3,

$$\mu = \max\{\rho(p_{i,j}), \rho(p_{k,l}), \rho(s_{i,j}), \rho(s_{k,l})\} \in \mathcal{O}(|A|^{\alpha} \cdot 2^{2\alpha n} \cdot \lambda^{\alpha}) \tag{5}$$

and

$$\nu = \max\{\rho(u_i), \rho(u_k)\} \in \mathcal{O}(\lambda^{\alpha}). \tag{6}$$

Note that $\rho(t) = \rho(t^{-1})$ for every trace t. The above condition (a') for x_i can be now written as

$$x_i = c_i + \sum_{(i,j,k,l)\in M} (a_{i,j,k,l} + b_{i,j,k,l} \cdot z_{i,j,k,l}) + \sum_{(k,l,i,j)\in M} (c_{k,l,i,j} + d_{k,l,i,j} \cdot z_{k,l,i,j}).$$

Note that the two sums in this equation contain in total $|K_i'| \leq 2^{2n+1}$ many summands (since for every $j \in K_i'$ there is a unique pair (k,l) with $(i,j,k,l) \in M$ or $(k,l,i,j) \in M$).

Hence, after a renaming of symbols, the initial equation (4) becomes equivalent to a finite disjunction of statements of the form: There exist $z_1, \ldots, z_m \in \mathbb{N}$ (these z_i are the above $z_{i,j,k,l}$ and $m = \max_i |K_i'|$) such that

$$x_i = a_i + \sum_{j=1}^{m} a_{i,j} z_j \text{ for all } 1 \leq i \leq n. \tag{7}$$

Moreover, we have the following size bounds:

- $m = \max_i |K_i'| \leq 2^{2n+1}$,
- $a_i \in \mathcal{O}(c_i + |K_i'| \cdot \mu^8 \cdot \nu^{8|A|}) \subseteq \mathcal{O}(2^{2n}(|A| + \lambda + \mu^8 \cdot \nu^{8|A|})) \subseteq \mathcal{O}(2^{2n} \cdot \mu^8 \cdot \nu^{8|A|})$
- $a_{i,j} \in \mathcal{O}(\mu^8 \cdot \nu^{8|A|})$

Recall that some of the variables x_i can be identical. W.l.o.g. assume that $x_1, \ldots, x_k$ are pairwise different and for all $k+1 \leq i \leq n$, $x_i = x_{f(i)}$, where $f\colon [k+1, n] \to [1, k]$. Then, the system of equations (7) is equivalent to

$$x_i = a_i + \sum_{j=1}^{m} a_{i,j} z_j \text{ for all } 1 \leq i \leq k$$

$$a_i - a_{f(i)} = \sum_{j=1}^{m} (a_{f(i),j} - a_{i,j}) z_j \text{ for all } k+1 \leq i \leq n.$$

The set of all $(x_1, \ldots, x_k) \in \mathbb{N}^k$ for which there exist $z_1, \ldots, z_m \in \mathbb{N}$ satisfying these equalities is semilinear by Lemma 12, and if it is non-empty then it contains a vector $(x_1, \ldots, x_k) \in \mathbb{N}^k$ such that

$$x_i \in \mathcal{O}(n! \cdot m^2 \cdot 2^{2n(n+1)} \cdot \mu^{8(n+1)} \cdot \nu^{8|A|(n+1)}) \subseteq \mathcal{O}(n! \cdot 2^{2n(n+3)} \cdot \mu^{8(n+1)} \cdot \nu^{8|A|(n+1)}).$$

Recall that in this bound we have to replace n by $\alpha \cdot n$ due to the initial preprocessing. This proves the theorem. □

Theorem 15. *Let (A, I) be a fixed independence alphabet. Solvability of compressed exponent equations over the graph group $\mathbb{G}(A, I)$ is in* NP.

Proof. Consider a compressed exponent equation

$$E = (v_0 u_1^{x_1} v_1 u_2^{x_2} v_2 \cdots u_n^{x_n} v_n = 1),$$

where $u_i = \mathrm{val}(\mathcal{G}_i)$ and $v_i = \mathrm{val}(\mathcal{H}_i)$ for given SLPs $\mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{H}_0, \ldots, \mathcal{H}_n$. Let $m = \max\{|\mathcal{G}_1|, \ldots, |\mathcal{G}_n|, |\mathcal{H}_0|, \ldots, |\mathcal{H}_n|\}$. By Theorem 14 we know that if there exists a solution for E then there exists a solution σ with $\sigma(x_i) \in \mathcal{O}((\alpha n)! \cdot 2^{2\alpha^2 n(n+3)} \cdot \mu^{8\alpha(n+1)} \cdot \nu^{8\alpha|A|(n+1)})$, where

- $\mu \in \mathcal{O}(|A|^\alpha \cdot 2^{2\alpha^2 n} \cdot \lambda^\alpha)$,
- $\nu \in \mathcal{O}(\lambda^\alpha)$,
- $\lambda = \max\{|u_1|, |u_2|, \ldots, |u_n|, |v_0|, |v_1|, \ldots, |v_n|\} \in 2^{\mathcal{O}(m)}$, and
- $\alpha \leq |A|$.

Note that the bound on the $\sigma(x_i)$ is exponential in the input length (the sum of the sizes of all $\mathcal{G}_i$ and $\mathcal{H}_i$). Hence, we can guess in polynomial time the binary encodings of numbers $k_i \in \mathcal{O}((\alpha n)! \cdot 2^{2\alpha^2 n(n+3)} \cdot \mu^{8\alpha(n+1)} \cdot \nu^{8\alpha|A|(n+1)})$ (where $k_i = k_j$ if $x_i = x_j$). Then, we have to verify whether

$$\mathrm{val}(\mathcal{H}_0)\mathrm{val}(\mathcal{G}_1)^{k_1}\mathrm{val}(\mathcal{H}_1)\mathrm{val}(\mathcal{G}_2)^{k_2}\mathrm{val}(\mathcal{H}_2) \cdots \mathrm{val}(\mathcal{G}_n)^{k_n}\mathrm{val}(\mathcal{H}_n) = 1$$

in the graph group $\mathbb{G}(A, I)$. This is an instance of the so-called *compressed word problem* for $\mathbb{G}(A, I)$, where the input consists of an SLP $\mathcal{G}$ over the alphabet $A^{\pm 1}$ and it is asked whether $\mathrm{val}(\mathcal{G}) = 1$ in $\mathbb{G}(A, I)$. Note that the big powers $\mathrm{val}(\mathcal{G}_i)^{k_i}$ can be produced with the productions of $\mathcal{G}_i$ and additional $\lceil \log k_i \rceil$ many productions (using iterated squaring). Since the compressed word problem for a graph group can be solved in deterministic polynomial time [32, 33] (NP would suffice), the theorem follows. For the last step, it is important that (A, I) is fixed. □

Remark 16. *Note that the bound on the exponents $\sigma(x_i)$ in the previous proof is still exponential in the input length if the independence alphabet (A, I) is part of the input as well. The problem is that we do not know whether the* uniform compressed word problem *for graph groups (where the input is an independence alphabet (A, I) together with an SLP over the terminal alphabet $A^{\pm 1}$) can be solved in polynomial time or at least in* NP*. The latter would suffice to get an* NP*-algorithm for solvability of compressed exponent equations over a graph group that is part of the input.*

9 Transfer Results

In this section, we show that the property of having an NP-algorithm for the knapsack problem (or compressed exponent equations) is preserved by certain group constructions. Specifically, we show that the class of groups that admit an NP-algorithm for knapsack is closed under (i) finite extensions, (ii) HNN-extensions with finite associated subgroups, and (iii) amalgamated free products with finite identified subgroups. In the case of finite extensions, the transfer also holds for compressed exponent equations.

Finite extensions and virtually special groups. Our first transfer result concerns finite extensions. Together with our result on graph groups, this will provide a large class of groups with an NP-algorithm for compressed exponent equations. A group G is called *virtually special* if it is a finite extension of a subgroup of a graph group. Recently, this class of groups turned out to be very rich. It contains the following classes of groups:

- Coxeter groups [20]
- one-relator groups with torsion [46]
- fully residually free groups [46]
- fundamental groups of hyperbolic 3-manifolds [1]

The following is our transfer theorem for finite extensions.

Theorem 17. *Let G and H be finitely generated groups such that H is a finite extension of G. If knapsack (resp. solvability of compressed exponent equations) belongs to* NP *for G, then the same holds for H.*

From Theorem 15 it follows that solvability of compressed exponent equations belongs to NP for every subgroup of a graph group. Therefore, our transfer theorem implies:

Theorem 18. *Solvability of compressed exponent equations belongs to* NP *for every virtually special group. In particular, solvability of compressed exponent equations belongs to* NP *for Coxeter groups, one-relator groups with torsion, fully residually free groups, and fundamental groups of hyperbolic 3-manifolds.*

We need the following statement, which is shown implicitly in the proof of [32, proof of Theorem 4.4].

Lemma 19. *Let G and H be finitely generated groups such that H is a finite extension of G and let C be a set of right coset representatives of G. Let A (resp. $B \supseteq A$) be a finite generating set for G (resp., H). From a given SLP $\mathcal{H}$ over the terminal alphabet $B^{\pm 1}$ one can compute in polynomial time (i) the unique coset representative $c \in C$ such that $\mathrm{val}(\mathcal{H}) \in Gc$ and (ii) an SLP $\mathcal{G}$ over the terminal alphabet $A^{\pm 1}$ such that $\mathrm{val}(\mathcal{G})c = \mathrm{val}(\mathcal{H})$ holds in the group H.*

Proof of Theorem 17. In [29], it was shown that for each finitely generated group, the knapsack problem and the solvability of exponential expressions where each variable occurs only once (the latter is called *generalized knapsack problem* there) are polynomially inter-reducible. Therefore, we shall prove that exponential expression over H can be reduced to exponential expressions over G. Moreover, the reduction preserves the property that each variable occurs only once. We only describe the case that all inputs are uncompressed; by means of Lemma 19, the compressed case can be treated analogously.

Assume that $[H : G] = m$ and let C be a set of coset representatives, $|C| = m$. Let A be a finite generating set for G and let $B \supseteq A$ be a finite generating set for H. Suppose we are given an exponent equation

$$v_0 u_1^{x_1} v_1 \cdots u_n^{x_n} v_n = 1 \tag{8}$$

in H where the v_i and the u_i are represented as words over $B^{\pm 1}$. As a first step, we guess which of the variables x_i assume a value smaller than m. For those that do, we can guess the value and merge the result in a neighboring v_i. This increases the size of the instance by at most a factor of m, which is a constant. Hence, from now on, we only look for solutions to (8) where $x_i \geq m$ for $1 \leq i \leq n$.

The next step of our NP algorithm is to guess the cosets occurring in a solution. This means, we guess $d_0, c_1, d_1, \ldots, c_n, d_n \in C$ and look for a solution to (8) such that $v_0 u_1^{x_1} v_1 \cdots u_i^{x_i} v_i \in Gd_i$ and $v_0 u_1^{x_1} v_1 \cdots u_i^{x_i} \in Gc_i$

for $0 \leq i \leq n$. This is equivalent to a solution where the elements

$$v_0 d_0^{-1}, \quad d_{i-1} u_i^{x_i} c_i^{-1}, \quad c_i v_i d_i^{-1}, \quad d_n$$

all belong to G for $1 \leq i \leq n$. We can verify in polynomial time that $v_0 d_0^{-1}$, $c_i v_i d_i^{-1}$ $(1 \leq i \leq n)$, and d_n belong to G. Therefore, we want to check whether there is a solution to (8) where $d_{i-1} u_i^{x_i} c_i^{-1} \in G$ for $1 \leq i \leq n$.

Consider the function $f_i \colon C \to C$, which is defined so that for each $c \in C$, $f_i(c)$ is the unique element $d \in C$ with $c u_i d^{-1} \in G$. Note that we can compute f_i in polynomial time. Then there are numbers $1 \leq k_i \leq m$ such that $f_i^{m+k_i}(d_{i-1}) = f_i^m(d_{i-1})$. With this notation, we have $d_{i-1} u_i^{x_i} c_i^{-1} \in G$ if and only if $f_i^{x_i}(d_{i-1}) = c_i$.

We may assume that there is an $x_i \geq m$ with $f_i^{x_i}(d_{i-1}) = c_i$: Otherwise, there is no solution and we can terminate our branch. Therefore, there is a $0 \leq r_i < k_i$ such that $f_i^{m+r_i}(d_{i-1}) = c_i$. This means, we have $f_i^{x_i}(d_{i-1}) = c_i$ for $x_i \geq m$ if and only if $x_i = m + k_i \cdot y_i + r_i$ for some $y_i \geq 0$. This allows us to construct an exponent equation over G.

Let $e_i = f_i^m(d_{i-1})$. Then, the elements $d_{i-1} u_i^m e_i^{-1}$, $e_i u_i^{k_i} e_i^{-1}$, and $e_i u_i^{r_i} c_i^{-1}$ all belong to G. Moreover, for $x_i = m + k_i \cdot y_i + r_i$, we have

$$\begin{aligned} v_0 u_1^{x_1} v_1 \cdots u_n^{x_n} v_n &= v_0 d_0^{-1} \prod_{i=1}^{n} d_{i-1} u_i^{m+k_i \cdot y_i + r_i} c_i^{-1} c_i v_i d_i^{-1} \\ &= (v_0 d_0^{-1}) \prod_{i=1}^{n} (d_{i-1} u_i^m e_i^{-1})(e_i u_i^{k_i} e_i^{-1})^{y_i} (e_i u_i^{r_i} c_i^{-1} c_i v_i d_i^{-1}) \end{aligned}$$

and each term in parentheses belongs to G. This clearly yields an exponent equation over G (with variables $y_1, \ldots, y_n$) that is solvable if and only if there is a solution of (8) of the kind we are looking for. It remains to verify that the new instance is polynomial in size.

There is a constant ℓ such that given a word w representing $h \in H$ and elements $c, d \in C$ such that $chd^{-1} \in G$, a word of length at most $\ell \cdot |w|$ representing chd^{-1} over $A^{\pm 1}$ is computable in linear time. Let $s_i, t_j \in (B^{\pm 1})^*$ represent v_i and u_j, respectively, for $0 \leq i \leq n$ and $1 \leq j \leq n$. Then, the new instance has size at most

$$\ell |s_0| + \sum_{i=1}^{n} \ell(m + k_i + r_i)|t_i| + \ell |s_i| \leq 3m\ell(|s_0| + |t_1| + |s_1| + \cdots |t_n| + |s_n|)$$

which is linear in the size of the old instance. □

HNN-extensions and amalgamated products. The remaining transfer results concern two constructions that are of fundamental importance in combinatorial group theory [37], namely HNN-extensions and amalgamated products. In their general form, HNN-extensions have been used to construct groups with an undecidable word problem, which means they may destroy desirable algorithmic properties. We consider the special case of finite associated (resp. identified) subgroups, for which these constructions already play a prominent role, for example, in Stallings' decomposition of groups with infinitely many ends [43] or the construction of virtually free groups [10]. Moreover, these constructions are known to preserve a wide range of important structural and algorithmic properties [2, 6, 21, 24, 25, 27, 28, 34, 35, 38].

Suppose $G = \langle \Sigma \mid R \rangle$ is a finitely generated group that has two isomorphic subgroups A and B with an isomorphism $\varphi\colon A \to B$. Then the corresponding *HNN-extension* is the group

$$H = \langle G, t \mid t^{-1}at = \varphi(a)\ (a \in A) \rangle,$$

where t is a new letter not contained in G. In other words, H is the group $H = \langle \Sigma \cup \{t\} \mid R \cup \{t^{-1}at = \varphi(a) \mid a \in A\} \rangle$ with $t \notin \Sigma$. Intuitively, H is obtained from G by adding a new element t such that conjugating elements of A with t applies the isomorphism φ. Here, t is called the *stable letter* and the groups A and B are the *associated subgroups*. A basic fact about HNN-extensions is that the group G embeds naturally into H [22].

Here, we only consider the case that A and B are finite groups, so that we may assume that $A \cup B \subseteq \Sigma$. To exploit the symmetry of the situation, we use the notation $A(+1) = A$ and $A(-1) = B$. Then, we have $\varphi^\alpha\colon A(\alpha) \to A(-\alpha)$ for $\alpha \in \{+1, -1\}$. By $h\colon (\Sigma^{\pm 1} \cup \{t, t^{-1}\})^* \to H$, we denote the canonical morphism that maps each word to the element of H it represents.

A word $u \in (\Sigma^{\pm 1} \cup \{t, t^{-1}\})^*$ is called *reduced* if it does not contain a factor $t^{-\alpha}wt^{\alpha}$ with $\alpha \in \{-1, 1\}$, $w \in (\Sigma^{\pm 1})^*$, and $h(w) \in A(\alpha)$. Note that the equation $t^{-1}at = \varphi(a)$, $a \in A$, allows us to replace such a factor $t^{-\alpha}wt^{\alpha}$ by $\varphi^\alpha(h(w)) \in A(-\alpha) \subseteq \Sigma$. Since this reduces the number of t's in the word, this allows us to turn every word into an equivalent reduced word. The following well-known fact describes the reduced words representing the identity [35, Lemma 5].

Lemma 20. *If $u \in (\Sigma^{\pm 1} \cup \{t, t^{-1}\})^*$ is a reduced word representing $1 \in H$, then $u \in (\Sigma^{\pm 1})^*$.*

Our algorithm for knapsack in HNN-extensions is an adaptation of the saturation algorithm of Benois [3] for the membership problem for rational

subsets of free groups. Here, for each path spelling aa^{-1}, one adds a parallel edge labeled with the empty word. Since knapsack is a special case of this problem, we have to use a suitable subclass of automata that is preserved by our saturation and corresponds to the knapsack problem.

Let G be a group with finite generating set Σ. A *finite automaton over* G is an NFA $\mathcal{A} = (Q, \Sigma^{\pm 1}, \Delta, q_0, F)$. A *(directed) cycle* in $\mathcal{A}$ is a sequence $p_1, \ldots, p_n$ of states such that there are edges (p_i, a_i, p_{i+1}) for $1 \leq i \leq n-1$ and (p_n, a_n, p_1) with $a_1, \ldots, a_n \in \Sigma^{\pm 1}$. In particular, a single state with a loop is regarded as a cycle. A sequence $p_1, \ldots, p_n$ is an *induced cycle* if it is a cycle and there are no other edges among the states $p_1, \ldots, p_n$. We call $\mathcal{A}$ a *knapsack automaton* if every strongly connected component of $\mathcal{A}$ is a singleton or an induced cycle. The *membership problem for knapsack automata over* G is the following decision problem:

Input: A knapsack automaton $\mathcal{A}$ over G and a word $w \in (\Sigma^{\pm 1})^*$.

Question: Does $\mathcal{A}$ accept a word w' that represents the same element of G as w?

Indeed, the membership problem for knapsack automata corresponds precisely to the knapsack problem in the following sense.

Lemma 21. *For each finitely generated group, knapsack belongs to* NP *if and only if membership for knapsack automata belongs to* NP*.*

Proof. It is easy to turn a knapsack instance into a knapsack automaton: Given words $w_1, \ldots, w_k, w \in (\Sigma^{\pm 1})^*$, one can clearly construct a knapsack automaton accepting $w_1^* \cdots w_k^*$. Then, the knapsack problem amounts to deciding the membership problem for w.

Now, suppose we are given a knapsack automaton $\mathcal{A}$ over G and a word $w \in (\Sigma^{\pm 1})^*$. We can clearly turn $\mathcal{A}$ into a knapsack automaton that first reads w^{-1} and then behaves like $\mathcal{A}$. Therefore, it suffices to solve the membership problem in the case that $w = \varepsilon$.

Consider a run r in $\mathcal{A}$ from the initial to a final state. Let $c_1, \ldots, c_n$ be the sequence of strongly connected components it visits. For each c_i that is not a singleton, let p_i and q_i be the state where r enters and leaves c_i, respectively. We call the sequence $c_1, \ldots, c_n$, together with the p_i and q_i the *skeleton* of r.

Our algorithm guesses a skeleton. Since $\mathcal{A}$ is a knapsack automaton, from this skeleton, we can determine words $v_0, u_1, v_1, \ldots, u_n, v_n \in (\Sigma^{\pm 1})^*$ such that $v_0 u_1^* v_1 \cdots u_n^* v_n$ is precisely the set of words labeling a path with this skeleton. Hence, deciding the membership problem for $\mathcal{A}$ amounts to checking whether there are $x_1, \ldots, x_n \in \mathbb{N}$ with $h_0 g_1^{x_1} h_1 \cdots g_n^{x_n} h_n = 1$, where

g_i (h_j, respectively) is the element represented by u_i (v_j, respectively). This is an exponential equation with pairwise distinct variables and the solvability of such equations is called the *generalized knapsack problem* in [29], where it was shown to be polynomially inter-reducible with the knapsack problem. □

Theorem 22. *Let H be an HNN-extension of the finitely generated group G with finite associated subgroups. If knapsack for G belongs to* NP*, then the same holds for H.*

Proof. According to Lemma 21, it suffices to prove that if membership for knapsack automata over G belongs to NP, then the same holds for H. Hence, let $\mathcal{A}$ be a knapsack automaton over H. As explained above, it suffices to check membership for the group identity, i.e., to check whether $\mathcal{A}$ accepts a word from $h^{-1}(1)$.

The basic idea of the proof is to saturate $\mathcal{A}$, yielding a knapsack automaton that is *saturated*, meaning: For each path from p to q labeled with a word $t^{-\alpha}wt^{\alpha}$ with $h(w) \in A(\alpha)$, there is an edge from p to q labeled with $\varphi^{\alpha}(h(w)) \in A(-\alpha)$. We will then show that $\mathcal{A}$ accepts a word from $h^{-1}(1)$ if and only if it accepts a word from $h^{-1}(1) \cap (\Sigma^{\pm 1})^*$. This will allow us to remove all $t^{\pm 1}$-edges and apply the algorithm for G. A path in a knapsack automaton that is labeled by a word $t^{-\alpha}wt^{\alpha}$ with $w \in (\Sigma^{\pm 1})^* \cap h^{-1}(A(\alpha))$ is called a *reduction path*. Among other things, the algorithm will introduce a *shortcut edge* for the reduction path, namely

$$p \xrightarrow{\varphi^{\alpha}(a)} q, \tag{9}$$

where $a = h(w) \in A(\alpha)$. Observe that $\varphi^{\alpha}(a) \in A(-\alpha)$ and $\varphi^{\alpha}(a) = h(t^{-\alpha}wt^{\alpha})$. By introducing intermediate states, we may assume that (i) there is no edge between states that belong to distinct cycles and (ii) the initial and the final state do not lie on a cycle.

Phase 1. The saturation proceeds in two phases. In the first phase, we saturate the directed cycles, which are the strongly connected components. This means, we modify the automaton so that there is no reduction path between two states on a cycle. This is done as follows. We successively guess tuples (p, α, a, q) where p and q are states from the same cycle, $\alpha \in \{-1, 1\}$, and $a \in A(\alpha)$. Then, employing the NP algorithm for G, we can clearly verify that there is a reduction path spelling $t^{-\alpha}wt^{\alpha}$ from p to q with $w \in (\Sigma^{\pm 1})^*$ and $h(w) = a$. Note that on this path, the first letter ($t^{-\alpha}$) occurs

only once, meaning the path visits each state at most once (i.e. it makes at most one round in the cycle). Let

$$p = r_0 \xrightarrow{u_1} r_1 \cdots \xrightarrow{u_n} r_n = q \tag{10}$$

be the reduction path and let

$$q = r_n \xrightarrow{u_{n+1}} r_{n+1} \cdots \xrightarrow{u_m} r_m = r_0 = p$$

be the rest of the cycle with $u_1, \ldots, u_m \in \Sigma^{\pm 1} \cup \{t, t^{-1}\}$ and $u_1 \cdots u_n = t^{-\alpha} w t^{\alpha}$. In particular, m is the length of the cycle. Let us now describe the saturation step. We remove all edges from (10) and all states incident to them, except for p and q. Instead, we add a shortcut edge (9). For each state s not on the cycle and for which there is an edge (s, v, r_i), $1 \leq i \leq n-1$, we glue in a path

$$s \xrightarrow{v} s_0 \xrightarrow{u_{i+1}} s_1 \cdots \xrightarrow{u_n} s_{n-i} = q, \tag{11}$$

where $s_0, \ldots, s_{n-i-1}$ are new states. Analogously, for each state s not on the cycle and for which there is an edge (r_i, v, s), $1 \leq i \leq n-1$, we glue in a path

$$p = s_0 \xrightarrow{u_1} s_1 \cdots \xrightarrow{u_i} s_i \xrightarrow{v} s, \tag{12}$$

where $s_1, \ldots, s_i$ are new states. Moreover, for each pair (s, s') of states not on the cycle and for which there are edges (s, v, r_i) and (r_j, v', s') with $1 \leq i < j \leq n-1$, we glue in a path

$$s \xrightarrow{v} s_0 \xrightarrow{u_{i+1}} s_1 \cdots \xrightarrow{u_j} s_{j-i} \xrightarrow{v'} s', \tag{13}$$

where $s_0, \ldots, s_{j-i}$ are new states. This completes our saturation step.

Let $\mathcal{A}'$ be the automaton resulting from one saturation step from $\mathcal{A}$. Then, $\mathcal{A}'$ is clearly a knapsack automaton: We only connect states that were connected before. Moreover, for states s, s' that exists in $\mathcal{A}$ and in $\mathcal{A}'$, the set of group elements represented on paths from s to s' does not change. Indeed, a path that avoids our cycle still exists. A path that involves the whole path (10) can use the shortcut edge (9). A path that either (i) enters (10) after p and follows it until q or (ii) follows (10) partly and then leaves before q can use the new paths (11) or (12), respectively. Finally, a path that follows only a part of (10) that starts after p and ends before q can use the new path (13) instead.

Let us estimate the number of added states during Phase 1. The *degree* of a cycle is the number of edges entering or leaving the cycle. Let d be the degree of our cycle. Let us first consider a single saturation step. The new

states of type (11) or (12) are each at most $d \cdot n$ many. The new states of type (13) are at most $d^2 \cdot n$ many. Hence, we add at most $(d^2+2d)n \leq (d^2+2d)m$ states in this saturation step. Observe that in this step, the length of the affected cycle decreases ($t^{-\alpha}wt^{\alpha}$ has length ≥ 2 and $h(t^{-\alpha}wt^{\alpha}) \in A(-\alpha)$ has length 1) and its degree is unchanged (the new edges from (11) and (12) clearly preserve the degree and those of (13) do not increase the degree because by our assumption that no edge connects two cycles, s and s' do not belong to a cycle). Now, we consider the whole phase. Suppose in the beginning, $\mathcal{A}$ has c cycles of maximal degree d and maximal length ℓ. Then, each saturation step adds at most $(d^2+2d)\ell$ states. Moreover, there can be at most $\ell \cdot c$ saturation steps, so that the first phase adds at most $(d^2+2d)\ell^2 c$ states, which is polynomial in the size of the input automaton.

Phase 2. In the second phase, we consider reduction paths between states that belong to distinct strongly connected components. Since here, adding an edge that runs parallel to the reduction path cannot violate the property of being a knapsack automaton, we may saturate by simply introducing new edges.

Again, we successively guess tuples (p, α, a, q) where $\alpha \in \{-1, 1\}$, and $a \in A(\alpha)$. However, we require that p and q are not from the same strongly connected component and that there is no shortcut edge (9) yet. As above, we employ the NP algorithm for G to verify that there is a reduction path spelling $t^{-\alpha}wt^{\alpha}$ from p to q with $w \in (\Sigma^{\pm 1})^*$ and $h(w) = a$. Then, we add the shortcut edge (9). As before, we have $h(t^{-\alpha}wt^{\alpha}) = \varphi^{\alpha}(a) \in A(-\alpha)$. This is all we do in the saturation step. Since now, we only add edges (and no states) and each correct guess leads to an increase in the number of edges, our sequence of saturation steps must terminate after a polynomial number of steps. This concludes the second phase and thus the saturation.

Finally, the algorithm applies the NP-algorithm for G. More precisely, we remove all edges labeled $\{t, t^{-1}\}$. This yields a knapsack automaton over G, so that we can use the algorithm for G to check whether it accepts $1 \in G$. Then, we answer "yes" if and only if the algorithm for G does.

It remains to be shown that this algorithm is sound and complete. If we answer "yes", then the input automaton accepts $1 \in H$. This is because each saturation step preserves the set of accepted elements. On the other hand, suppose the input automaton $\mathcal{A}$ accepts $1 \in H$ and consider the branch of our nondeterministic algorithm that guesses in such a way that in the end, there are no more reduction paths without a shortcut edge. Let $\mathcal{B}$ be the resulting saturated knapsack automaton. Since $\mathcal{A}$ accepts $1 \in H$, there is

an accepting run in $\mathcal{B}$ that accepts $1 \in H$. Consider such a run reading a word $u \in (\Sigma^{\pm 1} \cup \{t, t^{-1}\})^*$ with a minimal number of occurrences of t. Since $\mathcal{B}$ is saturated, this implies that u is reduced: Otherwise, u would have a factor $t^{-\alpha} w t^{\alpha}$ with $w \in (\Sigma^{\pm 1})^*$ and $h(w) \in A(\alpha)$. This factor, however, lies on a reduction path and we could have used the shortcut edge instead, which would result in a run with fewer t's. Since u is reduced and represents $1 \in H$, it contains neither t nor t^{-1} (Lemma 20). Hence, our application of the algorithm for G answers "yes" because of u. □

In our last transfer theorem, we consider amalgamated free products. For $i \in \{0, 1\}$, let $G_i = \langle \Sigma_i \mid R_i \rangle$ be a finitely generated group and let F be a finite group that is embedded in each G_i, meaning that there are injective morphisms $\varphi_i \colon F \to G_i$ for $i \in \{0, 1\}$. Then, the *free product with amalgamation with identified subgroup* F is defined as

$$G_0 *_F G_1 = \langle G_0 * G_1 \mid \varphi_0(f) = \varphi_1(f) \ (f \in F) \rangle.$$

Here, $G_0 * G_1$ denotes the free product $G_0 * G_1 = \langle \Sigma_0 \uplus \Sigma_1 \mid R_0 \uplus R_1 \rangle$. Note that the product depends on the morphisms φ_i, although they are omitted in the notation $G_0 *_F G_1$. Equivalently, $G_0 *_F G_1$ is given by the presentation

$$\langle \Sigma_0 \uplus \Sigma_1 \mid R \uplus S \cup \{\varphi_0(f) = \varphi_1(f) \mid f \in F\} \rangle.$$

Let us consider the free product $G_0 * G_1$. Let $h \colon (\Sigma_0^{\pm 1} \cup \Sigma_1^{\pm 1})^* \to G_0 * G_1$ be the canonical morphism that maps a word to the group element it represents. If $w \in (\Sigma_0^{\pm 1} \cup \Sigma_1^{\pm 1})^*$, then a *syllable of* w is a factor of w that is contained in $(\Sigma_0^{\pm 1})^+ \cup (\Sigma_1^{\pm 1})^+$ and that is maximal with this property. The definition of the free product immediately implies the following.

Lemma 23. *If in the free product $G_0 * G_1$, a word represents $1 \in G_0 * G_1$, then it contains a syllable s with $h(s) = 1$.*

The transfer theorem states that taking amalgamated products with finite identified subgroups preserves NP-membership of knapsack.

Theorem 24. *Let G_0 and G_1 be finitely generated groups with a common finite subgroup F. If knapsack for G_0 and for G_1 belongs to NP, then the same holds for the amalgamated product $G_0 *_F G_1$.*

Proof. It is well-known [37, Theorem 2.6, p. 187] that $G_0 *_F G_1$ can be embedded into the HNN-extension

$$I = \langle G_0 * G_1, t \mid t^{-1} \varphi_0(f) t = \varphi_1(f) \ (f \in F) \rangle$$

by way of the morphism $\Phi\colon G_0 *_F G_1 \to I$ with

$$\Phi(g) = \begin{cases} t^{-1}gt & \text{if } g \in G_0 \\ g & \text{if } g \in G_1. \end{cases}$$

Since Theorem 22 already tells us that NP-membership of knapsack is preserved by HNN-extensions with finite associated subgroups, it suffices to show that free products preserve NP-membership.

We use a slight modification of the nondeterministic algorithm from the proof of Theorem 22 and show that if membership for knapsack automata belongs to NP for G_0 and G_1, the same holds for $G_0 * G_1$. During the saturation, we maintain the following invariants:

(i) There is no edge between states that belong to distinct cycles.

(ii) The initial and the final states do not lie on a cycle.

(iii) Every edge entering a cycle is labeled with the empty word ε.

By introducing intermediate states, we can clearly achieve them in the beginning. As in the proof of Theorem 22, we add shortcut edges for reduction paths. For states p and q, a *reduction path (from p to q)* is a path labeled by a word $w \in (\Sigma_i^{\pm 1})^+$ for some $i \in \{0,1\}$ such that (a) $h(w) = 1$ and (b) if p and q lie on a cycle, then this cycle also contains a letter in $\Sigma_{1-i}^{\pm 1}$. Here, we need the additional condition (b) to make sure that short-cutting a reduction path actually reduces the cycle (Without requiring (b), it could happen that a reduction path occupies more than one round of a cycle.) A *shortcut edge* is then simply (p, ε, q).

Again, our saturation consists of two phases and in the first one, we shortcut reduction paths inside of cycles. We guess tuples (p, i, q) such that p and q lie on a cycle and $i \in \{0,1\}$. Using the NP-algorithm for G_i, we verify that there is a reduction path from p to q labeled with $w \in (\Sigma_i^{\pm 1})^+$. Then, we proceed as in the proof of Theorem 22 and replace the reduction path with a shortcut edge and add new paths almost as in (11), (12), and (13): The only difference is that the new paths of type (11) are are prolonged with an ε-edge at the end so as to preserve invariant (iii).

While in the proof of Theorem 22, the length the cycle decreases in a saturation step, this is not guaranteed here. This is because in the proof of Theorem 22, we always remove edges labeled t and t^{-1}. Here, it could happen that the reduction path consists of one edge labeled $a \in \Sigma_i^{\pm 1}$ with $h(a) = 1$. Then, the length of the cycle is unchanged. We do, however, reduce the number of letters on the cycle. Therefore, an analogous estimation

of the number of introduced states applies and shows that it is polynomially bounded.

The second phase works just as for Theorem 22. We guess triples (p, i, q) such that p and q are not in the same strongly connected component but there is no shortcut edge (p, ε, q) yet. Then, we verify that there is a reduction path from p to q with label $w \in (\Sigma_i^{\pm 1})^+$. If this is the case, we add a shortcut edge (p, ε, q).

In the end, we guess $i \in \{0, 1\}$ and verify, using the NP-algorithm for G_i, that the automaton, restricted to $\Sigma_i^{\pm 1}$, accepts a word representing $1 \in G_i$.

Let us show that this algorithm is sound and complete. As above, we can argue that if it answers "yes", then the input automaton clearly accepts $1 \in G_0 * G_1$. For the completeness, we have to argue slightly differently. Suppose the input automaton accepts a word representing $1 \in G_0 * G_1$. We consider a branch of the nondeterministic algorithm that saturates every reduction path. Let $\mathcal{B}$ be the resulting automaton. Since $\mathcal{B}$ also accepts a word representing $1 \in G_0 * G_1$, we consider such a word $w \in (\Sigma_0^{\pm 1} \cup \Sigma_1^{\pm 1})^*$ with a minimal number of syllables.

Suppose w has more than one syllable. By Lemma 23, it contains a syllable $s \in (\Sigma_i^{\pm 1})^+$ with $h(s) = 1$. Consider the accepting run r for w and let p and q be the states occupied before and after reading s. The path taken by r from p to q is not a reduction path, because otherwise we could have taken a shortcut edge instead, in contradiction to the minimality of w. This means, p and q lie on a cycle that contains only letters in $\Sigma_i^{\pm 1}$. Since s is a syllable, this implies that r enters this cycle at p. Let p' be the state occupied in r directly before p: Note that r cannot start in p because of invariant (ii). Because of invariant (iii), the edge from p' to p is labeled with ε. Thus, the path taken by r from p' to q is a reduction path, again contradicting the minimality of w.

Hence, w has at most one syllable, which means $w \in (\Sigma_j^{\pm 1})^*$ for some $j \in \{0, 1\}$ and our application of the NP-algorithm for G_j answers "yes". □

10 Hardness Results

Since knapsack for binary encoded integers is NP-complete, it follows that the compressed knapsack problem is NP-hard for every group that contains an element of infinite order. In this section, we prove that (uncompressed) knapsack and subset sum are NP-complete for a direct product of two free groups of rank at least two. This solves an open problem from [17].

With $F(\Sigma)$ we denote the free group generated by the set Σ. Moreover,

let $F_2 = F(\{a, b\})$.

Theorem 25. *The subset sum problem and the knapsack problem are* NP*-complete for $F_2 \times F_2$. For knapsack,* NP*-hardness already holds for the variant where the exponent variables are allowed to take values from $\mathbb{Z}$ (see Remark 1).*

Proof. In [41] it was shown that there exists a fixed set $D \subseteq F_2 \times F_2$ such that the following problem (called the bounded submonoid problem) is NP-complete:

Input: A unary encoded number n (i.e., n is given by the string a^n) and an element $g \in F_2 \times F_2$

Question: Do there exist $g_1, \dots g_n \in D$ (not necessarily distinct) such that $g = g_1 g_2 \cdots g_n$ in $F_2 \times F_2$?

Let us briefly explain the NP-hardness proof, since we will reuse it. We start with a finitely presented group $\langle \Sigma, R \rangle$ having an NP-complete word problem and a polynomial Dehn function. Such a group was constructed in [7]. To this group, the following classical construction by Mihaĭlova [39] is applied: Let

$$D = \{(r^\epsilon, 1) \mid r \in R, \epsilon \in \{-1, 1\}\} \cup \{(a, a) \mid a \in \Sigma^{\pm 1}\},$$

which is viewed as a subset of $F(\Sigma) \times F(\Sigma)$. Note that D is closed under taking inverses. Let $\langle D \rangle \leq F(\Sigma) \times F(\Sigma)$ be the subgroup generated by D. Mihaĭlova proved that for every word $w \in (\Sigma^{\pm 1})^*$ the following equivalence holds:

$$w = 1 \text{ in } \langle \Sigma, R \rangle \iff (w, 1) \in \langle D \rangle \text{ in } F(\Sigma) \times F(\Sigma).$$

Moreover, based on the fact that $\langle \Sigma, R \rangle$ has a polynomial Dehn function $p(n)$, the following equivalence was shown in [41], where $q(n) = p(n) + 8(c \cdot p(n) + n)$, c is the maximal length of a relator in R, and D^n is the set of all products of n elements from D:

$$w = 1 \text{ in } \langle \Sigma, R \rangle \iff \exists n \leq q(|w|) : (w, 1) \in D^n \text{ in } F(\Sigma) \times F(\Sigma).$$

From these two equivalences it follows directly that the following three statements are equivalent for all words $w \in (\Sigma^{\pm 1})^*$, where $D = \{g_1, g_2, \dots, g_k\}$:

- $w = 1$ in $\langle \Sigma, R \rangle$
- $(w, 1) = \prod_{i=1}^{q(|w|)} (g_1^{a_{1,i}} g_2^{a_{2,i}} \cdots g_k^{a_{k,i}})$ in $F(\Sigma) \times F(\Sigma)$ for $a_{j,i} \in \{0, 1\}$
- $(w, 1) = \prod_{i=1}^{q(|w|)} (g_1^{a_{1,i}} g_2^{a_{2,i}} \cdots g_k^{a_{k,i}})$ in $F(\Sigma) \times F(\Sigma)$ for $a_{j,i} \in \mathbb{Z}$

This shows that the subset sum problem and the knapsack problem are NP-hard for the group $F(\Sigma) \times F(\Sigma)$, where for knapsack we allow integer exponents. To get the same results for $F_2 \times F_2$, we use the fact that F_2 contains a copy of $F(\Sigma)$. □

References

[1] I. Agol. The virtual Haken conjecture. Technical report, arXiv.org, 2012. `http://arxiv.org/abs/1204.2810`.

[2] R. B. J. T. Allenby and R. J. Gregorac. On locally extended residually finite groups. In *Conference on Group Theory (Univ. Wisconsin-Parkside, Kenosha, Wis., 1972)*, number 319 in Lecture Notes in Mathematics, pages 9–17. Springer, Berlin, 1973.

[3] M. Benois. Parties rationnelles du groupe libre. *C. R. Acad. Sci. Paris*, Sér. A, 269:1188–1190, 1969.

[4] A. Bertoni, G. Mauri, and N. Sabadini. Membership problems for regular and context free trace languages. *Information and Computation*, 82:135–150, 1989.

[5] M. Bestvina and N. Brady. Morse theory and finiteness properties of groups. *Inventiones Mathematicae*, 129(3):445–470, 1997.

[6] V. N. Bezverkhniĭ. On the intersection of subgroups in HNN-groups. *Fundamentalnaya i Prikladnaya Matematika*, 4(1):199–222, 1998.

[7] J.-C. Birget, A. Y. Ol′shanskii, E. Rips, and M. V. Sapir. Isoperimetric functions of groups and computational complexity of the word problem. *Annals of Mathematics. Second Series*, 156(2):467–518, 2002.

[8] M. Charikar, E. Lehman, A. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005.

[9] J. Crisp and B. Wiest. Embeddings of graph braid and surface groups in right-angled Artin groups and braid groups. *Algebraic & Geometric Topology*, 4:439–472, 2004.

[10] W. Dicks and M. J. Dunwoody. *Groups Acting on Graphs*. Cambridge University Press, 1989.

[11] V. Diekert. *Combinatorics on Traces*, volume 454 of *Lecture Notes in Computer Science*. Springer, 1990.

[12] V. Diekert and J. Kausch. Logspace computations in graph products. *Journal of Symbolic Computation*, 2015. to appear.

[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 A. Muscholl. Solvability of equations in graph groups is decidable. *International Journal of Algebra and Computation*, 16(6):1047–1069, 2006.

[15] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.

[16] M. Elberfeld, A. Jakoby, and T. Tantau. Algorithmic meta theorems for circuit classes of constant and logarithmic depth. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:128, 2011.

[17] E. Frenkel, A. Nikolaev, and A. Ushakov. Knapsack problems in products of groups. *Journal of Symbolic Computation*, 2014. to appear.

[18] R. Ghrist and V. Peterson. The geometry and topology of reconfiguration. *Advances in Applied Mathematics*, 38(3):302–323, 2007.

[19] C. Haase. *On the complexity of model checking counter automata*. PhD thesis, University of Oxford, St Catherine's College, 2011.

[20] F. Haglund and D. T. Wise. Coxeter groups are virtually special. *Advances in Mathematics*, 224(5):1890–1903, 2010.

[21] N. Haubold and M. Lohrey. Compressed word problems in HNN-extensions and amalgamated products. *Theory of Computing Systems*, 49(2):283–305, 2011.

[22] G. Higman, B. H. Neumann, and H. Neumann. Embedding theorems for groups. *Journal of the London Mathematical Society. Second Series*, 24:247–254, 1949.

[23] B. Jenner. Knapsack problems for NL. *Information Processing Letters*, 54(3):169–174, 1995.

[24] M. Kambites, P. V. Silva, and B. Steinberg. On the rational subset problem for groups. *Journal of Algebra*, 309(2):622–639, 2007.

[25] I. Kapovich, R. Weidmann, and A. Myasnikov. Foldings, graphs of groups and the membership problem. *International Journal of Algebra and Computation*, 15(1):95–128, 2005.

[26] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[27] A. Karrass and D. Solitar. The subgroups of a free product of two groups with an amalgamated subgroup. *Transactions of the American Mathematical Society*, 150:227–255, 1970.

[28] A. Karrass and D. Solitar. Subgroups of HNN groups and groups with one defining relation. *Canadian Journal of Mathematics*, 23:627–643, 1971.

[29] D. König, M. Lohrey, and G. Zetzsche. Knapsack and subset sum problems in nilpotent, polycyclic, and co-context-free groups. Technical report, arXiv.org, 2015. `http://arxiv.org/abs/1507.05145`.

[30] D. Kuske and M. Lohrey. Logical aspects of Cayley-graphs: the monoid case. *International Journal of Algebra and Computation*, 16(2):307–340, 2006.

[31] M. Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.

[32] M. Lohrey. *The Compressed Word Problem for Groups*. SpringerBriefs in Mathematics. Springer, 2014.

[33] M. Lohrey and S. Schleimer. Efficient computation in groups via compression. In *Proceedings of Computer Science in Russia, CSR 2007*, volume 4649 of *Lecture Notes in Computer Science*, pages 249–258. Springer, 2007.

[34] M. Lohrey and G. Sénizergues. Theories of HNN-extensions and amalgamated products. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP'06)*, number 4052 in Lecture Notes in Computer Science, pages 504–515. Springer, 2006.

[35] M. Lohrey and G. Sénizergues. Rational subsets in HNN-extensions and amalgamated products. *International Journal of Algebra and Computation*, 18(1):111–163, 2008.

[36] M. Lohrey and B. Steinberg. The submonoid and rational subset membership problems for graph groups. *Journal of Algebra*, 320(2):728–755, 2008.

[37] R. C. Lyndon and P. E. Schupp. *Combinatorial Group Theory*. Springer, 1977.

[38] V. Metaftsis and E. Raptis. Subgroup separability of graphs of abelian groups. *Proceedings of the American Mathematical Society*, 132:1873–1884, 2004.

[39] K. A. Mihaĭlova. The occurrence problem for direct products of groups. *Math. USSR Sbornik*, 70:241–251, 1966. English translation.

[40] A. Muscholl and D. Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In *Proceedings of the 24th International Symposium on Mathematical Foundations of Computer Science (MFCS'99)*, number 1672 in Lecture Notes in Computer Science, pages 81–91. Springer, 1999.

[41] A. Myasnikov, A. Nikolaev, and A. Ushakov. Knapsack problems in groups. *Mathematics of Computation*, 84:987–1016, 2015.

[42] C. H. Papadimitriou. On the complexity of integer programming. *Journal of the Association for Computing Machinery*, 28(4):765–768, 1981.

[43] J. R. Stallings. *Group Theory and Three-Dimensional Manifolds*. Number 4 in Yale Mathematical Monographs. Yale University Press, 1971.

[44] A. W. To. Unary finite automata vs. arithmetic progressions. *Information Processing Letters*, 109(17):1010–1014, 2009.

[45] J. von zur Gathen and M. Sieveking. A bound on solutions of linear integer equalities and inequalities. *Proceedings of the American Mathematical Society*, 72(1):155–158, 1978.

[46] D. T. Wise. Research announcement: the structure of groups with a quasiconvex hierarchy. *Electronic Research Announcements in Mathematical Sciences*, 16:44–55, 2009.