TC^0 Circuits for Algorithmic Problems in Nilpotent Groups

Alexei Myasnikov[1] and Armin Weiß[2]

1 **Stevens Institute of Technology, Hoboken, NJ, USA**
2 **Universität Stuttgart, Germany**

Abstract

Recently, Macdonald et. al. showed that many algorithmic problems for finitely generated nilpotent groups including computation of normal forms, the subgroup membership problem, the conjugacy problem, and computation of subgroup presentations can be done in LOGSPACE. Here we follow their approach and show that all these problems are complete for the uniform circuit class TC^0 – uniformly for all r-generated nilpotent groups of class at most c for fixed r and c.

Moreover, if we allow a certain binary representation of the inputs, then the word problem and computation of normal forms is still in uniform TC^0, while all the other problems we examine are shown to be TC^0-Turing reducible to the problem of computing greatest common divisors and expressing them as linear combinations.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.0 Discrete Mathematics

Keywords and phrases nilpotent groups, TC^0, abelian groups, word problem, conjugacy problem, subgroup membership problem, greatest common divisors

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.23

1 Introduction

The word problem (given a word over the generators, does it represent the identity?) is one of the fundamental algorithmic problems in group theory introduced by Dehn in 1911 [3]. While for general finitely presented groups all these problems are undecidable [22, 2], for many particular classes of groups decidability results have been established – not just for the word problem but also for a wide range of other problems. Finitely generated nilpotent groups are a class where many algorithmic problems are (efficiently) decidable (with some exceptions like the problem of solving equations – see e. g. [6]). In 1958, Mal'cev [17] established decidability of the word and subgroup membership problem by investigating finite approximations of nilpotent groups. In 1965, Blackburn [1] showed decidability of the conjugacy problem. However, these methods did not allow any efficient (e. g. polynomial time) algorithms. Nevertheless, in 1966 Mostowski provided "practical" algorithms for the word problem and several other problems [18]. In terms of complexity, a major step was the result by Lipton and Zalcstein [15] that the word problem of linear groups is in LOGSPACE. Together with the fact that finitely generated nilpotent groups are linear (see e. g. [7, 10]) this gives a LOGSPACE solution to the word problem of nilpotent groups, which was later improved to uniform TC^0 by Robinson [23]. A typical algorithmic approach to nilpotent groups is using so-called Mal'cev (or Hall–Mal'cev) bases (see e. g. [7, 10]), which allow to carry out group operations by evaluating polynomials (see Lemma 2). This approach was systematically used in [11] and [18] or – in the more general setting of polycyclic presentations – in [24] for solving (among others) the subgroup membership and conjugacy

problem of polycyclic groups. Recently in [19, 20] polynomial time bounds for the equalizer and subgroup membership problems in nilpotent groups have been given. Finally, in [16] the following problems were shown to be in LOGSPACE using the Mal'cev basis approach. Here, $\mathcal{N}_{c,r}$ denotes the class of nilpotent groups of nilpotency class at most c generated by at most r elements.

- The *word problem*: given $G \in \mathcal{N}_{c,r}$ and $g \in G$, is $g = 1$ in G?
- Given $G \in \mathcal{N}_{c,r}$ and $g \in G$, compute the (Mal'cev) normal form of g.
- The *subgroup membership problem*: Given $G \in \mathcal{N}_{c,r}$ and $g, h_1, \ldots, h_n \in G$, decide whether $g \in \langle h_1, \ldots, h_n \rangle$ and, if so, express g as a word over the subgroup generators $h_1, \ldots, h_n$ (in [16] only the decision version was shown to be in LOGSPACE – for expressing g as a word over the original subgroup generators a polynomial time bound was given).
- Given $G, H \in \mathcal{N}_{c,r}$ and $K = \langle g_1, \ldots, g_n \rangle \leq G$, together with a homomorphism $\varphi : K \to H$ specified by $\varphi(g_i) = h_i$, and some $h \in \mathrm{Im}(\varphi)$, compute a generating set for $\ker(\varphi)$ and find $g \in G$ such that $\varphi(g) = h$.
- Given $G \in \mathcal{N}_{c,r}$ and $K = \langle g_1, \ldots, g_n \rangle \leq G$, compute a presentation for K.
- Given $G \in \mathcal{N}_{c,r}$ and $g \in G$, compute a generating set for the centralizer of g.
- The *conjugacy problem*: Given $G \in \mathcal{N}_{c,r}$ and $g, h \in G$, decide whether or not there exists $u \in G$ such that $u^{-1}gu = h$ and if so find such an element u.

Notice that these problems are not only of interest in themselves, but also might serve as building blocks for solving the same problems in polycyclic groups – which are of particular interest because of their possible application in non-commutative cryptography [4]. In this work we follow [16] and extend these results in several ways:

- We give a complexity bound of uniform TC^0 for all the above problems.
- In order to derive this bound, we show that the extended gcd problem with unary coefficients is in TC^0.
- Our description of circuits is for the uniform setting where $G \in \mathcal{N}_{c,r}$ is part of the input (in [16] the uniform setting is also considered; however, only in some short remarks).
- Since nilpotent groups have polynomial growth, it is natural to allow compressed inputs: we give a uniform TC^0 solution for the word problem allowing words with binary exponents as input – this contrasts with the situation with straight-line programs (i. e., context-free grammars which produces precisely one word – another method of exponential compression) as input: then the word problem is hard for $\mathsf{C}_=\mathsf{L}$ [12]. Thus, the difficulty of the word problem with straight-line programs is not due to their compression but rather due to the difficulty of evaluating a straight-line program.
- We show that the other of the above problems are uniform-TC^0-Turing-reducible to the extended gcd problem (compute the greatest common divisor and express it as a linear combination) when the inputs (both the ambient group and the subgroup etc.) are given as words with binary exponents.

Thus, in the unary case we settle the complexity of the above problems completely. Moreover, it also seems rather unlikely that the subgroup membership problem can be solved without computing gcds – in this case our results on binary inputs would be also optimal. Altogether, our results mean that many algorithmic problems are no more complicated in nilpotent groups than in abelian groups. Notice that while in [16] explicit length bounds on the outputs for all these problems are proven, we obtain polynomial length bounds simply by the fact that everything can be computed in uniform TC^0 (for which in the following we only write TC^0). Throughout the paper we follow the outline of [16]. For a concise presentation, we copy many definitions from [16]. Most of our theorems involve two statements: one for unary encoded inputs and one for binary encoded inputs. In order to have a concise presentation, we always put them in one statement. We only consider finitely generated nilpotent groups without mentioning that further.

Outline. We start with basic definitions on complexity as well as on nilpotent groups. In Section 2 we describe how subgroups of nilpotent groups can be represented and develop a "nice" presentation for all groups in $\mathcal{N}_{c,r}$. Section 3 deals with the word problem and computation of normal forms. Based on this we introduce the so-called matrix reduction and solve the subgroup membership problem. Finally, in Section 5 we present our result for the remaining of the above problems – the proofs are essentially repeated applications of the matrix reduction. Due to space constraints many of the proofs are omitted – they can be found in the full version on arXiv [21].

1.1 Preliminaries on Complexity

For a finite *alphabet* Σ, the set of *words* over Σ is denoted by Σ^*. Computation or decision problems are given by functions $f : \Delta^* \to \Sigma^*$ for some finite alphabets Δ and Σ. A decision problem (= formal language) L is identified with its characteristic function $\chi_L : \Delta^* \to \{0, 1\}$ with $\chi_L(x) = 1$ if, and only if, $x \in L$. (In particular, the word and conjugacy problems can be seen as functions $\Sigma^* \to \{0, 1\}$.) We use circuit complexity as described in [25].

Circuit Classes. The class TC^0 is defined as the class of functions computed by families of circuits of constant depth and polynomial size with unbounded fan-in Boolean gates (and, or, not) and majority gates. A majority gate (denoted by Maj) returns 1 if the number of 1s in its input is greater or equal to the number of 0s. In the following we always assume that the alphabets Δ and Σ are encoded over the binary alphabet $\{0, 1\}$ such that each letter uses the same number of bits. We say a function f is TC^0-*computable* if $f \in \mathsf{TC}^0$.

In the following, we only consider Dlogtime-uniform circuit families and we simply write TC^0 as shorthand for Dlogtime-uniform TC^0. Dlogtime-uniform means that there is a deterministic Turing machine which decides in time $\mathcal{O}(\log n)$ on input of two gate numbers (given in binary) and the string 1^n whether there is a wire between the two gates in the n-input circuit and also computes of which type some gates is. Note that the binary encoding of the gate numbers requires only $\mathcal{O}(\log n)$ bits – thus, the Turing machine is allowed to use time linear in the length of the encodings of the gates. For more details on these definitions we refer to [25]. We have the inclusions $\mathsf{AC}^0 \subsetneq \mathsf{TC}^0 \subseteq \mathsf{LOGSPACE} \subseteq \mathsf{P}$ (note that even $\mathsf{TC}^0 \subseteq \mathsf{P}$ is not known to be strict).

Reductions. A function f is TC^0-*Turing-reducible* to a function g if there is a Dlogtime-uniform family of TC^0 circuits computing f which, in addition to the Boolean and majority gates, also may use oracle gates for g (i. e., gates which on input x output $g(x)$). This is expressed by $f \in \mathsf{TC}^0(g)$. Note that if $f_1, \ldots, f_k$ are in TC^0, then $\mathsf{TC}^0(f_1, \ldots, f_k) = \mathsf{TC}^0$.

In particular, if f and g are TC^0-computable functions, then also the composition $g \circ f$ is TC^0-computable. We will extensively make use of this observation – which will also guarantee the polynomial size bound on the outputs of our circuits without additional calculations.

We will also use another fact frequently without giving further reference: on input of two alphabets Σ and Δ (coded over the binary alphabet), a list of pairs (a, v_a) with $a \in \Sigma$ and $v_a \in \Delta^*$ such that each $a \in \Sigma$ occurs in precisely one pair, and a word $w \in \Sigma^*$, the image $\varphi(w)$ under the homomorphism φ defined by $\varphi(a) = v_a$ can be computed in TC^0 [13].

Encoding numbers: unary vs. binary. There are essentially two ways of representing integer numbers: the usual way as a binary number where a string $a_0 \cdots a_n$ with $a_i \in \{0, 1\}$ represents $\sum a_i 2^{n-i}$, and as a unary number where $k \in \mathbb{N}$ is represented by $1^k = \underbrace{11 \cdots 1}_{k}$ (respectively by $0^{n-k}1^k$ if n is the number of input bits).

We will state most results in this paper with both representations. The unary representation corresponds to group elements given as words over the generators, whereas the binary encoding will be used if inputs are given in a compressed form.

Arithmetic in TC^0. Iterated Addition (resp. Iterated Multiplication) are the following computation problems: On input of n binary integers $a_1, \ldots, a_n$ each having n bits (i. e., the input length is $N = n^2$), compute the binary representation of the sum $\sum_{i=1}^{n} a_i$ (resp. product $\prod_{i=1}^{n} a_i$). For Integer Division the input are two binary n-bit integers a, b; the binary representation of the integer $c = \lfloor a/b \rfloor$ has to be computed. The first statement of Theorem 1 is a standard fact, see [25]; the other statements are due to Hesse, [8, 9].

▶ **Theorem 1** ([8, 9, 25]). *The problems* Iterated Addition, Iterated Multiplication, Integer Division *are all in* TC^0 *no matter whether inputs are given in unary or binary.*

Note that if the numbers a and b are encoded in unary (as strings 1^a and 1^b), division can be seen to be in TC^0 very easily: just try for all $0 \leq c \leq a$ whether $0 \leq a - bc < b$.

Representing groups for algorithmic problems. We consider finitely generated groups G together with finite generating sets A. Group elements are represented as words over the generators and their inverses (i. e., as elements of $(A \cup A^{-1})^*$). We make no distinction between words and the group elements they represent. Whenever it might be unclear whether we mean equality of words or of group elements, we write "$g = h$ in G" for equality in G.

Words over the generators ± 1 of $\mathbb{Z}$ correspond to unary representation of integers. As a generalization of binary encoded integers, we introduce the following notion: a *word with binary exponents* is a sequence $w_1, \ldots, w_n$ where the w_i are from a fixed generating set of the group together with a sequence of exponents $x_1, \ldots, x_n$ where the $x_i \in \mathbb{Z}$ are encoded in binary. The word with binary exponents represents the word (or group element) $w = w_1^{x_1} \cdots w_n^{x_n}$. Note that in a fixed nilpotent group *every* word of length n can be rewritten as a word with binary exponents using $\mathcal{O}(\log n)$ bits (this fact is well-known and also a consequence of Theorem 5 below); thus, words with binary exponents are a natural way of representing inputs for algorithmic problems in nilpotent groups.

1.2 Preliminaries on Nilpotent groups and Mal'cev coordinates

Let G be a group. For $x, y \in G$ we write $[x, y] = x^{-1}y^{-1}xy$ for the *commutator* of x and y. For subgroups $H_1, H_2 \leq G$, we have $[H_1, H_2] = \langle \{[h_1, h_2] \mid h_1 \in H_1, h_2 \in H_2\} \rangle$. A group G is called *nilpotent* if it has a finite central series, i.e.

$$G = G_1 \geq G_2 \geq \cdots \geq G_c \geq G_{c+1} = 1 \tag{1}$$

such that $[G, G_i] \leq G_{i+1}$ for all $i = 1, \ldots, c$. If G is finitely generated, so are the abelian quotients G_i/G_{i+1}, $1 \leq i \leq c$. Let $a_{i1}, \ldots, a_{im_i}$ be a basis of G_i/G_{i+1}, i.e. a generating set such that G_i/G_{i+1} has a presentation $\left\langle a_{i1}, \ldots, a_{im_i} \,\middle|\, a_{ij}^{e_{ij}}, [a_{ik}, a_{i\ell}], \text{ for } j \in \mathcal{T}_i,\ k, \ell \in \{1, \ldots, m_i\} \right\rangle$, where $\mathcal{T}_i \subseteq \{1, \ldots, m_i\}$ (here $\mathcal{T}$ stands for torsion) and $e_{ij} \in \mathbb{Z}_{>0}$ (be aware that we explicitly allow $e_{ij} = 1$, which is necessary for our definition of quotient presentations in Section 2). Formally, we put $e_{ij} = \infty$ for $j \notin \mathcal{T}_i$. We call $A = (a_{11}, a_{12}, \ldots, a_{cm_c})$ a *Mal'cev basis associated to the central series* (1). Sometimes we use A interchangeably also for the set $A = \{a_{11}, a_{12}, \ldots, a_{cm_c}\}$.

For convenience, we will also use a simplified notation, in which the generators a_{ij} and exponents e_{ij} are renumbered by replacing each subscript ij with $j + \sum_{\ell<j} m_\ell$, so the generating

sequence A can be written as $A = (a_1, \ldots, a_m)$. We allow the expression ij to stand for $j + \sum_{\ell<j} m_\ell$ in other notations as well. We also denote $\mathcal{T} = \{i \mid e_i < \infty\}$. By the choice of $\{a_1, \ldots, a_m\}$, every element $g \in G$ may be written uniquely in the form $g = a_1^{\alpha_1} \cdots a_m^{\alpha_m}$, where $\alpha_i \in \mathbb{Z}$ and $0 \le \alpha_i < e_i$ whenever $i \in \mathcal{T}$. The m-tuple $(\alpha_1, \ldots, \alpha_m)$ is called the *coordinate vector* or *Mal'cev coordinates* of g and is denoted $\mathrm{Coord}(g)$, and the expression $a_1^{\alpha_1} \cdots a_m^{\alpha_m}$ is called the *(Mal'cev) normal form* of g. We also denote $\alpha_i = \mathrm{Coord}_i(g)$.

To a Mal'cev basis A we associate a presentation of G as follows. For each $1 \le i \le m$, let n_i be such that $a_i \in G_{n_i} \setminus G_{n_i+1}$. If $i \in \mathcal{T}$, then $a_i^{e_i} \in G_{n_i+1}$, hence a relation

$$a_i^{e_i} = a_\ell^{\mu_{i\ell}} \cdots a_m^{\mu_{im}} \tag{2}$$

holds in G for $\mu_{ij} \in \mathbb{Z}$ and $\ell > i$ such that $a_\ell, \ldots, a_m \in G_{n_i+1}$. We call this the *power relation* for a_i. Let $1 \le i < j \le m$. Since the series (1) is central, relations of the form

$$a_j a_i = a_i a_j a_\ell^{\alpha_{ij\ell}} \cdots a_m^{\alpha_{ijm}} \qquad\qquad a_j^{-1} a_i = a_i a_j^{-1} a_\ell^{\beta_{ij\ell}} \cdots a_m^{\beta_{ijm}} \tag{3}$$

hold in G for $\alpha_{ijk}, \beta_{ijk} \in \mathbb{Z}$ and $l > j$ such that $a_\ell, \ldots, a_m \in G_{n_j+1}$. Now, G is the group with generators $\{a_1, \ldots, a_m\}$ subject to the relation of the the form (2)–(3).

A presentation with relations of the form (2)–(3) for all i resp. i and j is called a *nilpotent presentation*. Indeed, any presentation of this form will define a nilpotent group. It is called *consistent* if the order of a_i modulo $\langle a_{i+1}, \ldots, a_m \rangle$ is precisely e_i for all i. While presentations of this form need not, in general, be consistent, those derived from a central series of a group G as above are consistent. Given a consistent nilpotent presentation, there is an easy way to solve the word problem: simply apply the rules of the form (3) to move all occurrences of $a_1^{\pm 1}$ in the input word to the left, then apply the power relations (2) to reduce their number modulo e_1; finally, continue with a_2 and so on.

Multiplication functions. An crucial feature of the coordinate vectors for nilpotent groups is that the coordinates of a product $(a_1^{\alpha_1} \cdots a_m^{\alpha_m})(a_1^{\beta_1} \cdots a_m^{\beta_m})$ may be computed as a "nice" function (polynomial if $\mathcal{T} = \emptyset$) of the integers $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m$.

▶ **Lemma 2** ([7, 10]). *Let G be a nilpotent group with Mal'cev basis $a_1, \ldots, a_m$ and $\mathcal{T} = \emptyset$. There exist $p_1, \ldots, p_m \in \mathbb{Z}[x_1, \ldots, x_m, y_1, \ldots, y_m]$ and $q_1, \ldots, q_m \in \mathbb{Z}[x_1, \ldots, x_m, z]$ such that for $g, h \in G$ with $\mathrm{Coord}(g) = (\gamma_1, \ldots, \gamma_m)$ and $\mathrm{Coord}(h) = (\delta_1, \ldots, \delta_m)$ and $l \in \mathbb{Z}$ we have*

- **(i)** $\mathrm{Coord}_i(gh) = p_i(\gamma_1, \ldots, \gamma_m, \delta_1, \ldots, \delta_m)$,
- **(ii)** $\mathrm{Coord}_i(g^l) = q_i(\gamma_1, \ldots, \gamma_m, l)$,
- **(iii)** $\mathrm{Coord}_1(gh) = \gamma_1 + \delta_1$ *and* $\mathrm{Coord}_1(g^l) = l\gamma_1$.

Notice that an explicit algorithm to construct the polynomials p_i, q_i is given in [14]. For further background on nilpotent groups we refer to [7, 10].

2 Presentation of subgroups

Before we start with algorithmic problems, we introduce a canonical way how to represent subgroups of nilpotent groups. This is important for two reasons: first, of course we need it to solve the subgroup membership problem, and, second, for the uniform setting it allows us to represent nilpotent groups as free nilpotent group modulo a kernel which is represented as a subgroup. Let $h_1, \ldots, h_n$ be elements of G given in normal form by $h_i = a_1^{\alpha_{i1}} \cdots a_m^{\alpha_{im}}$, for

$i = 1, \ldots, n$, and let $H = \langle h_1, \ldots, h_n \rangle$. We associate the *matrix of coordinates*

$$A = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1m} \\ \vdots & \ddots & \vdots \\ \alpha_{n1} & \cdots & \alpha_{nm} \end{pmatrix}, \tag{4}$$

to the tuple $(h_1, \ldots, h_n)$ and conversely, to any $n \times m$ integer matrix, we associate an n-tuple of elements of G, whose Mal'cev coordinates are given as the rows of the matrix, and the subgroup H generated by the tuple. For each $i = 1, \ldots, n$ where row i is non-zero, let π_i be the column of the first non-zero entry ('pivot') in row i. The sequence $(h_1, \ldots, h_n)$ is said to be in *standard form* if the matrix of coordinates A is in row-echelon form and its pivot columns are maximally reduced, more specifically, if A satisfies the following properties:

(i) all rows of A are non-zero (i.e. no h_i is trivial),
(ii) $\pi_1 < \pi_2 < \cdots < \pi_s$ (where s is the number of pivots),
(iii) $\alpha_{i\pi_i} > 0$, for all $i = 1, \ldots, n$,
(iv) $0 \leq \alpha_{k\pi_i} < \alpha_{i\pi_i}$, for all $1 \leq k < i \leq s$
(v) if $\pi_i \in \mathcal{T}$, then $\alpha_{i\pi_i}$ divides e_{π_i}, for $i = 1, \ldots, s$.

The sequence (resp. matrix) is called *full* if in addition

(vi) $H \cap \langle a_i, a_{i+1}, \ldots, a_m \rangle$ is generated by $\{h_j \mid \pi_j \geq i\}$, for all $1 \leq i \leq m$.

Note that $\{h_j \mid \pi_j \geq i\}$ consists of those elements having 0 in their first $i-1$ coordinates. It is an easy exercise (see also [16]) to show that 6 holds for a given i if and only if

- for all $1 \leq k < j \leq s$ with $\pi_k < i$, $h_k^{-1} h_j h_k$ and $h_k h_j h_k^{-1}$ are elements of $\langle\, h_l \mid l > k \,\rangle$, and
- for all $1 \leq k \leq s$ with $\pi_k < i$ and $\pi_k \in \mathcal{T}$, $h_k^{e_{\pi_k}/\alpha_{k\pi_k}} \in \langle\, h_l \mid l > k \,\rangle$.

We will use full sequences and the associated matrices in full form interchangeably without mentioning it explicitly. For simplicity we assume that the inputs of algorithms are given as matrices. The importance of full sequences is described in the following lemma – a proof can be found in [24] Propositions 9.5.2 and 9.5.3.

▶ **Lemma 3** ([16, Lem. 3.1]). *Let $H \leq G$. There is a unique full sequence $U = (h_1, \ldots, h_s)$ that generates H. We have $s \leq m$ and $H = \{h_1^{\beta_1} \cdots h_s^{\beta_s} \mid \beta_i \in \mathbb{Z}$ and $0 \leq \beta_i < e_{\pi_i}$ if $\pi_i \in \mathcal{T}\}$.*

Thus, computing a full sequence will be the essential tool for solving the subgroup membership problem. Before we focus on subgroup membership, we will first solve the word problem and introduce how the nilpotent group can be part of the input.

Quotient Mal'cev presentations. Let $c, r \in \mathbb{N}$ be fixed. The free nilpotent group $F_{c,r}$ of class c and rank r is defined as $F_{c,r} = \langle\, a_1, \ldots, a_r \mid [x_1, \ldots, x_{c+1}] = 1 \text{ for } x_1, \ldots, x_{c+1} \in F_{c,r} \,\rangle$ where $[x_1, \ldots, x_{c+1}] = [[x_1, \ldots, x_c], x_{c+1}]$, i. e., $F_{c,r}$ is the r-generated group only subject to the relations that weight $c+1$ commutators are trivial. Throughout, we fix a Mal'cev basis $A = (a_1, \ldots, a_m)$ (which we call the *standard Mal'cev basis*) associated to the lower central series of $F_{c,r}$ such that the associated nilpotent presentation consists only of relations of the form (3) (i. e., $\mathcal{T} = \emptyset$ – such a presentation exists since $F_{c,r}$ is torsion-free), $a_1, \ldots, a_r$ generates $F_{c,r}$, and all other Mal'cev generators are iterated commutators of $a_1, \ldots, a_r$.

Denote by $\mathcal{N}_{c,r}$ the set of r-generated nilpotent groups of class at most c. Every group $G \in \mathcal{N}_{c,r}$ is a quotient of the free nilpotent group $F_{c,r}$, i. e., $G = F_{c,r}/N$ for some normal subgroup $N \leq F_{c,r}$. Assume that $T = (h_1, \ldots, h_s)$ is a full sequence generating N. Adding T to the set of relators of the free nilpotent group yields a new nilpotent presentation. This presentation will be called *quotient presentation* of G. For inputs of algorithms, we assume that a quotient presentation is always given as its matrix of coordinates in full form.

Depending whether the entries of the matrix are encoded in unary or binary, we call the quotient presentation be given in *unary* or *binary*.

▶ **Lemma 4** ([16, Prop. 5.1]). *Let c and r be fixed integers and let $A = (a_1, \dots, a_m)$ be the standard Mal'cev basis of $F_{c,r}$. Moreover, denote by S the set of relators of $F_{c,r}$ with respect to A. Let $G \in \mathcal{N}_{c,r}$ with $G = F_{c,r}/N$ and let T be the full-form sequence for the subgroup N of $F_{c,r}$. Then, $\langle A \mid S \cup T \rangle$ is a consistent nilpotent presentation of G.*

For a proof of Lemma 4 see [21]. For the following we always assume that a quotient presentation is part of the input, but c and r are fixed. Later, we will show how to compute quotient presentations from an arbitrary presentation.

▶ Remark. Lemma 4 ensures that each group element has a unique normal form with respect to the quotient presentation; thus, it guarantees that all our manipulations of Mal'cev coordinates are well-defined.

3 Word problem and computation of Mal'cev coordinates

In this section we deal with the word problem of nilpotent groups, which is well-known to be in TC^0 [23]. Here, we generalize this result by allowing words with binary exponents (recall that *word with binary exponents* is a sequence $w = w_1^{x_1} \cdots w_n^{x_n}$ where $w_i \in \{a_1, \dots, a_m\}$ and the $x_i \in \mathbb{Z}$). By using words with binary exponents the input can be compressed exponentially – making the word problem, a priori, harder to solve. Nevertheless, it turns out that the word problem still can be solved in TC^0 when allowing the input to be given as a word with binary exponents. Note that this contrasts with the situation where the input is given as straight-line program (which like words with binary exponents allow an exponential compression) – then the word problem is complete for the counting class $\mathsf{C_=L}$ [12].

▶ **Theorem 5.** *Let $c, r \geq 1$ be fixed and let $(a_1, \dots, a_m)$ be the standard Mal'cev basis of $F_{c,r}$. The following problem is TC^0-complete: on input of $G \in \mathcal{N}_{c,r}$ given as a binary encoded quotient presentation and a word with binary exponents $w = w_1^{x_1} \cdots w_n^{x_n}$, compute integers $y_1, \dots, y_m$ (in binary) such that $w = a_1^{y_1} \cdots a_m^{y_m}$ in G and $0 \leq y_i < e_i$ for $i \in \mathcal{T}$. Moreover, if the input is given in unary (both G and w), then the output is in unary.*

Note that the statement for unary inputs is essentially the one of [23]. Be aware that in the formulation of the theorem, $\mathcal{T}$ and e_i for $i \in \mathcal{T}$ depend on the input group G. These parameters can be read from the full matrix of coordinates representing G (recall that π_i denotes the column index of the i-th pivot): $\mathcal{T} = \{\pi_i \mid i \in \{1, \dots, n\}\}$ (all columns which have a pivot) and $e_i = \alpha_{ji}$ if $\pi_j = i$. As an immediate consequence of Theorem 5, we obtain:

▶ **Corollary 6.** *Let $c, r \geq 1$ be fixed. The uniform, binary version of the word problem for groups in $\mathcal{N}_{c,r}$ is TC^0-complete (where the input is given as in Theorem 5).*

The proof of Theorem 5 follows the outline given in Section 1.2; however, we cannot apply the rules (2)–(3) one by one. Instead we do only two steps for each generator: first apply all possible rules (3) in one step and then apply the rules (2) in one step.

Proof of Theorem 5. The hardness part is clear since already the word problem of $\mathbb{Z}$ is TC^0-complete. For describing a TC^0 circuit, we proceed by induction along the standard Mal'cev basis $(a_1, \dots, a_m)$ of the free nilpotent group $F_{c,r}$. If w does not contain any letter a_1, we have $y_1 = 0$ and we can compute y_i for $i > 1$ by induction.

Otherwise, we rewrite w as $a_1^{y_1}uv$ (with $0 \leq y_1 < e_1$ if $1 \in \mathcal{T}$) such that u and v are words with binary exponents not containing any a_1s. Once this is completed, uv can be rewritten as $a_2^{y_2} \cdots a_m^{y_m}$ by induction. For computing y_1, u and v, we proceed in two steps:

First, we rewrite w as $a_1^{\tilde{y}_1}v$ with $\tilde{y}_1 = \sum_{w_i=a_1} x_i$ (this is possible by Lemma 2 (iii)). The exponent $\tilde{y}_1$ can be computed by iterated addition, which by Theorem 1 is in TC^0 (in the unary case $\tilde{y}_1$ can be written down in unary). Now, v consists of what remains from w after a_1 has been "eliminated": for every position i in w with $w_i \neq a_1$, we compute $z_i = \sum_{w_j=a_1}^{j>i} x_j$ using iterated addition. Let $w_i = a_k$. By Lemma 2 (i) there are fixed polynomials $p_{k,k+1}, \ldots, p_{k,m} \in \mathbb{Z}[x,y]$ such that in the free nilpotent group holds $a_k^x a_1^y = a_1^y a_k^x a_{k+1}^{p_{k,k+1}(x,y)} \cdots a_m^{p_{k,m}(x,y)}$ for all $x, y \in \mathbb{Z}$. Hence, in order to obtain $\tilde{w}$, it remains to replace every $w_i^{x_i}$ with $w_i = a_1$ by the empty word and every $w_i^{x_i}$ with $w_i = a_k \neq a_1$ by $a_k^{x_i} a_{k+1}^{p_{k,k+1}(x_i,z_i)} \cdots a_m^{p_{k,m}(x_i,z_i)}$, which is a word with binary exponents (resp. as a word of polynomial length in the unary case), for $k = 2, \ldots, m$. The exponents can be computed in TC^0 by Theorem 1. Since the $p_{k,i}$ are bounded by polynomials, in the unary case, $a_k^{x_i} a_{k+1}^{p_{k,k+1}(x_i,z_i)} \cdots a_m^{p_{k,m}(x_i,z_i)}$ can be written as a word without exponents.

The second step is only applied if $1 \in \mathcal{T}$ (as explained above, this can be decided and e_i can be read directly from the quotient presentation by checking whether there is a pivot in the first column) – otherwise $y_1 = \tilde{y}_1$ and u is the empty word. We rewrite $a_1^{\tilde{y}_1}$ to $a_1^{y_1}u$ with $y_1 = \tilde{y}_1 \bmod e_1$ and a word with binary exponents u not containing any a_1. Again y_1 can be computed in TC^0 by Theorem 1. Let $a_1^{e_1} = a_2^{\mu_{12}} \cdots a_m^{\mu_{1m}}$ be the power relation for a_1 (which can be read from the quotient presentation – it is just the row where the pivot is in the first column) and write $\tilde{y}_1 = s \cdot e_1 + y_1$. Now, u should be equal to $(a_2^{\mu_{12}} \cdots a_m^{\mu_{1m}})^s$ in $F_{c,r}$. We use the fixed polynomials $q_i \in \mathbb{Z}[x_1, \ldots, x_m, z]$ from Lemma 2 (ii) for $F_{c,r}$ yielding $u = a_2^{q_2(0,\mu_{12},\ldots,\mu_{1m},s)} \cdots a_m^{q_m(0,\mu_{12},\ldots,\mu_{1m},s)}$ (which, in the binary setting, is a word with binary exponents, and in the unary setting a word without exponents of polynomial length). Now, we have $w = a_1^{y_1}uv$ in G as desired. ◀

4 Matrix reduction and subgroup membership problem

Before we solve the subgroup membership problem, let us take a look at one essential step, namely the problem of computing greatest common divisors. Indeed, consider the nilpotent group $\mathbb{Z}$ and let $a, b, c \in \mathbb{Z}$. Then $c \in \langle a, b \rangle$ if, and only if, $\gcd(a, b) \mid c$.

Binary gcds. The *extended gcd problem* (ExtGCD) is the following problem: on input of binary encoded numbers $a_1, \ldots, a_n \in \mathbb{Z}$, compute $x_1, \ldots, x_n \in \mathbb{Z}$ such that $x_1a_1 + \cdots + x_na_n = \gcd(a_1, \ldots, a_n)$. Clearly this can be done in P using the Euclidean algorithm, but it is not known whether it is actually in NC. Since we need to compute greatest common divisors, we will reduce the subgroup membership problem to the computation of gcds.

Unary gcds. Computing the gcd of numbers encoded in unary is straightforward in TC^0 by an exhaustive search. Also, for just two numbers $a, b \in \mathbb{Z}$ the gcd easily can be expressed as a linear combination in TC^0: there are $x, y \leq \max\{|a|, |b|\}$ such that $ax + by = \gcd(a, b)$. Now, x, y can be computed in TC^0 by simply checking all values with $|x|, |y| \leq \max\{|a|, |b|\}$. Similarly, there are $x_1, \ldots, x_n \leq |\max\{|a_1|, \ldots, |a_n|\}|$ with $x_1a_1 + \cdots + x_na_n = \gcd(a_1, \ldots, a_n)$. However, for computing these x_i, we cannot check all possible combinations of values in TC^0 because there are $|\max\{|a_1|, \ldots, |a_n|\}|^n$ (i. e., exponentially) many. Expressing the gcd as a linear combination can be viewed as a linear equation with integral coefficients. Recently, in [5, Thm. 3.14] it has been shown that, if all the coefficients are given in unary, it can be

decided in TC^0 whether such an equation or a system of a fixed number of equations has a solution. Since from the proof of [5, Thm. 3.14] it is not obvious how to find an actual solution, we prove the following result in our full version on arXiv [21]:

▶ **Proposition 7.** *The following problem is in TC^0: Given integers $a_1, \dots, a_n$ in unary, compute $x_1, \dots, x_n \in \mathbb{Z}$ (either in unary or binary) such that $x_1a_1 + \cdots + x_na_n = \gcd(a_1, \dots, a_n)$ and $|x_i| \leq (n+1)\left(\max\{|a_1|, \dots, |a_n|\}\right)^2$.*

Matrix reduction. The matrix reduction procedure converts an arbitrary matrix of coordinates into its full form and, thus, is an essential step for solving the subgroup membership problem and several other problems. It was first described in [24] – however, without a precise complexity estimate. In this section, we repeat the presentation from [16] and show that for fixed c and r, it can be actually computed uniformly for groups in $\mathcal{N}_{c,r}$ in TC^0 – in the case that the inputs are given in unary (as words). If the inputs are represented as words with binary exponents, then we still can show that it is TC^0-Turing-reducible to ExtGCD. In Section 2, we defined the matrix representation of subgroups of nilpotent groups. We adopt all notation from Section 2.

As before, let $c, r \in \mathbb{N}$ be fixed and let $(a_1, \dots, a_m)$ be the standard Mal'cev basis of $F_{c,r}$. Let $G \in \mathcal{N}_{c,r}$ be given as quotient presentation, i. e., as a matrix in full form (either with unary or binary coefficients). We define the following operations on tuples $(h_1, \dots, h_n)$ (our subgroup generators) of elements of G and the corresponding operations on the associated matrix, with the goal of converting $(h_1, \dots, h_n)$ to a sequence in full form generating the same subgroup $H = \langle h_1, \dots, h_n \rangle$:

1. Swap h_i with h_j. This corresponds to swapping row i with row j.
2. Replace h_i by $h_i h_j^l$ ($i \neq j$, $l \in \mathbb{Z}$). This corresponds to replacing row i by $\mathrm{Coord}(h_i h_j^l)$.
3. Add or remove a trivial element from the tuple. This corresponds to adding or removing a row of zeros; or (3') a row of the form $(0 \ \dots \ 0 \ e_i \ \alpha_{i+1} \ \dots \ \alpha_m)$, where $i \in \mathcal{T}$ and $a_i^{-e_i} = a_{i+1}^{\alpha_{i+1}} \cdots a_m^{\alpha_m}$.
4. Replace h_i with h_i^{-1}. This corresponds to replacing row i by $\mathrm{Coord}(h_i^{-1})$.
5. Append an arbitrary product $h_{i_1}^{l_1} \cdots h_{i_k}^{l_k}$ with $i_1, \dots, i_k \in \{1, \dots, n\}$ and $l_1, \dots, l_k \in \mathbb{Z}$ to the tuple: add a new row with $\mathrm{Coord}(h_{i_1}^{l_1} \cdots h_{i_k}^{l_k})$.

Clearly, all these operations preserve H.

▶ **Lemma 8.** *On input of a quotient presentation of $G \in \mathcal{N}_{c,r}$ in unary (resp. binary) and a matrix of coordinates A given in unary (resp. binary), operations (1)–(5) can be done in TC^0. The output matrix will be also encoded in unary (resp. binary). For operations (2) and (5), we require that the exponents l, $l_1, \dots, l_k$ are given in unary (resp. binary).*

Moreover, as long as the rows in the matrix which are changed are pairwise distinct, a polynomial number of such steps can be done in parallel in TC^0.

Proof. Operations (1) and (3), clearly can be done in TC^0. Notice that operation (3') means simply that a row of the quotient presentation of G is appended to the matrix.

In the unary case, it follows directly from Theorem 5 that operations (2), (4), and (5) are in TC^0 because, since l, $l_1, \dots, l_k$ are given in unary, the respective group elements can be written down as words.

In the case of binary inputs, (5) works as follows ((2) and (4) analogously): by Lemma 2 (ii), there are functions $q_1, \dots, q_m \in \mathbb{Z}[x_1, \dots, x_m, z]$ such that for every $h \in F_{c,r}$ with $\mathrm{Coord}(h) = (\gamma_1, \dots, \gamma_m)$ anda $l \in \mathbb{Z}$, we have $\mathrm{Coord}_i(h^l) = q_i(\gamma_1, \dots, \gamma_m, l)$ in $F_{c,r}$. These functions can be used to compute $\mathrm{Coord}(h_{i_j}^{l_j})$ for $j = 1, \dots, k$. After that, $h_{i_1}^{l_1} \cdots h_{i_k}^{l_k}$ can be written down as word with binary exponents and Theorem 5 can be applied. ◀

Using the row operations defined above, in [16] it is shown how to reduce any coordinate matrix to its unique full form. Let us repeat these steps:

Let A_0 be a matrix of coordinates, as in (4) in Section 2. Recall that π_k denotes the column index of the k-th pivot (of the full form of A_0). We produce matrices $A_1, \ldots, A_s$, where s is the number of pivots in the full form of A_0, such that for every $k = 1, \ldots, s$ the first π_k columns of A_k form a matrix satisfying conditions 2-5 of being a full sequence, condition 6 is satisfied for all $i < \pi_{k+1}$, and A_s is the full form of A_0. Here we formally denote $\pi_{s+1} = m + 1$. Set $\pi_0 = 0$ and assume that A_{k-1} has been constructed for some $k \geq 1$. In the steps below we construct A_k. We let n and m denote the number of rows and columns, respectively, of A_{k-1}. At all times during the computation, h_i denotes the group element corresponding to row i of A_k and α_{ij} denotes the (i,j)-entry of A_k, which is $\mathrm{Coord}_j(h_i)$. These may change after every operation.

Step 1. Locate the column π_k of the next pivot, which is the minimum integer $\pi_{k-1} < \pi_k \leq m$ such that $\alpha_{i\pi_k} \neq 0$ for at least one $k \leq i \leq n$. If no such integer exists, then $k-1 = s$ and A_s is already constructed. Otherwise, set A_k to be a copy of A_{k-1} and denote $\pi = \pi_k$. Compute a linear expression of $d = \gcd(\alpha_{k\pi}, \ldots, \alpha_{n\pi}) = l_k\alpha_{k\pi} + \cdots + l_n\alpha_{n\pi}$. Let $h_{n+1} = h_k^{l_k} \cdots h_n^{l_n}$ and note that h_{n+1} has coordinates of the form $\mathrm{Coord}(h_{n+1}) = (0, \ldots, 0, d, \ldots)$ with d occurring in position π. Perform operation 5 to append h_{n+1} as row $n+1$ of A_k.

Step 2. For each $i = k, \ldots, n$, perform operation 2 to replace row i by $\mathrm{Coord}(h_i \cdot h_{n+1}^{-\alpha_{i\pi}/d})$. and for each $i = 1, \ldots, k-1$, use 2 to replace row i by $\mathrm{Coord}(h_i \cdot h_{n+1}^{-\lfloor \alpha_{i\pi}/d \rfloor})$. After that, swap row k with row $n+1$ using 1. At this point, properties 2-4 hold on the first k columns of A_k.

Step 3. If $\pi \in \mathcal{T}$, we additionally ensure condition 5 as follows. Perform row operation (3'), with respect to π, to append a trivial element h_{n+2} with $\mathrm{Coord}(h_{n+2}) = (0, \ldots, 0, e_\pi, \ldots)$ to A_k. Let $\delta = \gcd(d, e_\pi)$ and compute the linear expression $\delta = n_1 d + n_2 e_\pi$, with $|n_1|, |n_2| \leq \max\{d, e_\pi\}$. Let $h_{n+3} = h_k^{n_1} h_{n+2}^{n_2}$ and append this row to A_k, as row $n+3$. Note that $\mathrm{Coord}(h_{n+3}) = (0, \ldots, 0, \delta, \ldots)$, with δ in position π. Replace row k by $\mathrm{Coord}(h_k \cdot h_{n+3}^{-d/\delta})$ and row $n+2$ by $\mathrm{Coord}(h_{n+2} \cdot h_{n+3}^{-e_\pi/\delta})$, producing zeros in column π in these rows. Swap row k with row $n+3$. At this point, 2, 3, and 5 hold (for the first π_k columns) but 4 need not, since the pivot entry is now δ instead of d. For each $j = 1, \ldots, k-1$, replace row j by $\mathrm{Coord}(h_j \cdot h_k^{-\lfloor \alpha_{j\pi}/\delta \rfloor})$, ensuring 4.

Step 4. Identify the next pivot π_{k+1} (like in Step 1). If π_k is the last pivot, we set $\pi_{k+1} = m+1$. We now ensure condition 6 for $i < \pi_{k+1}$. Observe that Steps 1-3 preserve $\langle\, h_j \mid \pi_j \geq i \,\rangle$ for all $i < \pi_k$. Hence 6 holds in A_k for $i < \pi_k$ since it holds in A_{k-1} for the same range. Now consider i in the range $\pi_k \leq i < \pi_{k+1}$. It suffices to establish (vi.i) for all $j > k$ and (vi.ii) for π_k only. To obtain (vi.i), notice that $h_k^{-1} h_j h_k, h_k h_j h_k^{-1} \in \langle\, h_\ell \mid \ell > k \,\rangle$ if, and only if, $[h_j, h_k^{\pm 1}] \in \langle\, h_\ell \mid \ell > k \,\rangle$. Further, note that the subgroup generated by $S_j = \{1, h_j, [h_j, h_k], \ldots, [h_j, h_k, \ldots, h_k]\}$, where h_k appears $m - \pi_k$ times in the last commutator, is closed under commutation with h_k since if h_k appears more than $m - \pi_k$ times then the commutator is trivial. An inductive argument shows that the subgroup $\langle S_j \rangle$ coincides with $\langle h_k^{-\ell} h_j h_k^{\ell} \mid 0 \leq \ell \leq m - \pi_k \rangle$. Similar observations can be made for conjugation by h_k^{-1}. Therefore, appending via operation 5 rows $\mathrm{Coord}(h_k^{-\ell} h_j h_k^{\ell})$ for all $1 \leq |\ell| \leq m - \pi_k$ and all $k < j \leq n+3$ delivers (vi.i) for all $j > k$. Note that (vi.i) remains true for $i < \pi_k$.

To obtain (vi.ii), in the case $\pi_k \in \mathcal{T}$, we add row $\mathrm{Coord}(h_k^{e_k/\alpha_{k\pi_k}})$. Note that this element commutes with h_k and therefore (vi.i) is preserved.

Step 5. Using operation 3, eliminate all zero rows. The matrix A_k is now constructed.

We have to show that each step can be performed in TC^0 given that all Mal'cev coordinates

are encoded in unary (resp. in $\mathsf{TC}^0(\text{ExtGCD})$ if Mal'cev coordinates are encoded in binary). Since the total number of steps is constant (only depending on the nilpotency class and number of generators), this gives a TC^0 (resp. $\mathsf{TC}^0(\text{ExtGCD})$) circuit for computing the full form of a given subgroup.

Step 1. The next pivot can be found in TC^0 since it is simply the next column in the matrix with a non-zero entry, which can be found as a simple Boolean combination of test whether the entries are zero. In the unary case, by Proposition 7, $d = \gcd(\alpha_{k\pi}, \dots, \alpha_{n\pi})$ can computed in TC^0 together with $l_k, \dots, l_n$ encoded in unary such that $d = l_k\alpha_{k\pi} + \dots + l_n\alpha_{n\pi}$. Now, by Lemma 8, Step 1 can be done in TC^0.

In the binary case, d and $l_k, \dots, l_n$ can be computed using ExtGCD. Hence, by Lemma 8, Step 1 can be done in $\mathsf{TC}^0(\text{ExtGCD})$.

Step 2. The numbers $\lfloor \alpha_{i\pi}/d \rfloor$ (either in unary or binary) can be computed in TC^0 for all i in parallel by Theorem 1. After that one operation (2) is applied to each row of the matrix. By Lemma 8, this can be done in parallel for all rows in TC^0. Finally, swapping rows k and $n+1$ can be done in TC^0.

Step 3. As explained in Section 3, $\mathcal{T}$ and e_i for $i \in \mathcal{T}$ can be read directly from the quotient presentation. Thus, it can be decided in TC^0 whether Step 3 has to be executed. Appending a new row is in TC^0. Computing $\gcd(d, e_\pi) = d = n_1 d n_2 e_\pi$ is in TC^0 by Proposition 7 (in the unary case) and in $\mathsf{TC}^0(\text{ExtGCD})$ in the binary case. After that one operation (5) is followed by two operations (2), one operation (1), and, finally, $k-1$ times operation (2), which all can be done in TC^0 again by Lemma 8.

Step 4. The next pivot can be found in TC^0 as outlined in Step 1. After that, Step 4 consists of an application of a constant number (only depending on the nilpotency class and number of generators) of operations (5) and thus, by Lemma 8, is in TC^0.

Step 5. Clearly that is in TC^0.

Thus, we have completed the proof of our main result:

▶ **Theorem 9.** *Let $c, r \in \mathbb{N}$ be fixed. The following problem is in TC^0: given a unary encoded quotient presentation of $G \in \mathcal{N}_{c,r}$ and $h_1, \dots, h_n \in G$, compute the full form of the associated matrix of coordinates encoded in unary and hence the unique full-form sequence $(g_1, \dots, g_s)$ generating $\langle h_1, \dots, h_n \rangle$. Moreover, if the G and $h_1, \dots, h_n$ are given in binary, then the full-form sequence with binary coefficients can be computed in $\mathsf{TC}^0(\text{ExtGCD})$.*

Subgroup membership problem. As an easy application of the matrix reduction we can solve the subgroup membership problem in TC^0 – for a proof details see [21].

▶ **Corollary 10.** *Let $c, r \in \mathbb{N}$ be fixed. The following problem is in TC^0 (resp. $\mathsf{TC}^0(\text{ExtGCD})$ for binary inputs): given a quotient presentation of $G \in \mathcal{N}_{c,r}$, elements $h_1, \dots, h_n \in G$ and $h \in G$, decide whether or not h is an element of the subgroup $H = \langle h_1, \dots, h_n \rangle$.*

Moreover, if $h \in H$, the circuit computes the unique expression $h = g_1^{\gamma_1} \cdots g_s^{\gamma_s}$ where $(g_1, \dots, g_s)$ is the full-form sequence for H with the γ_i encoded in unary (resp. binary).

Alternatively, for unary inputs, the output can be given as word $h = h_{i_1}^{\epsilon_1} \cdots h_{i_t}^{\epsilon_t}$ where $i_j \in \{1, \dots, n\}$ and $\epsilon_j = \pm 1$.

Note that we do not know whether there is an analog of the second type of output for binary inputs. A possible way of expressing the output would be as a word with binary exponents over $h_1, \dots, h_n$. However, simply applying the same procedure as for unary inputs will not lead to a word with binary exponents.

Subgroup presentations. The full-form sequence associated to a subgroup H forms a Mal'cev basis for H. This allows us to compute a consistent nilpotent presentation for H. Note, however, that the resulting presentation is *not* a quotient presentation (although it can be transformed into one, see Proposition 14) – partly this is due to the fact that, in general, $H \notin \mathcal{N}_{c,r}$. The following is the TC^0 version of [16, Thm. 3.11]:

▶ **Corollary 11.** *Let $c, r \in \mathbb{N}$ be fixed. The following is in TC^0 for unary inputs and in $\mathsf{TC}^0(\text{ExtGCD})$ for binary inputs:*

Input: *a quotient presentation for $G \in \mathcal{N}_{c,r}$ and elements $h_1, \dots, h_n \in G$.*

Output: *a consistent nilpotent presentation for $H = \langle h_1, \dots, h_n \rangle$ given by a list of generators $(g_1, \dots, g_s)$ and numbers $\mu_{ij}, \alpha_{ijk}, \beta_{ijk} \in \mathbb{Z}$ encoded in unary (resp. binary) for $1 \le i < j < k \le s$ representing the relations (2)-(3).*

5 More algorithmic problems

The next two theorems are applications of Theorem 9. Their proofs (in [21]) follow essentially the proofs of their counterparts Theorems 4.1 and 4.6 in [16].

▶ **Theorem 12** (Kernels and preimages)**.** *Let $c, r \in \mathbb{N}$ be fixed. The following is in TC^0 for unary inputs and in $\mathsf{TC}^0(\text{ExtGCD})$ for binary inputs: On input of*

- *$G, H \in \mathcal{N}_{c,r}$ given as quotient presentations,*
- *a subgroup $K = \langle g_1, \dots, g_n \rangle \le G$,*
- *a list of elements $h_1, \dots, h_n$ defining a homomorphism $\varphi : K \to H$ via $\varphi(g_i) = h_i$, and*
- *optionally, an element $h \in H$ guaranteed to be in the image of φ,*

compute a generating set X for the kernel of φ, and an element $g \in G$ such that $\varphi(g) = h$.

In case of unary inputs, X and g will be returned as words, and for binary inputs, as words with binary exponents.

▶ **Theorem 13** (Conjugacy Problem)**.** *Let $c, r \in \mathbb{N}$ be fixed. The following is in TC^0 for unary inputs and in $\mathsf{TC}^0(\text{ExtGCD})$ for binary inputs: On input of some $G \in \mathcal{N}_{c,r}$ given as quotient presentation and elements $g, h \in G$, either produce some $u \in G$ such that $g = u^{-1}hu$, or determine that no such element u exists. In case of unary inputs, u will be returned as a word, for binary inputs, as a word with binary exponents.*

Computing quotient presentations. The results in the previous sections always required that the group is given as a quotient presentation. However, we can use Theorem 9 to transform an arbitrary presentation with at most r generators of a group in $\mathcal{N}_{c,r}$ into a quotient presentation. For a proof see [21].

▶ **Proposition 14.** *Let c and r be fixed integers. The following is in TC^0: given an arbitrary finite presentation with generators $a_1, \dots, a_r$ of a group $G \in \mathcal{N}_{c,r}$ (as a list of relators given as words over $\{a_1, \dots, a_r\}^{\pm 1}$), compute a quotient presentation of G (encoded in unary) and an explicit isomorphism. Moreover, if the relators are given as words with binary exponents, then the binary encoded quotient presentation can be computed in $\mathsf{TC}^0(\text{ExtGCD})$.*

▶ Remark. Because of Proposition 14, in all theorems above where the input is a quotient presentation, we can also take an arbitrary r-generated presentation of a group in $\mathcal{N}_{c,r}$ as input. However, be aware that for the word problem (Theorem 5 and Corollary 6) the complexity changes from TC^0 to $\mathsf{TC}^0(\text{ExtGCD})$ in the binary case.

Conclusion and Open Problems. We have seen that most problems which in [16] were shown to be in LOGSPACE indeed are in TC^0 even in the uniform setting where the number of generators and nilpotency class is fixed. Moreover, their binary versions are in $\mathsf{TC}^0(\text{ExtGCD})$ meaning that nilpotent groups are no more complicated than abelian groups in many algorithmic aspects. This contrasts with the slightly larger class of polycyclic groups: there the word problem is still in TC^0 [23, 12], but the conjugacy problem is not even known to be in NP. We conclude with some possible generalizations of our results:

- Does a uniform version of Theorem 5 hold (i. e., is the uniform word problem still in TC^0) for fixed nilpotency class but an arbitrary number of generators? What happens to the complexity if also the nilpotency class is part of the input? Note that in that case it is even not clear whether the word problem is still in polynomial time.
- Is there a way to solve the conjugacy problem for nilpotent groups with binary exponents in TC^0? Notice that we needed to compute gcds to solve the subgroup membership problem. However, the conjugacy problem might be solved using another method.
- What is the complexity of the uniform conjugacy problem with arbitrary nilpotency class?

References

1 Norman Blackburn. Conjugacy in nilpotent groups. *Proceedings of the American Mathematical Society*, 16(1):143–148, 1965.

2 William W. Boone. The Word Problem. *Ann. of Math.*, 70(2):207–265, 1959.

3 Max Dehn. Ueber unendliche diskontinuierliche Gruppen. *Math. Ann.*, 71:116–144, 1911.

4 Bettina Eick and Delaram Kahrobaei. Polycyclic groups: A new platform for cryptology? *ArXiv Mathematics e-prints*, 2004. `arXiv:math/0411077`.

5 Michael Elberfeld, Andreas Jakoby, and Till Tantau. Algorithmic meta theorems for circuit classes of constant and logarithmic depth. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:128, 2011. URL: `http://eccc.hpi-web.de/report/2011/128`.

6 Alberd Garreta, Alexei Miasnikov, and Denis Ovchinnikov. Properties of random nilpotent groups. *ArXiv e-prints*, December 2016. `arXiv:1612.01242`.

7 Philip Hall. *The Edmonton notes on nilpotent groups*. Queen Mary College Mathematics Notes. Mathematics Department, Queen Mary College, London, 1969.

8 William Hesse. Division is in uniform TC^0. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *ICALP*, volume 2076 of *Lecture Notes in Computer Science*, pages 104–114. Springer, 2001. `doi:10.1007/3-540-48224-5_9`.

9 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65:695–716, 2002.

10 Mikhail I. Kargapolov and Ju. I. Merzljakov. *Fundamentals of the theory of groups*, volume 62 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1979. Translated from the second Russian edition by Robert G. Burns.

11 Mikhail I. Kargapolov, Vladimir. N. Remeslennikov, N. S. Romanovskii, Vitaly A. Roman'kov, and V. A. Čurkin. Algorithmic questions for σ-powered groups. *Algebra i Logika*, 8:643–659, 1969.

12 Daniel König and Markus Lohrey. Evaluating matrix circuits. In *Computing and combinatorics*, volume 9198 of *Lecture Notes in Comput. Sci.*, pages 235–248. Springer, Cham, 2015. `doi:10.1007/978-3-319-21398-9_19`.

13 Klaus-Jörn Lange and Pierre McKenzie. On the complexity of free monoid morphisms. In Kyung-Yong Chwa and Oscar H. Ibarra, editors, *Algorithms and Computation, 9th International Symposium, ISAAC'98, Taejon, Korea, December 14-16, 1998, Proceedings*,

volume 1533 of *Lecture Notes in Computer Science*, pages 247–256. Springer, 1998. doi:10.1007/3-540-49381-6_27.

14 Charles. R. Leedham-Green and Leonard H. Soicher. Symbolic collection using Deep Thought. *LMS J. Comput. Math.*, 1:9–24 (electronic), 1998. doi:10.1112/S1461157000000127.

15 Richard J. Lipton and Yechezkel Zalcstein. Word problems solvable in logspace. *J. ACM*, 24(3):522–526, July 1977. doi:10.1145/322017.322031.

16 Jeremy MacDonald, Alexei G. Myasnikov, Andrey Nikolaev, and Svetla Vassileva. Logspace and compressed-word computations in nilpotent groups. *CoRR*, abs/1503.03888, 2015. URL: http://arxiv.org/abs/1503.03888.

17 Anatoly I. Mal'cev. On homomorphisms onto finite groups. *Transl., Ser. 2, Am. Math. Soc.*, 119:67–79, 1983. Translation from Uch. Zap. Ivanov. Gos. Pedagog Inst. 18, 49-60 (1958).

18 Andrzej Mostowski. Computational algorithms for deciding some problems for nilpotent groups. *Fundamenta Mathematicae*, 59(2):137–152, 1966. URL: http://eudml.org/doc/213887.

19 Alexei Myasnikov, Andrey Nikolaev, and Alexander Ushakov. The Post correspondence problem in groups. *J. Group Theory*, 17(6):991–1008, 2014. doi:10.1515/jgth-2014-0022.

20 Alexei Myasnikov, Andrey Nikolaev, and Alexander Ushakov. Non-commutative lattice problems. *J. Group Theory*, 19(3):455–475, 2016. doi:10.1515/jgth-2016-0506.

21 Alexei G. Myasnikov and Armin Weiß. TC^0 circuits for algorithmic problems in nilpotent groups. *CoRR*, abs/1702.06616, 2017. URL: http://arxiv.org/abs/1702.06616.

22 Pyotr S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov*, pages 1–143, 1955. In Russian.

23 David Robinson. *Parallel Algorithms for Group Word Problems*. PhD thesis, University of California, San Diego, 1993.

24 Charles C. Sims. *Computation with finitely presented groups*, volume 48 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1994. doi:10.1017/CBO9780511574702.

25 Heribert Vollmer. *Introduction to Circuit Complexity*. Springer, Berlin, 1999.