Noname manuscript No.
(will be inserted by the editor)

Evaluation of Circuits over Nilpotent and Polycyclic Groups

Daniel König · Markus Lohrey

the date of receipt and acceptance should be inserted later

Abstract We study the circuit evaluation problem (also known as the compressed word problem) for finitely generated linear groups. The best upper bound for this problem is coRP (the complements of problems in randomized polynomial time), which is shown by a reduction to polynomial identity testing for arithmetic circuits. Conversely, the compressed word problem for the linear group $\mathsf{SL}_3(\mathbb{Z})$ is equivalent to polynomial identity testing. In the paper, we show that the compressed word problem for every finitely generated nilpotent group is in $\mathsf{DET} \subseteq \mathsf{NC}^2$. Within the larger class of polycyclic groups we find examples where the compressed word problem is at least as hard as polynomial identity testing for skew arithmetic circuits. It is a major open problem, whether polynomial identity testing for skew arithmetic circuits can be solved in polynomial time

1 Introduction

1.1 Circuit evaluation problems

The study of circuit evaluation problems has a long tradition in theoretical computer science and is tightly connected to many aspects in computational complexity theory. One of the most important circuit evaluation problems is *polynomial identity testing* (PIT). Here, the input is an arithmetic circuit, whose internal gates are labelled with either addition or multiplication and its input gates are labelled with variables $(x_1, x_2, \ldots)$ or constants $(-1, 0, 1)$, and it is asked whether the output gate evaluates to the zero polynomial (in this paper, we always work in the polynomial ring over the coefficient ring $\mathbb{Z}$ or $\mathbb{Z}_p$ for a prime p). Based on the Schwartz-Zippel-DeMillo-Lipton Lemma, Ibarra and Moran [19] proved that polynomial identity testing over $\mathbb{Z}$ or $\mathbb{Z}_p$ belongs to the class coRP (the complements of problems in randomized polynomial time). Whether there is a deterministic polynomial time algorithm for

Universität Siegen, Germany
E-mail: {koenig,lohrey}@eti.uni-siegen.de

polynomial identity testing is an important problem. In [20] it is shown that if there exists a language in $\mathsf{DTIME}(2^{\mathcal{O}(n)})$ that has circuit complexity $2^{\Omega(n)}$, then $\mathsf{P} = \mathsf{BPP}$ (and hence $\mathsf{P} = \mathsf{RP} = \mathsf{coRP}$). There is also an implication that goes the other way round: Kabanets and Impagliazzo [21] have shown that if polynomial identity testing belongs to P, then (i) there is a language in $\mathsf{NEXPTIME}$ that does not have polynomial size circuits, or (ii) the permanent is not computable by polynomial size arithmetic circuits. Both conclusions represent major open problem in complexity theory. Hence, although it is quite plausible that polynomial identity testing belongs to P (by [20]), it will be probably very hard to prove (by [21]).

Circuit evaluation problems can be also studied for other structures than polynomial rings, in particular non-commutative structures. For finite monoids, the circuit evaluation problem has been studied in [11], where it was shown using Barrington's technique [9] that for every non-solvable finite monoid the circuit evaluation problem is P-complete, whereas for every solvable monoid, the circuit evaluation problem belongs to the parallel complexity class $\mathsf{DET} \subseteq \mathsf{NC}^2$. Related to this work is the paper [33], which studies the prediction problem for cellular automata where the dynamics is defined by multiplication in a finite monoid. This prediction problem can be seen as the circuit evaluation problem for particular trellis-like circuits. Starting with [27] the circuit evaluation problem has been also studied for infinite finitely generated (f.g) monoids, in particular infinite f.g. groups. In this context, the input gates of the circuit are labelled with generators of the monoid and the internal gates compute the product of the two input gates.

1.2 Compressed word problems

In [27] and subsequent work, the circuit evaluation problem for a monoid is also called the *compressed word problem*. Recall that the *word problem* for a f.g. monoid M asks whether two given words over the alphabet of monoid generators evaluate to the same element of M. In case M is a group, this problem is equivalent to the question whether a given word over the generators evaluates to the identity element of the group. Now, if we consider a multiplicative circuit over a f.g. monoid M, then one can evaluate the circuit also in the free monoid Γ^* where Γ is the set of monoid generators that appear at the input gates of the circuit. The result will be a word over Γ, whose length can be exponential in the number of circuit gates. Hence, the circuit can be seen as a compressed representation of the word it produces. Circuits over a free monoid are also known as *straight-line programs* (SLPs) and are intensively studied in the area of algorithms for compressed words, see [28] for an overview. Formally, the compressed word problem for the f.g. monoid M asks whether the words (over the generators of M) produced by two given SLPs evaluate the same monoid element of M (or, in the case of a group, whether the word produced by a single SLP evaluates to the group identity). This problem is equivalent to the question, whether two circuits over M evaluate to the same monoid element (or, in case of a group, whether a single circuit evaluates to the group identity).

We will restrict our attention to the compressed word problem for finitely generated groups in this paper. For this problem, polynomial time algorithms have been

developed for many important classes of groups, e.g., finite groups, f.g. nilpotent groups, f.g. free groups, graph groups (also known as right-angled Artin groups or partially commutative groups), and virtually special groups. The latter contain all Coxeter groups, one-relator groups with torsion, fully residually free groups, and fundamental groups of hyperbolic 3-manifolds; see [29] for details. For the important class of f.g. linear groups, i.e., f.g. groups of matrices over a field, it was shown in [29] that the compressed word problem can be reduced to polynomial identity testing (over $\mathbb{Z}$ or $\mathbb{Z}_p$, depending on the characteristic of the field) and hence belongs to coRP. Vice versa, in [29] it was shown that polynomial identity testing over $\mathbb{Z}$ can be reduced to the compressed word problem for the linear group $\mathsf{SL}_3(\mathbb{Z})$. The proof is based on a construction of Ben-Or and Cleve [12]. This result indicates that derandomizing the compressed word problem for a f.g. linear group will be in general very difficult.

1.3 Content of the paper

In this paper, we further investigate the tight correspondence between commutative circuits over rings and non-commutative circuits over linear groups. In Section 5.1 we study the complexity of the compressed word problem for f.g. nilpotent groups. For these groups, the compressed word problem can be solved in polynomial time [29]. Here, we show that for every f.g. nilpotent group the compressed word problem belongs to the parallel complexity class $\mathsf{DET} \subseteq \mathsf{NC}^2$ (Theorem 26), which is the class of all problems that are AC^0-reducible to the computation of the determinant of an integer matrix, see [14]. To the knowledge of the authors, f.g. nilpotent groups are the only examples of infinite groups for which the compressed word problem can be shown to belong to NC. In [25] we proved that the compressed word problem for wreath products $G \wr \mathbb{Z}^n$, where G is a direct product of copies of $\mathbb{Z}$ and $\mathbb{Z}_p$ for primes p, belongs to coRNC^2 (the complement of the randomized version of NC^2), but no deterministic NC-algorithm is known for these groups. On the other hand, even for free groups, the compressed word problem is P-complete [27].

The main step of our proof for f.g. nilpotent groups is to show that for a torsion-free f.g. nilpotent group G the compressed word problem belongs to the logspace counting class $\mathsf{C}_=\mathsf{L}$ (and is in fact $\mathsf{C}_=\mathsf{L}$-complete if G is nontrivial). To show this, we use the well-known fact that a f.g. torsion-free nilpotent group can be embedded into the group $\mathsf{UT}_d(\mathbb{Z})$ of d-dimensional unitriangular matrices over $\mathbb{Z}$ for some fixed d. Then, the compressed word problem for $\mathsf{UT}_d(\mathbb{Z})$ is reduced to the question whether two additive circuits over the natural numbers evaluate to the same number, which is $\mathsf{C}_=\mathsf{L}$-complete. Let us mention that there are several $\mathsf{C}_=\mathsf{L}$-complete problems related to linear algebra [2].

In Section 5.2, we study the compressed word problem for the matrix group $\mathsf{UT}_d(\mathbb{Z})$ for the case that the dimension d is not fixed, i.e., part of the input. In this case, the compressed word problem turns out to be complete for the counting class $\mathsf{C}_=\mathsf{LogCFL}$, which is the LogCFL-analogue of $\mathsf{C}_=\mathsf{L}$; see Theorem 27.

Finally, in Section 5.3 we move from nilpotent groups to polycyclic groups. These are solvable groups where every subgroup is finitely generated. By results of

Additional properties	Complexity	Reference
variable-free, constant multiplicative depth	$\mathsf{C}_{=}\mathsf{L}$-complete	Corollary 18
variable-free	coRP	[3]
skew	coRNC	[21]
no restriction	coRP	[19]

Table 1: Results about polynomial identity testing over $(\mathbb{Z}, +, \cdot)$

Class of finitely generated groups	Complexity	Reference
finite solvable	DET	[11]
$(\mathbb{Z}, +)$	$\mathsf{C}_{=}\mathsf{L}$-complete	[43]
nilpotent, torsion-free, non-trivial	$\mathsf{C}_{=}\mathsf{L}$-complete	Theorem 19
nilpotent	DET	Theorem 26
linear	coRP	[29]
a fixed concrete polycyclic group	hard for $\mathsf{PIT}(\mathbb{Z})$ for skew circuits	Theorem 29

Table 2: Results about the compressed word problem

Mal'cev [31], Auslander [8], and Swan [40] these are exactly the solvable subgroups of $\mathsf{GL}_d(\mathbb{Z})$ for some d. We prove that polynomial identity testing for skew arithmetic circuits reduces to the compressed word problem for a specific 2-generator polycyclic group of Hirsch length three; see Corollary 32. A skew arithmetic circuit is an arithmetic circuit (as defined below) such that for every multiplication gate, one of its input gates is an input gate of the circuit, i.e., a variable or a constant. These circuits exactly correspond to algebraic branching programs. Even for skew arithmetic circuits, no polynomial time identity test is currently known (although the problem belongs to coRNC^2), see for instance [7, p. 6].

We also prove a new result for the standard (non-compressed word problem): We show that for every f.g. solvable linear group (this includes in particular all polycyclic groups), the word problem can be solved in DLOGTIME-uniform TC^0 (see Theorem 7 in Section 3), which is a subclass of logarithmic space. For f.g. linear (not necessarily solvable) groups Lipton and Zalcstein [26] and Simon [39] have shown that the word problem can be solved in logarithmic space.

1.4 Overview of the results

Let us give an overview on the results of this paper and the related known results. Table 1 summarize the results about polynomial identity testing (PIT) over $(\mathbb{Z}, +, \cdot)$ for various restricted circuit classes. Table 2 summarizes results about the compressed word problem for various classes of groups.

2 Preliminaries and definitions

We assume that the reader has some basic knowledge in computational complexity theory, see e.g. [6] for a detailed introduction. Logspace always refers to deterministic logarithmic space. We use several times the well known fact that the composition of two (and hence and constant number of) logspace computable functions is logspace computable as well.

The complexity class RP is the set of all problems A for which there exists a polynomial time bounded randomized Turing machine R such that: (i) if $x \in A$ then R accepts x with probability at least $1/2$, and (ii) if $x \notin A$ then R accepts x with probability 0. The class coRP is the class of all complements of problems from RP.

We will also need some notions from circuit complexity, which will be explained in Section 2.2. But before, let us first introduce some background on arithmetic circuits.

2.1 Arithmetic circuits

Let us fix a set $X = \{x_1, x_2, \ldots\}$ of variables. Moreover, let $(R, +, \cdot)$ be one of the rings $(\mathbb{Z}, +, \cdot)$ or $(\mathbb{Z}_n, +, \cdot)$ for $n \geq 2$. Arithmetic circuits are usually defined as certain directed acyclic graphs. For our purpose, it is more convenient to use the equivalent formalism below, which is also known under the term "arithmetic straight-line program". An *arithmetic circuit* over the ring $(R, +, \cdot)$ is a triple $C = (V, S, \mathsf{rhs})$ with the following properties:

- V is a finite set of *gates*.
- $S \in V$ is the *output gate*.
- For every gate A, $\mathsf{rhs}(A)$ (the *right-hand side of* A) is either a variable from X, one of the constants -1, 0, 1, or an expression of the form $B + C$ or $B \cdot C$ where B and C are gates.
- There is a linear order $<$ on V such that $B < A$ whenever B occurs in $\mathsf{rhs}(A)$.

A gate A where $\mathsf{rhs}(A)$ has the form $B + C$ (resp., $B \cdot C$) is called an addition gate (resp., multiplication gate). A gate that is labelled with a variable or a constant is an *input gate*. Notice that we can restrict the input values to $\{-1, 0, 1\}$, since this set generates the ring $(R, +, \cdot)$. In fact, for the finite rings $(\mathbb{Z}_n, +, \cdot)$ we can restrict to the input values 0 and 1. Also note that for the ring $(\mathbb{Z}, +, \cdot)$ one can produce the integer $a \in \mathbb{Z}$ by a circuit of size $O(\log |a|)$.

Assume that $\mathcal{C} = (V, S, \mathsf{rhs})$ is an arithmetic circuit over $(R, +, \cdot)$ in which the variables $x_1, \ldots, x_n$ occur. Then we can evaluate every gate $A \in V$ to a polynomial $\mathsf{val}_{\mathcal{C}}(A) \in R[x_1, \ldots, x_n]$ in the obvious way (here,"val" stands for "value"). Moreover let $\mathsf{val}(\mathcal{C}) = \mathsf{val}_{\mathcal{C}}(S)$ be the polynomial to which $\mathcal{C}$ evaluates. Two arithmetic circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent if they evaluate to the same polynomial.

Fix an arithmetic circuit $\mathcal{C} = (V, S, \mathsf{rhs})$. We can view $\mathcal{C}$ as a directed acyclic graph (dag) where every node is labelled with a variable or a constant or an operator $+$, $\cdot$. If $\mathsf{rhs}(A) = B \circ C$ (for $\circ$ one of the operators), then there is an edge from B to A and C to A. The *depth* $\mathsf{depth}(A)$ (resp., *multiplication depth* $\mathsf{mdepth}(A)$)

of the gate A is the maximal number of gates (resp., multiplication gates) along a path from an input gate to A. So, input gates have depth one and multiplication depth zero. The *depth* (resp., *multiplication depth*) of $\mathcal{C}$ is $\mathsf{depth}(\mathcal{C}) = \mathsf{depth}(S)$ (resp., $\mathsf{mdepth}(\mathcal{C}) = \mathsf{mdepth}(S)$). The *formal degree* $\mathsf{deg}(A)$ of a gate A is 1 if A is an input gate, $\max\{\mathsf{deg}(B), \mathsf{deg}(C)\}$ if $\mathsf{rhs}(A) = B + C$, and $\mathsf{deg}(B) + \mathsf{deg}(C)$ if $\mathsf{rhs}(A) = B \cdot C$. The formal degree of $\mathcal{C}$ is $\mathsf{deg}(\mathcal{C}) = \mathsf{deg}(S)$. We deal with the following subclasses of arithmetic circuits (and the intersections of these classes):

- *positive circuits*: these are circuits over $(\mathbb{Z}, +, \cdot)$ without input gates labelled by the constant -1.
- *addition circuits*: these are circuits without multiplication gates.
- *variable-free circuits*: these are circuits without variables.
- *skew circuits*: these are arithmetic circuits such that for every multiplication gate A with $\mathsf{rhs}(A) = B \cdot C$, B is an input gate.

Variable-free circuits evaluate to elements of the underlying ring. Variable-free positive addition circuits (i.e., circuits that only contain input gates labelled with 0 and 1 and addition gates), are also known as addition chains [24].

In the rest of the paper we will allow for convenience more complicated expressions in right-hand sides for gates. For instance, we may have a gate with $\mathsf{rhs}(A) = (B + C) \cdot (D + E)$ or simply $\mathsf{rhs}(A) = B$, where B is again a gate (so-called copy gates). We also say that circuits where all right-hand sides fulfill the form of the definition are in *normal form*. We will make use of the following fact:

Lemma 1 *A given circuit over a ring $(R, +, \cdot)$ can be transformed in logarithmic space into an equivalent normal form circuit.*

Proof We proceed in two stages. In a first step, we eliminate copy gates, i.e., gates A with $\mathsf{rhs}(A) = B$ for another gate B. Consider the directed graph G that contains for every copy gate A the gate A as well as the gate $\mathsf{rhs}(A)$. Moreover, there is a directed edge from A to $B = \mathsf{rhs}(A)$. This is a directed forest where the edges are oriented towards the roots since every node has at most one outgoing edge (and the graph is acyclic). Clearly, G can be computed in logarithmic space. Using G, one can compute in logarithmic space for every copy gate A the unique gate B such that B is reachable from A in the graph G and B has outdegree zero in G (and hence is not a copy gate). For this, we follow the unique path in G that starts in the gate A and only store the current gate. We then set the right-hand side of A to $\mathsf{rhs}(B)$.

After the first step, the circuit does not contain copy gates. It remains to split up all remaining right-hand sides that violate the normal form by introducing fresh gates. For instance, if $\mathsf{rhs}(A) = (B + C) \cdot (D + E)$, one has to introduce fresh gates A' and A'' and set $\mathsf{rhs}(A) = A' \cdot A''$ and $\mathsf{rhs}(A') = B + C$ and $\mathsf{rhs}(A'') = D + E$. This can be done by a logspace machine that traverses the right-hand sides and thereby outputs the new gates and their right-hand sides. □

Polynomial identity testing for a ring R is the following computational problem: Given an arithmetic circuit $\mathcal{C}$ (with variables $x_1, \ldots, x_n$), does $\mathsf{val}(\mathcal{C}) = 0$ hold, i.e., does $\mathcal{C}$ evaluate to the zero-polynomial in $R[x_1, \ldots, x_n]$? It is an outstanding open problem in algebraic complexity theory, whether polynomial identity testing

for $(\mathbb{Z}, +, \cdot)$ can be solved in polynomial time. Ibarra and Moran [19] proved that polynomial identity testing for $(\mathbb{Z}, +, \cdot)$ or $(\mathbb{Z}_p, +, \cdot)$ (p a prime) belongs to the class coRP. For the rings $(\mathbb{Z}_n, +, \cdot)$ with p not a prime, membership in coRP was shown in [1].

2.2 Circuit complexity and logspace counting classes

The counting class #L consists of all functions $f : \Sigma^* \to \mathbb{N}$ for which there is a nondeterministic logarithmic space bounded Turing machine M such that for every $w \in \Sigma^*$, $f(w)$ is the number of accepting computation paths of M on input w. The class $\mathsf{C}_=\mathsf{L}$ contains all languages A for which there are two functions $f_1, f_2 \in \#\mathsf{L}$ such that for every $w \in \Sigma^*$, $w \in A$ if and only if $f_1(w) = f_2(w)$. The class $\mathsf{C}_=\mathsf{L}$ is closed under logspace many-one reductions. The canonical $\mathsf{C}_=\mathsf{L}$-complete problem is the following: The input consists of two dags G_1 and G_2 and vertices s_1, t_1 (in G_1) and s_2, t_2 (in G_2), and it is asked whether the number of different paths from s_1 to t_1 in G_1 is equal to the number of different paths from s_2 to t_2 in G_2. This problem is easily seen to be equivalent to the following problem: Given two variable-free positive addition circuits $\mathcal{C}_1$ and $\mathcal{C}_2$, does $\mathsf{val}(\mathcal{C}_1) = \mathsf{val}(\mathcal{C}_2)$ hold? Another important $\mathsf{C}_=\mathsf{L}$-complete problem is the question whether the determinant of a given integer matrix is zero [42,44].

We use standard definitions concerning circuit complexity, see e.g. [45]. We only consider polynomially bounded families $(C_n)_{n\geq 0}$ of Boolean circuits where the number of gates of C_n is bounded by a polynomial $p(n)$. For such a family, gates of C_n can be encoded with bit strings of length $O(\log n)$. We will consider the class TC^0 of all problems that can be recognized by a polynomial size circuit family of constant depth built up from NOT-gates (which have fan-in one) and AND-gates, OR-gates and MAJORITY-gates (i.e., gates that return 1 if and only if more than the half of its inputs are 1) of unbounded fan-in. If MAJORITY-gates are not allowed, we obtain the class AC^0. The class NC^k ($k \geq 1$) is defined by polynomial size circuit families of depth $O(\log^k n)$ that use NOT-gates, and AND- and OR-gates of fan-in two. One defines $\mathsf{NC} = \bigcup_{k\geq 1} \mathsf{NC}^k$. A family of AC^0- resp. TC^0-circuits $(C_n)_{n\geq 0}$ is DLOGTIME-uniform, if for given binary coded gates u, v of C_n, one can (i) compute the type of gate u in time $O(\log n)$ and (ii) check in time $O(\log n)$ whether u is an input gate for v. Note that the time bound $O(\log n)$ is linear in the input length $|u|+|v|$. To define DLOGTIME-uniformity for NC^1-circuits one needs the so-called extended connection language. We do not have to define this in detail (which can be found in [38,45]), since we will not work with uniformity explicitly. For $k \geq 2$, DLOGTIME-uniformity for NC^k is equivalent to logspace-uniformity. The latter means that the n-th circuit in the family can be computed in logarithmic space from the unary encoding of n. A language A is AC^0-reducible to languages $B_1, \ldots, B_k$ if A can be solved with a DLOGTIME-uniform polynomial size circuit family of constant depth that uses NOT-gates and unbounded fan-in AND-gates, OR-gates, and B_i-gates ($1 \leq i \leq k$). Here, a B_i-gate (it is also called an oracle gate) receives an ordered tuple of inputs $x_1, x_2, \ldots, x_n$ and outputs 1 if and only if $x_1 x_2 \cdots x_n \in B_i$. Sometimes, also the term "uniform constant depth reducibility" is used for this type of reductions. In the

same way, the weaker NC^1-reducibility can be defined. Here, one counts the depth of a B_i-gate with inputs $x_1, x_2, \ldots, x_n$ as $\log n$. The class DET contains all problems that are AC^0-reducible to the computation of the determinant of an integer matrix, see [14]. Actually, Cook originally defined DET as the class of all problems that are NC^1-reducible to the computation of the determinant of an integer matrix, but later [15] remarked that the above definition via AC^0-circuits seems to be more natural. For instance, it implies that DET is equal to the $\#\mathsf{L}$-hierarchy. It is known that $\mathsf{C}_=\mathsf{L} \subseteq \mathsf{DET} \subseteq \mathsf{NC}^2$, see e.g. [5, Section 4].

An *NAuxPDA* is a nondeterministic Turing machine with an additional pushdown store. The class $\mathsf{LogCFL} \subseteq \mathsf{NC}^2$ is the class of all languages that can be accepted by a polynomial time bounded NAuxPDA whose work tape is logarithmically bounded (but the pushdown store is unbounded). If we assign to the input the number of accepting computation paths of such an NAuxPDA, we obtain the counting class $\#\mathsf{LogCFL}$. In [44] it is shown that $\#\mathsf{LogCFL}$ is the class of all functions $f : \{0,1\}^* \to \mathbb{N}$ (a non-binary input alphabet Σ has to be encoded into $\{0,1\}^*$) for which there exists a logspace-uniform family $(\mathcal{C}_n)_{n\geq 1}$ of positive arithmetic circuits such that $\mathcal{C}_n$ computes the mapping f restricted to $\{0,1\}^n$ and there is a polynomial $p(n)$ such that the formal degree of $\mathcal{C}_n$ is bounded by $p(n)$. The class $\mathsf{C}_=\mathsf{LogCFL}$ contains all languages A for which there are two functions $f_1, f_2 \in \#\mathsf{LogCFL}$ such that for every $w \in \Sigma^*$, $w \in A$ if and only if $f_1(w) = f_2(w)$.

2.3 Matrices and groups

Let A be a square matrix of dimension d over some commutative ring R. With $A[i,j]$ we denote the entry of A in row i and column j. The matrix A is called *triangular* if $A[i,j] = 0$ whenever $i > j$, i.e., all entries below the main diagonal are 0. A *unitriangular matrix* is a triangular matrix A such that $A[i,i] = 1$ for all $1 \leq i \leq d$, i.e., all entries on the main diagonal are 1. We denote the set of unitriangular matrices of dimension d over the ring R by $\mathsf{UT}_d(R)$. It is well known that for every commutative ring R, the set $\mathsf{UT}_d(R)$ is a group (with respect to matrix multiplication).

Let $1 \leq i < j \leq d$. With $T_{i,j}$ we denote the matrix from $\mathsf{UT}_d(R)$ (so all diagonal elements are 1) such that $T_{i,j}[i,j] = 1$ and $T_{i,j}[k,l] = 0$ for all k, l with $1 \leq k < l \leq d$ and $(k,l) \neq (i,j)$. The notation $T_{i,j}$ does not specify the dimension d of the matrix, but the dimension will be always clear from the context. The group $\mathsf{UT}_d(\mathbb{Z})$ is generated by the finite set $\Gamma_d = \{T_{i,i+1} \mid 1 \leq i < d\}$, see e.g. [13].

As usual we denote with $[x,y] = x^{-1}y^{-1}xy$ the commutator of x and y. We will make use of the following lemma, which shows how to encode multiplication with unitriangular matrices, see [30] for a proof (note that $T_{i,j}^a$ denotes the a-th power of the matrix $T_{i,j}$ and that this is also defined for $a < 0$ since $T_{i,j}$ is invertible):

Lemma 2 ([30, Lemma 3.1]) *For all $a, b \in \mathbb{Z}$ and $1 \leq i < j < k \leq d$ we have $[T_{i,j}^a, T_{j,k}^b] = T_{i,k}^{ab}$.*

In this paper we are concerned with certain subclasses of *linear groups*. A group is linear if it is isomorphic to a subgroup of $\mathsf{GL}_d(F)$ (the group of all invertible $(d \times d)$-matrices over the field F) for some field F.

A (n-step) solvable group G is a group G, which has a a subnormal series $G = G_n \rhd G_{n-1} \rhd G_{n-2} \rhd \cdots \rhd G_1 \rhd G_0 = 1$ (i.e., G_i is a normal subgroup of G_{i+1} for all $0 \leq i \leq n-1$) such that every quotient G_{i+1}/G_i is abelian ($0 \leq i \leq n-1$). If every quotient G_{i+1}/G_i is cyclic, then G is called *polycyclic*. The number of $0 \leq i \leq n-1$ such that $G_{i+1}/G_i \cong \mathbb{Z}$ is called the *Hirsch length* of G; it does not depend on the chosen subnormal series. If $G_{i+1}/G_i \cong \mathbb{Z}$ for all $0 \leq i \leq n-1$ then G is called *strongly polycyclic*. A group is polycyclic if and only if it is solvable and every subgroup is finitely generated. Polycyclic groups are linear. More precisely, Auslander and Swan [8,40] proved that the polycyclic groups are exactly the solvable groups of integer matrices.

For a group G its *lower central series* is the series $G = G_1 \rhd G_2 \rhd G_3 \rhd \cdots$ of subgroups where $G_{i+1} = [G_i, G]$, which is the subgroup generated by all commutators $[g, h]$ with $g \in G_i$ and $h \in G$. Indeed, G_{i+1} is a normal subgroup of G_i. The group G is *nilpotent*, if its lower central series terminates after finitely many steps in the trivial group 1. Every f.g. nilpotent group is polycyclic. We need the following results about nilpotent and solvable groups:

Theorem 3 ([37, Chapter 5]) *Subgroups and quotients of solvable (resp., nilpotent) groups are solvable (resp., nilpotent) as well.*

Theorem 4 ([22, Theorem 17.2.2]) *Every f.g. nilpotent group G has a torsion-free normal subgroup H of finite index (i.e., there is no $a \in H$ such that a^n is the group identity for some $n \in \mathbb{N}$).*

Theorem 5 ([22, Theorem 17.2.5]) *For every torsion-free f.g nilpotent group G there exists $d \geq 1$ such that G can be embedded into $\mathsf{UT}_d(\mathbb{Z})$.*

A group G is called metabelian if the commutator subgroup $[G, G]$ is abelian. In other words, the metabelian groups are the 2-step solvable groups. Even if G is f.g. metabelian, this does not imply that G is polycyclic, since $[G, G]$ is not necessarily finitely generated.

Let G be a f.g. group and let G be finitely generated as a group by Σ. Then, as a monoid G is finitely generated by $\Sigma \cup \Sigma^{-1}$ (where $\Sigma^{-1} = \{a^{-1} \mid a \in \Sigma\}$ is a disjoint copy of Σ and a^{-1} stands for the inverse of the generator $a \in \Sigma$). Recall that the *word problem* for G is the following computational problem: Given a word $w \in (\Sigma \cup \Sigma^{-1})^*$, does w evaluate to the identity of G? Note that the group G is fixed here. Thus, all parameters that only depend on the group G (but not on the input word w) can be treated as fixed constants in an algorithm for the word problem for G.

Kharlampovich proved in [23] that there exist finitely presented 3-step solvable groups with an undecidable word problem. On the other hand, for every f.g. linear groups Lipton and Zalcstein [26] and Simon [39] proved the following important result:

Theorem 6 ([26,39]) *For every f.g. linear group the word problem can be solved in deterministic logarithmic space.*

Lipton and Zalcstein [26] proved this result for a linear group over a field of characteristic zero, whereas Simon [39] considered fields of prime characterisitic. Theorem 6 implies that the word problem for every polycyclic group can be solved in

logspace. Robinson proved in his thesis that the word problem for a polycyclic group belongs to TC^0 [36], but his circuits are not uniform. For f.g. nilpotent groups, Robinsion [36] proved that the word problem belongs to $\mathsf{DLOGTIME}$-uniform TC^0. Waack considered in [46] arbitrary f.g. solvable linear groups (which include the polycyclic groups) and proved that their word problems belong to logspace-uniform NC^1. In the next section, we combine Waack's technique with the famous division breakthrough results by Hesse, Allender, and Barrington [18] to show that for every f.g. solvable linear group the word problem belongs to $\mathsf{DLOGTIME}$-uniform TC^0. This answers a question from [36] positively. For the Baumslag-Solitar group $\mathsf{BS}_{1,2}$, which is solvable and linear, Diekert, Myasnikov and Weiß proved that also the conjugacy problem can be solved in $\mathsf{DLOGTIME}$-uniform TC^0 [16], see also [48] for related results.

3 The complexity of the classical word problem for finitely generated linear groups

Before we study the compressed word problem for f.g. linear groups, we first prove the aforementioned result on the ordinary (uncompressed) word problem for f.g. solvable linear groups:

Theorem 7 *Let G be a f.g. linear group.*

- *If G is infinite solvable, then the word problem for G is complete for* $\mathsf{DLOGTIME}$*-uniform* TC^0.
- *If G is virtually solvable (i.e. G has a solvable subgroup of finite index), then the word problem for G belongs to* $\mathsf{DLOGTIME}$*-uniform* NC^1.

For the proof, we first have to consider the complexity of iterated multiplication and division with remainder of polynomials in several variables. Recall that $\mathbb{Z}[x_1, \ldots, x_k]$ denotes the ring of polynomials in the variables $x_1, \ldots, x_k$ with coefficients from $\mathbb{Z}$. For a polynomial $p \in \mathbb{Z}[x_1, \ldots, x_k]$ and a variable x_i we denote with $\deg_{x_i}(p)$ the maximal value d such that x_i^d appears in a monomial of p. We specify polynomials from $\mathbb{Z}[x_1, \ldots, x_k]$ by writing down for every non-zero term $ax_1^{n_1} \cdots x_k^{n_k}$ the tuple of integers $(a, n_1, \ldots, n_k)$, where a is represented in binary notation and the exponents are represented in unary notation. Iterated multiplication of polynomials in the ring $\mathbb{Z}[x_1, \ldots, x_k]$ is the task of computing from a given list of polynomials $p_1, p_2, \ldots, p_n \in \mathbb{Z}[x_1, \ldots, x_k]$ the product polynomial $p_1 p_2 \cdots p_n$. Division with remainder in the ring $\mathbb{Z}[x]$ (later, we will generalize this to several variables) is the task of computing for given polynomials $s, t \in \mathbb{Z}[x]$ such that $t \neq 0$ and the leading coefficient of t is 1 the unique polynomials $s \bmod t$ and $s \operatorname{div} t$ such that $s = (s \operatorname{div} t) \cdot t + s \bmod t$ and $\mathsf{deg}(s \bmod t) < \mathsf{deg}(t)$ where $\mathsf{deg}(p)$ denotes the degree of the polynomial t.

The following result was shown in [17, 18]:[1]

[1] Explicitly, the result is stated in [18, Corollary 6.5], where the authors note that Eberly's reduction [17] from iterated polynomial multiplication to iterated integer multiplication is actually an AC^0-reduction, which yields a $\mathsf{DLOGTIME}$-uniform TC^0 bound with the main result from [18].

Theorem 8 ([17,18]) *Iterated multiplication and division with remainder of polynomials in the ring $\mathbb{Z}[x]$ (respectively, $\mathbb{F}_p[x]$) belong to* DLOGTIME*-uniform* TC^0.

We need generalizations of Theorem 8 to multivariate polynomials. In the following proofs we always use the fact that iterated addition, iterated multiplication and division with remainder of binary coded integers can be done in DLOGTIME-uniform TC^0 [18].

Lemma 9 *Let k be a fixed constant. Iterated multiplication of polynomials in the ring $\mathbb{Z}[x_1, \ldots, x_k]$ (respectively, $\mathbb{F}_p[x_1, \ldots, x_k]$) belongs to* DLOGTIME*-uniform* TC^0.

Proof We only prove the result for $\mathbb{Z}[x_1, \ldots, x_k]$; exactly the same proof also works for $\mathbb{F}_p[x_1, \ldots, x_k]$.

For $d \geq 1$ let $\mathbb{Z}[x_1, \ldots, x_k]_d \subseteq \mathbb{Z}[x_1, \ldots, x_k]$ be the set of all polynomials $p \in \mathbb{Z}[x_1, \ldots, x_k]$ such that $\deg_{x_i}(p) \leq d$ for all $1 \leq i \leq k$. For $d \geq 2$ we define the mapping $\mathcal{U}_d : \mathbb{Z}[x_1, \ldots, x_k] \to \mathbb{Z}[z]$ by

$$\mathcal{U}_d(p(x_1, x_2, \ldots, x_k)) = p(z^{d^0}, z^{d^1}, \ldots, z^{d^k}).$$

The mapping $\mathcal{U}_d$ is also used in [1] to reduce polynomial identity testing to univariate polynomial identity testing. The mapping $\mathcal{U}_{d+1}$ restricted to $\mathbb{Z}[x_1, \ldots, x_k]_d$ is injective, since for a polynomial $p \in \mathbb{Z}[x_1, \ldots, x_k]_d$ we obtain the polynomial $\mathcal{U}_{d+1}(p)$ by replacing for every monomial $a \cdot x_1^{n_1} \cdots x_k^{n_k}$ by the monomial $a \cdot z^m$ where m is the number with base-$(d+1)$ expansion $(n_1 \cdots n_k)$ (with the most significant digit on the right). Moreover, for all polynomials $p, q \in \mathbb{Z}[x_1, \ldots, x_k]$ and all $d \geq 2$ we have

$$\mathcal{U}_d(p+q) = \mathcal{U}_d(p) + \mathcal{U}_d(q) \text{ and } \mathcal{U}_d(pq) = \mathcal{U}_d(p)\mathcal{U}_d(q). \tag{1}$$

We can calculate $\mathcal{U}_d(p)$ for a given polynomial $p \in \mathbb{Z}[x_1, \ldots, x_k]$ and a unary encoded number $d \geq 2$ in DLOGTIME-uniform TC^0: For a monomial $a x_1^{n_1} \cdots x_k^{n_k}$ (which is represented by the tuple $(a, n_1, \ldots, n_k)$) we have to compute the pair $(a, \sum_{i=0}^{k-1} n_{i+1} d^i)$, which is possible in DLOGTIME-uniform TC^0. Similarly, we can compute $\mathcal{U}_{d+1}^{-1}(p)$ for a polynomial $p \in \mathcal{U}_{d+1}(\mathbb{Z}[x_1, \ldots, x_k]_d)$ in DLOGTIME-uniform TC^0: From a given monomial az^m (represented by the pair (a, m)) we have to compute the tuple $(a, n_1, \ldots, n_k)$ where $n_i = (m \text{ div } (d+1)^{i-1}) \bmod (d+1)$, which can be done in DLOGTIME-uniform TC^0.

We now multiply given polynomials $p_1, \ldots, p_n \in \mathbb{Z}[x_1, \ldots, x_k]$ in the following way, where all steps can be carried out in DLOGTIME-uniform TC^0 by the above remarks.

- Compute the number $d = \max\{\sum_{i=1}^{n} \deg_{x_j}(p_i) \mid 1 \leq j \leq k\}$. This number bounds the degree of the product polynomial $p_1 p_2 \cdots p_n$ in any of the variables $x_1, \ldots, x_n$, i.e., $p_1 p_2 \cdots p_n \in \mathbb{Z}[x_1, \ldots, x_k]_d$.
- Compute in parallel $s_i(z) = \mathcal{U}_{d+1}(p_i)$ for $1 \leq i \leq n$.
- Using Theorem 8, compute the product $S(z) = s_1(z)s_2(z)\cdots s_n(z)$, which is $\mathcal{U}_{d+1}(p_1 p_2 \cdots p_n)$ by (1).
- Finally, compute $\mathcal{U}_{d+1}^{-1}(S)$, which is $p_1 p_2 \cdots p_n$. □

For polynomial division in several variables, we need a distinguished variable. Therefore, we consider the polynomial ring $\mathbb{Z}[x_1, \ldots, x_k, y]$. We view polynomials from this ring as polynomials in the variable y where coefficients are polynomials from $\mathbb{Z}[x_1, \ldots, x_k]$. We will only divide by a polynomial t for which the leading monomial $p(x_1, \ldots, x_n)y^m$ of t satisfies $p(x_1, \ldots, x_n) = 1$. This ensures that the coefficients of the quotient and remainder polynomial are again in $\mathbb{Z}[x_1, \ldots, x_k]$ (and not in the quotient field $\mathbb{Q}(x_1, \ldots, x_n)$).

Lemma 10 *Let k be a fixed constant. Division with remainder of polynomials in the ring $\mathbb{Z}[x_1, \ldots, x_k, y]$ (respectively, $\mathbb{F}_p[x_1, \ldots, x_k, y]$) belongs to* DLOGTIME*-uniform* TC^0.

Proof Again, we only prove the result for $\mathbb{Z}[x_1, \ldots, x_k, y]$; exactly the same proof works for $\mathbb{F}_p[x_1, \ldots, x_k, y]$ as well. As in the proof of Lemma 9 consider the set $\mathbb{Z}[x_1, \ldots, x_k, y]_d \subseteq \mathbb{Z}[x_1, \ldots, x_k, y]$ of all polynomials in $\mathbb{Z}[x_1, \ldots, x_k, y]$ such that for every monomial $a \cdot x_1^{n_1} \cdots x_k^{n_k} y^n$ we have $n_1, \ldots, n_k, n < d$, and the mapping $\mathcal{U}_d : \mathbb{Z}[x_1, \ldots, x_k, y] \to \mathbb{Z}[z]$ with

$$\mathcal{U}_d(p(x_1, x_2, \ldots, x_k, y)) = p(z^{d^0}, z^{d^1}, \ldots, z^{d^{k-1}}, z^{d^k}).$$

Note that for polynomials $p, q \in \mathbb{Z}[x_1, \ldots, x_k, y]_d$ with $\deg_y(p) < \deg_y(q)$ we have $\mathsf{deg}(\mathcal{U}_{d+1}(p)) < \mathsf{deg}(\mathcal{U}_{d+1}(q))$, since the exponent of y becomes the most significant digit in the base-$(d+1)$ representation. Then, for all polynomials $s, t \in \mathbb{Z}[x_1, \ldots, x_k, y]_d$ (where the leading coefficient of t is 1) we have

$$\mathcal{U}_{d^2+1}(s \bmod t) = \mathcal{U}_{d^2+1}(s) \bmod \mathcal{U}_{d^2+1}(t).$$

To see this, assume that $s = qt + r$ with $\deg_y(r) < \deg_y(t)$, so that $r = s \bmod t$. We have $q, r \in \mathbb{Z}[x_1, \ldots, x_k, y]_{d^2}$, which can be checked by tracing the polynomial division algorithm. By (1) we have

$$\mathcal{U}_{d^2+1}(s) = \mathcal{U}_{d^2+1}(q)\mathcal{U}_{d^2+1}(t) + \mathcal{U}_{d^2+1}(r).$$

Moreover, $\mathsf{deg}(\mathcal{U}_{d^2+1}(r)) < \mathsf{deg}(\mathcal{U}_{d^2+1}(t))$. Hence

$$\mathcal{U}_{d^2+1}(r) = \mathcal{U}_{d^2+1}(s) \bmod \mathcal{U}_{d^2+1}(t).$$

Now we can compute the remainder $s \bmod t$ for polynomials $s, t \in \mathbb{Z}[x_1, \ldots, x_k, y]$ (where the leading coefficient of t is 1) in DLOGTIME-uniform TC^0 as follows:

- Compute the number $d = \max\{\deg_z(p) \mid p \in \{s, t\}, z \in \{x_1, \ldots, x_k, y\}\}$, so that $s, t \in \mathbb{Z}[x_1, \ldots, x_k, y]_d$.
- Compute in parallel $u(z) = \mathcal{U}_{d^2+1}(s)$ and $v(z) = \mathcal{U}_{d^2+1}(t)$.
- Compute, using Theorem 8, $R(z) = u(z) \bmod v(z)$, which is $\mathcal{U}_{d^2+1}(s \bmod t)$.
- Finally, compute $\mathcal{U}^{-1}_{d^2+1}(R)$ which is $s \bmod t$. □

In the same way we can also compute the quotient, but we only will need the remainder $s \bmod t$ in the following.

Finally, we will need the following result from [36]:

Theorem 11 ([36, Theorem 5.2]) *Let G be a f.g. group with a normal subgroup H of finite index. Then, the word problem for G is AC^0-reducible to the word problems for H and G/H.*

Now we are ready to prove Theorem 7.

Proof of Theorem 7. Let us first assume that G is f.g. solvable and linear over a field F. By a theorem of Mal'cev (see e.g. [47, Theorem 3.6]), G contains a normal subgroup H of finite index, which is triangularizable over a finite extension of F. Using Theorem 11 we know that the word problem for G is AC^0-reducible to the word problems for H and G/H. The latter is a finite solvable group, see Theorem 3. Hence, its word problem belongs to $\mathsf{DLOGTIME}$-uniform TC^0 (actually ACC^0) by [10].

By the previous discussion, it suffices to show that the word problem for a f.g. triangular matrix group H over some field F belongs to $\mathsf{DLOGTIME}$-uniform TC^0. Let d be the dimension of the matrices in H (which is a fixed constant that only depends on the group G) and let P be the prime field of F. We can replace F by the finitely generated extension of P that is generated by all finitely many matrix entries in the generators of H. It is known that the field extension $[F : P]$ has a separating transcendence base $\{x_1, \ldots, x_k\}$, which means that $[F : P(x_1, \ldots, x_k)]$ is a finite separable extension; see e.g. [49, Theorem 31].[2] Hence, the theorem of the primitive element applies, which says that F is generated over $P(x_1, \ldots, x_k)$ by a single element $\alpha \in F$, which is algebraic over $P(x_1, \ldots, x_k)$. The number k is a fixed constant, that only depends on the group G.

Assume that $P = \mathbb{Q}$ (in case $P = \mathbb{F}_p$ for a prime p we have to replace in all arguments below $\mathbb{Z}$ by $\mathbb{F}_p$). Consider the minimal polynomial $p(y) \in \mathbb{Q}(x_1, \ldots, x_k)[y]$ of α. We can write it as

$$p(y) = y^m + \frac{p_1}{q} y^{m-1} + \frac{p_2}{q} y^{m-2} \cdots + \frac{p_m}{q} \tag{2}$$

for polynomials $p_1, \ldots, p_m, q \in \mathbb{Z}[x_1, \ldots, x_k]$ with $q \neq 0$. Since $q \in \mathbb{Z}[x_1, \ldots, x_k] \setminus \{0\} \subseteq \mathbb{Q}(x_1, \ldots, x_k) \setminus \{0\}$ is a non-zero element of the base field $\mathbb{Q}(x_1, \ldots, x_k)$, the element $\beta = q \cdot \alpha \in F$ also generates F over $\mathbb{Q}(x_1, \ldots, x_k)$, and its minimal polynomial is

$$r(y) = y^m + p_1 \cdot y^{m-1} + p_2 q \cdot y^{m-2} + \cdots + p_m q^{m-1} \in \mathbb{Z}[x_1, \ldots, x_k, y]$$

(multiply (2) by q^m). We have

$$F = \mathbb{Q}(x_1, \ldots, x_k)[y] / \langle r(y) \rangle$$

where $\langle r(y) \rangle = \{a(x) \cdot r \mid a(x) \in \mathbb{Q}(x_1, \ldots, x_k)[y]\}$ is the ideal generated by r.

Each of the finitely many generators of the group H is a $(d \times d)$-matrix, whose entries are polynomials in the variable y with coefficients from the fraction field $\mathbb{Q}(x_1, \ldots, x_k)$. Every such coefficient is a fraction $a(x_1, \ldots, x_k)/b(x_1, \ldots, x_k)$ with

[2] Every finitely generated extension field of a perfect field has a separating transcendence base and every prime field is perfect.

$a(x_1,\ldots,x_k)$, $b(x_1,\ldots,x_k) \in \mathbb{Z}[x_1,\ldots,x_k]$. Let $g(x_1,\ldots,x_k)$ be the least common multiple of all denominators $b(x_1,\ldots,x_k)$, which is a fixed polynomial that only depends on the group G. Instead of asking whether $A_1 \cdots A_n \equiv \mathsf{Id} \bmod q(y)$ (for group generators $A_1,\ldots,A_n$ of H) we can ask whether

$$gA_1 \cdots gA_n \equiv g^n \mathsf{Id} \bmod q(y).$$

Here, for two $(d\times d)$-matrices A and B, $A \equiv B \bmod q(x)$ means that $A[i,j] \equiv B[i,j] \bmod q(x)$ for all $1 \leq i,j \leq d$. So far, the proof has been following more or less closely Waack's arguments from [46].

Let $M_i = gA_i$, which is a triangular matrix of dimension d (a fixed constant) with entries from $\mathbb{Z}[x_1,\ldots,x_k,y]$. Let us write $M_i = D_i + U_i$ where D_i is a diagonal matrix and U_i is upper triangular with all diagonal entries equal to zero. We get

$$M_1 \cdots M_n = \prod_{i=1}^{n}(D_i + U_i) = \sum_{R_1 \in \{D_1,U_1\}} \cdots \sum_{R_n \in \{D_n,U_n\}} \prod_{j=1}^{n} R_j. \tag{3}$$

If there are more than $d-1$ factors U_i in a product $\prod_{j=1}^{n} R_j$, then the product is the zero matrix. So there are at most $\sum_{i=0}^{d-1}\binom{n}{i} \leq d\binom{n}{d} \leq dn^d$ summands (for $n > 2d$) in the sum (3) that are not equal to zero. When we look at one of the products $\prod_{j=1}^{n} R_j$ with at most $d-1$ many factors U_i, we can write it as

$$\begin{aligned}&\left(\prod_{i=1}^{m_1-1} D_i\right) U_{m_1} \left(\prod_{i=m_1+1}^{m_2-1} D_i\right) \cdots U_{m_l} \left(\prod_{i=m_l+1}^{n} D_i\right)\\ &= D_{1,m_1-1} U_{m_1} D_{m_1+1,m_2-1} \cdots U_{m_l} D_{m_l+1,n}\end{aligned}$$

for some $0 \leq l \leq d-1$ and $1 \leq m_1 < \cdots < m_l \leq n$ where $D_{s,t} = \prod_{i=s}^{t} D_i$ $(1 \leq s \leq t+1,\ 0 \leq t \leq n)$ is a product of at most n diagonal matrices. Each of these products can be calculated by calculating d products of at most n polynomials from $\mathbb{Z}[x_1,\ldots,x_k,y]$, which can be done in DLOGTIME-uniform TC^0 by Lemma 9 (recall that k is a constant). Moreover, all products $D_{s,t}$ for $1 \leq s < t \leq n$ can be computed in parallel. Once these products are computed, we can, in parallel, compute for all $0 \leq l \leq d-1$ and $1 \leq m_1 < \cdots < m_l \leq n$ the matrix product $D_{1,m_1-1} U_{m_1} D_{m_1+1,m_2-1} \cdots U_{m_l} D_{m_l+1,n}$. Note that this is a product of $2l+1 \leq 2d-1$ (and hence constantly many) matrices that have been computed before. Hence, the computation of $D_{1,m_1-1} U_{m_1} D_{m_1+1,m_2-1} \cdots U_{m_l} D_{m_l+1,n}$ involves a constant number of polynomial multiplications and additions. So, all the above matrix products can be computed in DLOGTIME-uniform TC^0 as well. Next, we have to compute the sum of all polynomially many matrices computed in the previous step. For this we have to compute d^2 many sums of polynomially many polynomials, which is again possible in DLOGTIME-uniform TC^0. The resulting matrix is $M_1 \cdots M_n = g^n A_1 \cdots A_n$. Finally we have to reduce all entries of the matrices $M_1 \cdots M_n$ and $g^n \mathsf{Id}$ modulo the minimal polynomial $q(y)$ which can also be done in DLOGTIME-uniform TC^0 by Lemma 10. Note that we divide by the polynomial $q(y)$, whose leading coefficient is indeed 1.

We have shown that the word problem for a f.g. solvable linear group G belongs to DLOGTIME-uniform TC^0. If G is in addition infinite, then G cannot be a torsion-group, since every f.g. linear torsion group is finite by a result of Schur, see [47, Corollary 4.9]. Therefore $\mathbb{Z}$ is a subgroup of G. Since the word problem for $\mathbb{Z}$ is already complete for DLOGTIME-uniform TC^0 (it corresponds to the problem of counting the number of ones in a word) we obtain the lower bound in the theorem.

Finally, let G be a f.g. virtually solvable linear group G. Then G contains a normal solvable subgroup H, for which we know that the word problem can be solved in DLOGTIME-uniform TC^0. Moreover, the quotient G/H is a finite group, for which the word problem belongs to DLOGTIME-uniform NC^1. Hence, Theorem 11 implies that the word problem for G belongs to DLOGTIME-uniform NC^1. $\square$

By Tits alternative [41], every linear group is either virtually solvable (i.e., has a solvable subgroup of finite index, which can be assumed to be normal) or contains a free group of rank two. Since by [36, Theorem 6.3], the word problem for a free group of rank two is hard for DLOGTIME-uniform NC^1, one gets:

Theorem 12 *For every f.g. linear group that is not virtually solvable, the word problem is hard for* DLOGTIME*-uniform* NC^1.

Theorem 7 and 12 leave open the case of a f.g. linear group G that is not solvable but a finite extension of a solvable group H. If the quotient G/H is solvable too, then G is solvable and we can apply Theorem 7. So, we can assume that the finite quotient G/H is not solvable. It seems plausible that in this case, the word problem for G is hard for DLOGTIME-uniform NC^1, since the word problem for every finite non-solvable group is hard for DLOGTIME-uniform NC^1 [9]. But it is not clear, whether the word problem for the finite quotient G/H reduces to the word problem for G (it reduces to the membership problem for H in G).

4 Straight-line programs and the compressed word problem

Let us now consider the compressed word problem. We start with the definition of a straight-line program (which should not be confused with arithmetic straight-line programs). A *straight-line program* (briefly, SLP) is basically a multiplicative circuit over a free monoid. We define an SLP over the finite alphabet Σ as a triple $\mathcal{G} = (V, S, \mathsf{rhs})$ where V is a finite set of variables (or gates), $S \in V$ is the start variable (or output gate), and rhs maps every variable to a right-hand side $\mathsf{rhs}(A)$, which is either a symbol $a \in \Sigma$, or of the form BC where $B, C \in V$. As for arithmetic circuits we require that there is a linear order $<$ on V such that $B < A$, whenever B occurs in $\mathsf{rhs}(A)$. The terminology "(start) variable" instead of "(output) gate" comes from the fact that an SLP is quite often defined as a context-free grammar that produces a single word over Σ. This word is defined in the obvious way by iteratively replacing variables by the corresponding right-hand sides, starting with the start variable. We denote this word with $\mathsf{val}(\mathcal{G})$. The unique word over Σ, derived from the variable $A \in V$, is denoted with $\mathsf{val}_{\mathcal{G}}(A)$. We will also allow more general right-hand sides from $(V \cup \Sigma)^*$, but by introducing new variables we can always obtain an equivalent SLP in the above form.

If we have a monoid M, which is finitely generated by the set Σ, then there exists a canonical monoid homomorphism $\eta : \Sigma^* \to M$. Then, an SLP $\mathcal{G}$ over the alphabet Σ can be evaluated over the monoid M, which yields the monoid element $\eta(\mathsf{val}(\mathcal{G}))$. In this paper, we are only interested in the case that the monoid M is a f.g. group G. Let G be finitely generated as a group by Σ. An SLP over the alphabet $\Sigma \cup \Sigma^{-1}$ is also called an SLP over the group G. In this case, we will quite often identify the word $\mathsf{val}(\mathcal{G}) \in (\Sigma \cup \Sigma^{-1})^*$ with the group element $g \in G$ to which it evaluates. We will briefly write "$\mathsf{val}(\mathcal{G}) = g$ in G" in this situation.

The main computational problem we are interested in is the *compressed word problem* for a f.g. group G (with a finite generating set Σ), briefly $\mathsf{CWP}(G)$. The input for this problem is an SLP $\mathcal{G}$ over the alphabet $\Sigma \cup \Sigma^{-1}$, and it is asked whether $\mathsf{val}(\mathcal{G}) = 1$ in G (where of course 1 denotes the group identity). The term "compressed word problem" comes from the fact that this problem can be seen as a succinct version of the classical word problem for G, where the input is an explicitly given word $w \in (\Sigma \cup \Sigma^{-1})^*$ instead of an SLP-compressed word.

The compressed word problem is related to the classical word problem. For instance, the classical word problem for a f.g. subgroup of the automorphism group of a group G can be reduced to the compressed word problem for G, and similar results are known for certain group extensions, see [29] for more details. Groups, for which the compressed word problem can be solved in polynomial time are [29]: finite groups, f.g. nilpotent groups, f.g. free groups, graph groups (also known as right-angled Artin groups or partially commutative groups), and virtually special groups, which are groups that have a finite index subgroup that embeds into a graph group. The latter groups form a rather large class that include for instance Coxeter groups, one-relator groups with torsion, fully residually free groups, and fundamental groups of hyperbolic 3-manifolds. In [11] the parallel complexity of the compressed word problem (there, called the circuit evaluation problem) for finite groups was studied, and the following result was shown:

Theorem 13 ([11]) *Let G be a finite group. If G is solvable, then $\mathsf{CWP}(G)$ belongs to the class $\mathsf{DET} \subseteq \mathsf{NC}^2$. If G is not solvable, then $\mathsf{CWP}(G)$ is P-complete.*

The following two results are proven in [29].

Theorem 14 ([29, Theorem 4.15]) *For every f.g. linear group the compressed word problem belongs to the class coRP.*

This result is shown by reducing the compressed word problem for a f.g. linear group to polynomial identity testing for the ring $\mathbb{Z}$. Also a kind of converse of Theorem 14 is shown in [29]:

Theorem 15 ([29, Theorem 4.16]) *The problems $\mathsf{CWP}(\mathsf{SL}_3(\mathbb{Z}))$ and polynomial identity testing for the ring $\mathbb{Z}$ are polynomial time reducible to each other.*

This result is shown by using the construction of Ben-Or and Cleve [12] for simulating arithmetic circuits by matrix products.

5 Evaluating arithmetic circuits

In the rest of the paper we will deal with the complexity of compressed word problems. For our results in Section 5.1 and 5.2 we will need some results about the complexity of evaluating variable-free arithmetic circuits that we prove in this section.

The following lemma is folklore. We give a proof for completeness.

Lemma 16 *Given an arithmetic circuit $\mathcal{C}$ over some ring $(R, +, \cdot)$ one can compute in logarithmic space two positive circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ over $(R, +, \cdot)$ such that $\mathsf{val}(\mathcal{C}) = \mathsf{val}(\mathcal{C}_1) - \mathsf{val}(\mathcal{C}_2)$. Moreover, for $i \in \{1, 2\}$ we have $\mathsf{deg}(\mathcal{C}_i) \leq \mathsf{deg}(\mathcal{C})$, $\mathsf{depth}(\mathcal{C}_i) \leq 2 \cdot \mathsf{depth}(\mathcal{C})$, and $\mathsf{mdepth}(\mathcal{C}_i) \leq \mathsf{mdepth}(\mathcal{C})$. Finally, if $\mathcal{C}$ is skew then $\mathcal{C}_1$ and $\mathcal{C}_2$ are both skew.*

Proof Let $\mathcal{C} = (V, S, \mathsf{rhs})$ be an arithmetic circuit over the ring $(R, +, \cdot)$. We define the positive circuits $\mathcal{C}_1 = (V', S_1, \mathsf{rhs}')$ and $\mathcal{C}_2 = (V', S_2, \mathsf{rhs}')$ as follows:

- $V' = \{A_i \mid A \in V, i \in \{1, 2\}\}$,
- $\mathsf{rhs}'(A_i) = B_i + C_i$ if $\mathsf{rhs}(A) = B + C$ for $i \in \{1, 2\}$,
- $\mathsf{rhs}'(A_1) = B_1C_1 + B_2C_2$ if $\mathsf{rhs}(A) = B \cdot C$,
- $\mathsf{rhs}'(A_2) = B_1C_2 + B_2C_1$ if $\mathsf{rhs}(A) = B \cdot C$,
- $\mathsf{rhs}'(A_1) = \mathsf{rhs}(A)$ if $\mathsf{rhs}(A) \in \{0, 1\} \cup X$,
- $\mathsf{rhs}'(A_2) = 0$ if $\mathsf{rhs}(A) \in \{0, 1\} \cup X$,
- $\mathsf{rhs}'(A_1) = 0$ if $\mathsf{rhs}(A) = -1$,
- $\mathsf{rhs}'(A_2) = 1$ if $\mathsf{rhs}(A) = -1$.

In fact, the final circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ are obtained by transforming the above circuits into normal form; for this we have to split up the right-hand sides $B_1C_1 + B_2C_2$ and $B_1C_2 + B_2C_1$. The resulting circuits are skew if $\mathcal{C}$ is skew.

We show by induction that for every gate $A \in V$ we have $\mathsf{val}(A) = \mathsf{val}(A_1) - \mathsf{val}(A_2)$. The case that A is an input gate is trivial. Now let A be an addition gate with $\mathsf{rhs}(A) = B + C$ such that the statement is true for B and C. Then

$$\begin{aligned}\mathsf{val}(A) &= \mathsf{val}(B) + \mathsf{val}(C)\\ &= \mathsf{val}(B_1) - \mathsf{val}(B_2) + \mathsf{val}(C_1) - \mathsf{val}(C_2)\\ &= (\mathsf{val}(B_1) + \mathsf{val}(C_1)) - (\mathsf{val}(B_2) + \mathsf{val}(C_2))\\ &= \mathsf{val}(A_1) - \mathsf{val}(A_2)\end{aligned}$$

Finally, let A be a multiplication gate with $\mathsf{rhs}(A) = B \cdot C$. We get

$$\begin{aligned}\mathsf{val}(A) &= \mathsf{val}(B)\mathsf{val}(C)\\ &= (\mathsf{val}(B_1) - \mathsf{val}(B_2))(\mathsf{val}(C_1) - \mathsf{val}(C_2))\\ &= \mathsf{val}(B_1)\mathsf{val}(C_1) + \mathsf{val}(B_2)\mathsf{val}(C_2) - \mathsf{val}(B_1)\mathsf{val}(C_2) - \mathsf{val}(B_2)\mathsf{val}(C_1)\\ &= \mathsf{val}(A_1) - \mathsf{val}(A_2).\end{aligned}$$

So the claim holds. The construction of $\mathcal{C}_1$ and $\mathcal{C}_2$ can be done in logarithmic space, since the right-hand sides of $\mathcal{C}_1$ and $\mathcal{C}_2$ are defined in a purely local way. By induction, it can be shown that for every gate A of $\mathcal{C}$ and every $i \in \{1, 2\}$, one has $\mathsf{deg}(A_i) = \mathsf{deg}(A)$, $\mathsf{depth}(A_i) \leq 2 \cdot \mathsf{depth}(A)$, and $\mathsf{mdepth}(A_i) = \mathsf{mdepth}(A)$. □

We need the following simple lemma, whose proof is based on folklore ideas (see also the proof of [44, Lemma 5.3]):

Lemma 17 *There is an NAuxPDA $\mathcal{P}$ that gets as input a positive variable-free arithmetic circuit $\mathcal{C} = (V, S, \mathsf{rhs})$ over $(\mathbb{Z}, +, \cdot)$ and such that the following properties hold:*

- *The number of accepting computations of $\mathcal{P}$ on input $\mathcal{C}$ is $\mathsf{val}(\mathcal{C})$.*
- *The running time is bounded polynomially in $\mathsf{depth}(\mathcal{C}) \cdot \mathsf{deg}(\mathcal{C})$.*
- *The work tape stores only $O(\log |V|)$ many bits.*
- *The number of bits stored on the pushdown is bounded by $O(\mathsf{mdepth}(\mathcal{C}) \cdot \log |V|)$.*

Proof The NAuxPDA $\mathcal{P}$ stores a sequence of gates on its pushdown and a single gate on its work tape. Every gate can be encoded using $\log(|V|)$ many bits. Initially, the pushdown is empty and the output gate S is stored on the work tape. The NAuxPDA $\mathcal{P}$ proceeds as follows, where A is the gate that is currently stored on the work tape.

- If $\mathsf{rhs}(A) = B + C$, then $\mathcal{P}$ replaces A on the work tape by B or C, where the choice is made nondeterministically. The pushdown is not changed.
- If $\mathsf{rhs}(A) = B \cdot C$, then $\mathcal{P}$ replaces A on the work tape by B and pushes C on the pushdown.
- If $\mathsf{rhs}(A) = 0$, then $\mathcal{P}$ rejects.
- If $\mathsf{rhs}(A) = 1$ and the pushdown is empty, then $\mathcal{P}$ accepts.
- If $\mathsf{rhs}(A) = 1$ and the pushdown is non-empty, then let C be the top gate stored on the pushdown. Then $\mathcal{P}$ replaces A on the work tape by C and pops C from the pushdown.

By induction on the depth of a gate A one shows the following: If $\mathcal{P}$ is started with empty pushdown and the gate on the work tape is A, then:

- the number of accepting computations of $\mathcal{P}$ is $\mathsf{val}(A)$, and
- the running time is bounded by $O(\mathsf{depth}(A) \cdot \mathsf{deg}(A))$ where we assume that each of the subprocedures in the above cases takes time $O(1)$.

This implies the first two statements of the lemma. The third statement is obvious. For the last statement, notice that if $C_1 C_2 \cdots C_k$ is a stack content (with C_k the topmost gate on the stack), then C_i is the right input gate for a multiplication gate A_i and $A_1 A_2 \cdots A_k$ lie along a path in the circuit. Hence, k is bounded by $\mathsf{mdepth}(\mathcal{C})$. □

Corollary 18 *Fix an arbitrary constant $c \geq 0$. The problem, whether a variable-free arithmetic circuit over $(\mathbb{Z}, +, \cdot)$ with multiplication depth at most c evaluates to zero is $\mathsf{C}_=\mathsf{L}$-complete.*

Proof Let us first show the lower bound for $c = 0$ (which implies the lower bound for any $c \geq 0$). As remarked in Section 2.2, the question whether $\mathsf{val}(\mathcal{C}_1) = \mathsf{val}(\mathcal{C}_2)$ for two positive variable-free addition circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ is complete for $\mathsf{C}_=\mathsf{L}$. It suffices to construct from the disjoint union of $\mathcal{C}_1$ and $\mathcal{C}_2$ a variable-free arithmetic circuit (without multiplication gates) such that $\mathsf{val}(\mathcal{C}) = \mathsf{val}(\mathcal{C}_1) - \mathsf{val}(\mathcal{C}_2)$, which is straightforward.

The crucial observation for the upper bound is the fact that for variable-free arithmetic circuits of multiplication depth at most c, the NAuxPDA from Lemma 17 is a nondeterministic logspace machine. Consider a variable-free arithmetic circuit $\mathcal{C}$ over $(\mathbb{Z}, +, \cdot)$ with multiplication depth at most c. By Lemma 16 we obtain from $\mathcal{C}$ in logspace two positive variable-free circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ with $\mathsf{val}(\mathcal{C}) = \mathsf{val}(\mathcal{C}_1) - \mathsf{val}(\mathcal{C}_2)$. The multiplication depth of $\mathcal{C}_1$ and $\mathcal{C}_2$ is still bounded by c. Hence, we can obtain two nondeterministic logspace machines $\mathcal{M}_1$ and $\mathcal{M}_1$ as follows: On input $\mathcal{C}$ as above, the machine $\mathcal{M}_i$ runs the NAuxPDA from Lemma 17 on the circuit $\mathcal{C}_i$ obtained from Lemma 16. This is a deterministic logspace computation followed by a nondeterministic logspace computation, which can be performed by a single nondeterministic logspace machine by standard techniques. From the construction it follows that $\mathsf{val}(\mathcal{C}) = 0$ if and only if the number of accepting computations of $\mathcal{M}_1$ on input $\mathcal{C}$ is equal to the number of accepting computations of $\mathcal{M}_2$ on input $\mathcal{C}$. $\square$

5.1 The compressed word problem for finitely generated nilpotent groups

The main result of this section is:

Theorem 19 *Let $G \neq 1$ be a f.g. torsion-free nilpotent group. Then $\mathsf{CWP}(G)$ is complete for the class $\mathsf{C}_=\mathsf{L}$.*

Every nontrivial torsion-free group contains a copy of $(\mathbb{Z}, +)$ as a subgroup. Since the compressed word problem for a f.g. subgroup H of a group G is logpace reducible to the compressed word problem for G [29, Proposition 4.3], the lower bound in Theorem 19 follows from the following lemma:

Lemma 20 $\mathsf{CWP}(\mathbb{Z}, +)$ *is hard for* $\mathsf{C}_=\mathsf{L}$.

Proof Clearly, an SLP $\mathcal{G}$ over the generator 1 of $\mathbb{Z}$ and its inverse -1 is nothing else than a variable-free arithmetic circuit $\mathcal{C}$ without multiplication gates. Hence, the result is just a reformulation of the lower bound in Corollary 18 for $c = 0$ $\square$

For the upper bound in Theorem 19, we use the fact that every torsion-free f.g. nilpotent group can be embedded into the group $\mathsf{UT}_d(\mathbb{Z})$ for some $d \geq 1$ (Theorem 5). Hence, it suffices to show the following result:

Lemma 21 *For every $d \geq 1$, $\mathsf{CWP}(\mathsf{UT}_d(\mathbb{Z}))$ belongs to $\mathsf{C}_=\mathsf{L}$.*

For the rest of this section let us fix a number $d \geq 1$ and consider the unitriangluar matrix group $\mathsf{UT}_d(\mathbb{Z})$. Consider an SLP $\mathcal{G} = (V, S, \mathsf{rhs})$ over the alphabet $\Gamma_d \cup \Gamma_d^{-1}$ where Γ_d is the finite generating set of $\mathsf{UT}_d(\mathbb{Z})$ from Section 2.3. Note that for every variable $A \in V$, $\mathsf{val}_{\mathcal{G}}(A)$ is a word over the alphabet $\Gamma_d \cup \Gamma_d^{-1}$. We identify in the following this word with the matrix to which it evaluates. Thus, $\mathsf{val}_{\mathcal{G}}(A) \in \mathsf{UT}_d(\mathbb{Z})$.

In order to show Lemma 21, it suffices by Corollary 18 to construct in logspace from an SLP $\mathcal{G}$ as above a variable-free arithmetic circuit $\mathcal{C}$ of multiplication depth at most d such that $\mathcal{G}$ evaluates to the identity matrix if and only if $\mathcal{C}$ evaluates to 0. This is the content of the following lemma. The degree bound in the following lemma will be only needed in Section 5.2.

Lemma 22 *From an SLP $\mathcal{G} = (V, S, \mathsf{rhs})$ over $\mathsf{UT}_d(\mathbb{Z})$ one can compute in logspace a variable-free arithmetic circuit $\mathcal{C}$ over $(\mathbb{Z}, +, \cdot)$ with $\mathsf{mdepth}(\mathcal{C}) \leq d-1$ and $\mathsf{deg}(\mathcal{C}) \leq 2(d-1)$, such that $\mathsf{val}(\mathcal{G}) = \mathsf{Id}_d$ if and only if $\mathsf{val}(\mathcal{C}) = 0$.*

Proof The set of gates of the circuit $\mathcal{C}$ is

$$W = \{A_{i,j} \mid A \in V, 1 \leq i < j \leq d\} \cup \{T\}$$

where T is the output gate. The idea is simple: Gate $A_{i,j}$ will evaluate to the matrix entry $\mathsf{val}_{\mathcal{G}}(A)[i,j]$. To achieve this, we define the right-hand side mapping of the circuit $\mathcal{G}$ (which we denote again with rhs) as follows:

$$\mathsf{rhs}(A_{i,j}) = \begin{cases} M[i,j] & \text{if } \mathsf{rhs}(A) = M \in \Gamma_d \cup \Gamma_d^{-1} \\ B_{i,j} + C_{i,j} + \sum_{i<k<j} B_{i,k} \cdot C_{k,j} & \text{if } \mathsf{rhs}(A) = BC \end{cases}$$

In the first line one has to notice that $M[i,j]$ is one of the numbers $-1, 0, 1$. The second line is simply the rule for matrix multiplication ($A_{i,j} = \sum_{k=1}^{d} B_{i,k} C_{k,j}$) taking into account that all matrices are unitriangular.

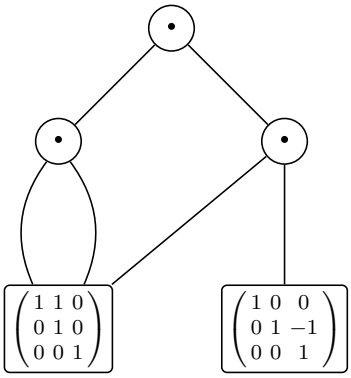
Now, $\mathsf{val}(\mathcal{G})$ is the identity matrix if and only if all matrix entries $\mathsf{val}_{\mathcal{G}}(S)[i,j]$ ($1 \leq i < j \leq d$) are zero. But this is the case if and only if the sum of squares $\sum_{1 \leq i < j \leq d} \mathsf{val}_{\mathcal{G}}(S)[i,j]^2$ is zero. Hence, we finally define

$$\mathsf{rhs}(T) = \sum_{1 \leq i < j \leq d} S_{i,j}^2.$$

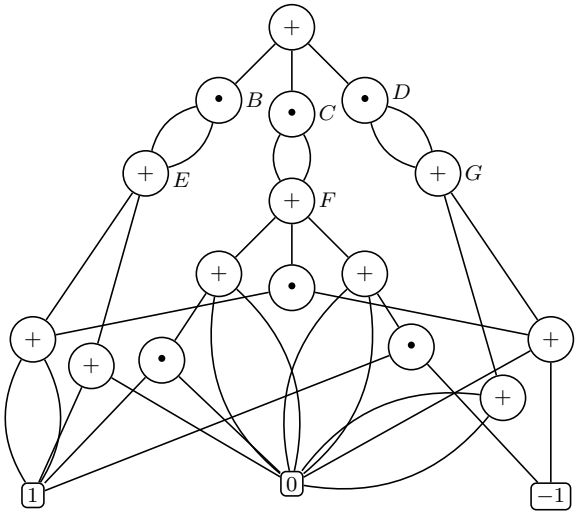
Concerning the multiplication depth, note that the multiplication depth of the gate $A_{i,j}$ is bounded by $j - i - 1$: The only multiplications in $\mathsf{rhs}(A_{i,j})$ are of the form $B_{i,k}C_{k,j}$ for $i < k < j$. Hence, by induction, the multiplication depth of $A_{i,j}$ is bounded by $1 + \max\{k - i - 1, j - k - 1 \mid i < k < j\} = j - i - 1$. It follows that every gate $S_{i,j}$ has multiplication depth at most $d - 2$, which implies that the output gate T has multiplication depth at most $d - 1$.

Similarly, it can be shown by induction that $\mathsf{deg}(A_{i,j}) \leq j - i$. Thus, $\mathsf{deg}(A_{i,j}) \leq d - 1$ for all $1 \leq i < j \leq d$, which implies that the formal degree of the circuit is bounded by $2(d-1)$. This proves the lemma and hence Lemma 21. □

Example 23 Figure 1 illustrates the transformation from an SLP $\mathcal{G}$ over $\mathsf{UT}_3(\mathbb{Z})$ into a variable-free arithmetic circuit over $(\mathbb{Z}, +, \cdot)$ of multiplication depth 2. The three multiplication gates B, C and D are used to square (and subsequently add) the values of E, F and G that represent the matrix entries of $\mathsf{val}(\mathcal{G})$. The rest of the circuit can be split in three parts. The left part beneath E that evaluates to $\mathsf{val}(\mathcal{G})[1,2]$, the right part beneath G that evaluates to $\mathsf{val}(\mathcal{G})[2,3]$ and the middle part that evaluates to $\mathsf{val}(\mathcal{G})[1,3]$. Notice that multiplication gates only appear in the middle part and their inputs are always from the left or the right part or input gates. This property illustrates why the multiplication depth of the arithmetic circuit is 2 and (more general) why the transformation of SLPs over $\mathsf{UT}_d(\mathbb{Z})$ leads to arithmetic circuits with multiplication depth at most $d - 1$.



(a) SLP $\mathcal{G}$ over $\mathsf{UT}_3(\mathbb{Z})$



(b) Circuit $\mathcal{C}$ of multiplication depth 2

Fig. 1: Transformation of an SLP over $\mathsf{UT}_3(\mathbb{Z})$ into a variable-free arithmetic circuit over $(\mathbb{Z}, +, \cdot)$.

So far, we have restricted to *torsion-free* f.g. nilpotent groups. For general f.g. nilpotent groups, we use the fact that every f.g. nilpotent group contains a torsion-free normal f.g. nilpotent subgroup of finite index (Theorem 4) in order to show that the compressed word problem for every f.g. nilpotent group belongs to the complexity class DET: To do this we need the following result:

Theorem 24 *Let G be a finitely generated group. For every normal subgroup H of G with a finite index, $\mathsf{CWP}(G)$ is AC^0-reducible to $\mathsf{CWP}(H)$ and $\mathsf{CWP}(G/H)$.*

Proof To show the lemma, we adopt the proof of [29, Theorem 4.4], where the statement is shown for polynomial time many-one reducibility instead of AC^0-reducibility.

Let G be a finitely generated group with the finite generating set Σ and let H be a normal subgroup of G of finite index (which must be f.g. as well) with the finite generating set Γ. As the generating set for the quotient G/H we can take the set Σ as well. Let $\{Hg_1, \ldots, Hg_n\}$ be the set of cosets of H in G where $g_1 = 1$. Moreover, let $\phi : G \to G/H$ be the canonical homomorphism and let $\eta : (\Sigma \cup \Sigma^{-1})^* \to G$ be the morphism that maps every word from $(\Sigma \cup \Sigma^{-1})^*$ to the group element in G to which it evaluates. Now let $\mathcal{G} = (V, S, \mathsf{rhs}_\mathcal{G})$ be an SLP over the alphabet $\Sigma \cup \Sigma^{-1}$. We have to construct an AC^0-circuit with oracle gates for $\mathsf{CWP}(H)$ and $\mathsf{CWP}(G/H)$ that checks whether $\mathsf{val}(\mathcal{G}) = 1$ in G.

Consider the set of triples

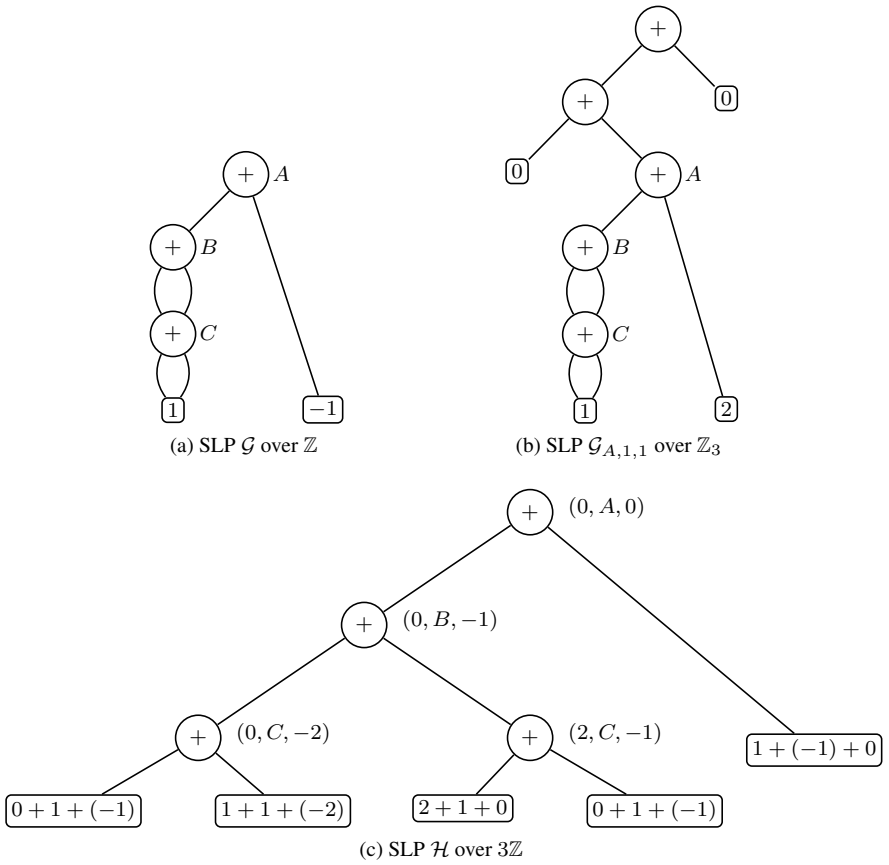
$$W = \left\{ [g_i, A, g_j^{-1}] \mid A \in V, 1 \leq i, j \leq n, g_i \eta(\mathsf{val}_\mathcal{G}(A)) g_j^{-1} \in H \right\}.$$

In a first step, we construct the set of all these triples in parallel using $n^2|V|$ oracle gates for $\mathsf{CWP}(G/H)$. More precisely, we construct for all $A \in V, 1 \leq i, j \leq n$ an SLP $\mathcal{G}_{A,i,j}$ that evaluates to the group element $\phi(g_i \eta(\mathsf{val}_\mathcal{G}(A)) g_j^{-1}) \in G/H$. For this, we take the SLP $\mathcal{G}$ and add a new start variable $S_{A,i,j}$ with the right-hand side $w_i A w_j^{-1}$ where $w_i \in (\Sigma \cup \Sigma^{-1})^*$ is a word that represents the group element g_i. We do not need to compute these words w_i; they can be "hard-wired" into the circuit. The SLP $\mathcal{G}_{A,i,j}$ can be clearly constructed in AC^0, and we have $\mathsf{val}(\mathcal{G}_{A,i,j}) = 1$ in G/H if and only if $g_i \eta(\mathsf{val}_\mathcal{G}(A)) g_j^{-1} \in H$.

Note that $\mathsf{val}(\mathcal{G}_{S,1,1}) = 1$ in G/H if and only if $\mathsf{val}(\mathcal{G})$ represents an element of the subgroup H. Thus, if it turns out that $\mathsf{val}(\mathcal{G}_{S,1,1}) \neq 1$ in G/H (which can be checked with an oracle gate for $\mathsf{CWP}(G/H)$), then the whole circuit will output zero. Otherwise (i.e., in case $\eta(\mathsf{val}(\mathcal{G})) \in H$), we construct in AC^0 an SLP $\mathcal{H}$ over the alphabet $\Gamma \cup \Gamma^{-1}$ (the monoid generating set for H) that will represent the group element $\eta(\mathsf{val}(\mathcal{G}))$. This SLP is then fed into an oracle gate for $\mathsf{CWP}(H)$, and the output bit of this oracle gate is the output of the whole circuit.

The variable set of $\mathcal{H}$ is W, the start variable is $[g_1, S, g_1^{-1}]$ and the right-hand sides are defined as follows: If $\mathsf{rhs}_\mathcal{G}(A) = a \in \Sigma \cup \Sigma^{-1}$, we set $\mathsf{rhs}_\mathcal{H}([g_i, A, g_j^{-1}]) = w_{a,i,j}$ where $w_{a,i,j} \in (\Gamma \cup \Gamma^{-1})^*$ is a word that represents the group element $g_i a g_j^{-1} = g_i \eta(\mathsf{val}_\mathcal{G}(A)) g_j^{-1} \in H$. Note again, that we do not have to compute these words $w_{a,i,j}$ (they are fixed). If $\mathsf{rhs}_\mathcal{G}(A) = BC$ and $[g_i, A, g_j^{-1}] \in W$, then we determine the unique k, so that $g_i \eta(\mathsf{val}_\mathcal{G}(B)) g_k^{-1} \in H$. To do this we have to go through the set W and look for the unique k such that $[g_i, B, g_k^{-1}] \in H$. Now we define $\mathsf{rhs}_\mathcal{H}([g_i, A, g_j^{-1}]) = [g_i, B, g_k^{-1}][g_k, C, g_j^{-1}]$. Clearly, this construction can be carried out by an AC^0-circuit. Finally, it is straightforward to show that $\mathsf{val}_\mathcal{H}([g_i, A, g_j^{-1}])$ represents the group element $g_i \eta(\mathsf{val}_\mathcal{G}(A)) g_j^{-1} \in H$. Hence, we have $\mathsf{val}(\mathcal{G}) = 1$ in G, if and only if $\mathsf{val}(\mathcal{H}) = 1$ in H. This finishes our reduction. Note that the overall circuit consists of $n^2|V|$ parallel $\mathsf{CWP}(G/H)$-oracle gates followed by a single $\mathsf{CWP}(H)$-oracle gate. □

Example 25 Figure 2 illustrates the proof of Theorem 24 with the group $G = (\mathbb{Z}, +)$, the normal subgroup $H = (3\mathbb{Z}, +)$ and the finite quotient $G/H \cong (\mathbb{Z}_3, +)$. As usual the representing elements of the cosets are chosen as $g_1 = 0$, $g_2 = 1$ and



(a) SLP $\mathcal{G}$ over $\mathbb{Z}$

(b) SLP $\mathcal{G}_{A,1,1}$ over $\mathbb{Z}_3$

(c) SLP $\mathcal{H}$ over $3\mathbb{Z}$

Fig. 2: Illustration of the proof of Lemma 24.

$g_3 = 2$. In picture (a) there is an SLP $\mathcal{G}$ over $\mathbb{Z}$ that evaluates to 3, so in particular, $\mathsf{val}(\mathcal{G}) \in H$ which is tested by the SLP $\mathcal{G}_{A,1,1}$ in picture (b) over $\mathbb{Z}_3$. Finally in picture (c) there is the SLP $\mathcal{H}$ over H (notice that all inputs evaluate to elements of H) that evaluates to the same element as $\mathcal{G}$ does. For the construction of $\mathcal{H}$ we calculate the set $W = \{(a, A, b^{-1}) \mid A \in V, a, b, \in \{0, 1, 2\}, a + \mathsf{val}(A) - b \bmod 3 = 0\}$ by the SLPs $\mathcal{G}_{A,i,j}$ and take the needed gates from this set as described in the proof.

We can now show:

Theorem 26 *For every f.g. nilpotent group, the compressed word problem is in* DET.

Proof Let G be a f.g. nilpotent group. If G is finite, then the result follows from Theorem 13 (every nilpotent group is solvable). If G is infinite, then by Theorem 4, G has a torsion-free normal subgroup H of finite index. By Theorem 3, H and G/H are nilpotent too; moreover H is finitely generated. By Theorem 19, $\mathsf{CWP}(H)$ belongs to $\mathsf{C}_{=}\mathsf{L} \subseteq \mathsf{DET}$. Moreover, by Theorem 13, $\mathsf{CWP}(G/H)$ belongs to DET as well. Finally, Theorem 24 implies that $\mathsf{CWP}(G)$ belongs to DET. □

Actually, Theorem 26 can be slightly extended to groups that are (f.g. nilpotent)-by-(finite solvable) (i.e., groups that have a normal subgroup, which is f.g. nilpotent, and where the quotient is finite solvable. This follows from Theorem 24 and the fact that the compressed word problem for a finite solvable group belongs to DET (Theorem 13).

Recently, Myasnikov and Weiß [34] prove that for every f.g. nilpotent group G, the following special case of the compressed word problem can be solved in $\mathsf{DLOGTIME}$-uniform TC^0: Given group elements $g_1, \ldots, g_n \in G$ and binary encoded integers $e_1, \ldots, e_n$, does $g_1^{e_1} \cdots g_n^{e_n} = 1$ hold in G? It is straightforward to construct a small SLP for $g_1^{e_1} \cdots g_n^{e_n}$.

5.2 The uniform compressed word problem for unitriangular groups

For Lemma 21 it is crucial that the dimension d is a constant. In this section, we consider a uniform variant of the compressed word problem for $\mathsf{UT}_d(\mathbb{Z})$. We denote this problem with $\mathsf{CWP}(\mathsf{UT}_*(\mathbb{Z}))$. The input consists of a unary encoded number d and an SLP, whose terminal symbols are generators of $\mathsf{UT}_d(\mathbb{Z})$ or their inverses. Alternatively, we can assume that the terminal symbols are arbitrary matrices from $\mathsf{UT}_d(\mathbb{Z})$ with binary encoded entries (given such a matrix M, it is easy to construct an SLP over the generator matrices that produces M). The question is whether the SLP evaluates to the identity matrix. We show that this problem is complete for the complexity class $\mathsf{C}_=\mathsf{LogCFL}$. Recall the definition of that class and the definition of NAuxPDA from the end of Section 2.2.

Theorem 27 *The problem* $\mathsf{CWP}(\mathsf{UT}_*(\mathbb{Z}))$ *is complete for* $\mathsf{C}_=\mathsf{LogCFL}$.

Proof We start with the upper bound. Consider an SLP $\mathcal{G}$, whose terminal symbols are generators of $\mathsf{UT}_d(\mathbb{Z})$ or their inverses. The dimension d is clearly bounded by the input size. Consider the variable-free arithmetic circuit $\mathcal{C}$ constructed from $\mathcal{G}$ in Lemma 22 and let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the two variable-free positive arithmetic circuits obtained from $\mathcal{C}$ using Lemma 16. Then $\mathcal{G}$ evaluates to the identity matrix if and only if $\mathsf{val}(\mathcal{C}_1) = \mathsf{val}(\mathcal{C}_2)$. Moreover, the formal degrees $\mathsf{deg}(\mathcal{C}_1)$ and $\mathsf{deg}(\mathcal{C}_2)$ are bounded by $2(d-1)$, i.e., polynomially bounded in the input length. We now compose a logspace machine that computes from the input SLP $\mathcal{G}$ the circuit $\mathcal{C}_i$ with the NAuxPDA from Lemma 17 to get an NAuxPDA $\mathcal{P}_i$ such that the number of accepting computation paths of $\mathcal{P}_i$ on input $\mathcal{G}$ is exactly $\mathsf{val}(\mathcal{C}_i)$. Moreover, the running time of $\mathcal{P}_i$ on input $\mathcal{G}$ is bounded polynomially in $(2d-1) \cdot \mathsf{depth}(\mathcal{C}_i) \in O(d \cdot |\mathcal{G}|)$.

Let us now show that $\mathsf{CWP}(\mathsf{UT}_*(\mathbb{Z}))$ is hard for the class $\mathsf{C}_=\mathsf{LogCFL}$. Let $f_1, f_2 : \{0,1\}^* \to \mathbb{N}$ be two functions in $\#\mathsf{LogCFL}$. As explained in Section 2.2, there exist two logspace-uniform families of positive arithmetic circuits $(\mathcal{C}_{1,n})_{n\geq 0}$ and $(\mathcal{C}_{2,n})_{n\geq 0}$, having polynomially bounded size and formal degree [44] such that, for $i \in \{1,2\}$, $(\mathcal{C}_{i,n})_{n\geq 0}$ computes f_i. Let $w = a_1 a_2 \cdots a_n \in \{0,1\}^n$ be an input for the n-th circuits $\mathcal{C}_{1,n}$ and $\mathcal{C}_{2,n}$. Let $\mathcal{C}_i = (V_i, S_i, \mathsf{rhs}_i)$ be the variable-free positive arithmetic circuit obtained from $\mathcal{C}_{i,n}$ by replacing every x_j-labelled input gate by the input bit $a_j \in \{0,1\}$. We can assume that $V_1 \cap V_2 = \emptyset$. Moreover, by [4, Lemma 3.2] we can assume that every gate of $\mathcal{C}_i$ is labelled by its formal degree. By adding if

necessary additional multiplication gates where one input is set to 1 (which adds 1 to the formal degree), we can assume that $\mathcal{C}_1$ and $\mathcal{C}_2$ have the same formal degree $d \leq p(n)$ for a polynomial p. Analogously, we can assume that if A is an addition gate in $\mathcal{C}_1$ or $\mathcal{C}_2$ with right-hand side $B + C$, then $\mathsf{deg}(B) = \mathsf{deg}(C) = \mathsf{deg}(A)$. All these preprocessing steps can be carried out in logarithmic space. Below, we write $\mathsf{val}_i(A)$ for $\mathsf{val}_{\mathcal{C}_i}(A)$, where $A \in V_i$.

We will construct in logarithmic space an SLP $\mathcal{G} = (V, S, \mathsf{rhs})$ over the alphabet $\Gamma_{d+1} \cup \Gamma_{d+1}^{-1}$, where Γ_{d+1} is our canonical generating set for the matrix group $\mathsf{UT}_{d+1}(\mathbb{Z})$ (see Section 2.3), such that $\mathsf{val}(\mathcal{G})$ evaluates to the identity matrix if and only if $\mathcal{C}_1$ and $\mathcal{C}_2$ evaluate to the same number. The latter means that $f_1(w) = f_2(w)$. As before, we will identify words over the alphabet $\Gamma_{d+1} \cup \Gamma_{d+1}^{-1}$ with the matrices to which they evaluate.

We set $V = \{A_j^b \mid A \in V_1 \cup V_2, 1 \leq j \leq d + 1 - \mathsf{deg}(A), b \in \{-1, 1\}\}$. We define rhs in such a way that $\mathsf{val}_{\mathcal{G}}(A_j^b) = T_{j,j+e}^{b \cdot v}$ where $v = \mathsf{val}_i(A)$ for $A \in V_i$ with $e = \mathsf{deg}(A)$ (recall the definition of $T_{i,j}$ from Section 2.3). Let $A \in V_i$, $e = \mathsf{deg}(A)$, $1 \leq j \leq d + 1 - e$ and $b \in \{-1, 1\}$.

Case 1. $\mathsf{rhs}_i(A) = 0$. We set $\mathsf{rhs}(A_j^b) = \mathsf{Id}$, which is $T_{j,j+1}^0$. Note that $e = 1$. Correctness is obvious in this case.

Case 2. $\mathsf{rhs}_i(A) = 1$. We set $\mathsf{rhs}(A_j^b) = T_{j,j+1}^b$. Correctness is again obvious.

Case 3. $\mathsf{rhs}_i(A) = B + C$. Note that $\mathsf{deg}(A) = \mathsf{deg}(B) = \mathsf{deg}(C) = e$. We set $\mathsf{rhs}(A_j^b) = B_j^b C_j^b$. Correctness follows immediately by induction: Assume that $\mathsf{val}(B_j^b) = T_{j,j+e}^{b \cdot \mathsf{val}_i(B)}$ and $\mathsf{val}(C_j^b) = T_{j,j+e}^{b \cdot \mathsf{val}_i(C)}$. We get

$$\begin{aligned}\mathsf{val}(A_j^b) = \mathsf{val}(B_j^b)\mathsf{val}(C_j^b) &= T_{j,j+e}^{b \cdot \mathsf{val}_i(B)} T_{j,j+e}^{b \cdot \mathsf{val}_i(C)} \\ &= T_{j,j+e}^{b \cdot (\mathsf{val}_i(B) + \mathsf{val}_i(C))} = T_{j,j+e}^{b \cdot \mathsf{val}_i(A)}.\end{aligned}$$

Note that the gates B_j^b and C_j^b exist for all $1 \leq j \leq d + 1 - e$, since $\mathsf{deg}(B) = \mathsf{deg}(C) = e$.

Case 4. $\mathsf{rhs}_i(A) = B \cdot C$. We set $\mathsf{rhs}(A_j^1) = B_j^{-1} C_k^{-1} B_j^1 C_k^1$ and $\mathsf{rhs}(A_j^{-1}) = C_k^{-1} B_j^{-1} C_k^1 B_j^1$ where $k = j + \mathsf{deg}(B)$. Since $1 \leq j \leq d + 1 - e$ and $e = \mathsf{deg}(A) = \mathsf{deg}(B) + \mathsf{deg}(C)$, we have $1 \leq j \leq d + 1 - \mathsf{deg}(B)$ and $1 \leq k \leq d + 1 - \mathsf{deg}(C)$, which means that the gates B_j^{-1}, B_j^1, C_k^{-1} and C_k^1 indeed exist. Correctness follows from Lemma 2 and induction (note that $k + \mathsf{deg}(C) = j + e$):

$$\begin{aligned}\mathsf{val}(A_j^1) &= \mathsf{val}(B_j^{-1})\mathsf{val}(C_k^{-1})\mathsf{val}(B_j^1)\mathsf{val}(C_k^1) \\ &= T_{j,k}^{-\mathsf{val}_i(B)} T_{k,j+e}^{-\mathsf{val}_i(C)} T_{j,k}^{\mathsf{val}_i(B)} T_{k,j+e}^{\mathsf{val}_i(C)} \\ &= T_{j,j+e}^{\mathsf{val}_i(B) \cdot \mathsf{val}_i(C)} = T_{j,j+e}^{\mathsf{val}_i(A)}.\end{aligned}$$

A similar calculation can be done for A_j^{-1}. Finally, for the start variable S of $\mathcal{G}$, we set $\mathsf{rhs}(S) = (S_1)_1^1 (S_2)_1^{-1}$. Hence, we get

$$\mathsf{val}(S) = T_{1,d+1}^{\mathsf{val}(\mathcal{C}_1)} T_{1,d+1}^{-\mathsf{val}(\mathcal{C}_2)} = T_{1,d+1}^{\mathsf{val}(\mathcal{C}_1) - \mathsf{val}(\mathcal{C}_2)}.$$

Hence, $\mathsf{val}(\mathcal{G})$ is the identity matrix if and only if $\mathsf{val}(\mathcal{C}_1) = \mathsf{val}(\mathcal{C}_2)$, which concludes the proof. □

5.3 The compressed word problem for polycyclic groups

In this section we consider the compressed word problem for polycyclic groups. Since every polycyclic group is f.g. linear, the compressed word problem for a polycyclic group can be reduced to polynomial identity testing. In this section, we show a lower bound: There exists a strongly polycyclic group G (which is also metabelian) such that polynomial identity testing for skew arithmetic circuits can be reduced to $\mathsf{CWP}(G)$.

Let us start with a specific example of a polycyclic group. Consider the two matrices

$$g_a = \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix} \text{ and } h = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{4}$$

where $a \in \mathbb{R}$, $a \geq 2$. Let $G_a = \langle g_a, h \rangle \leq \mathsf{GL}_2(\mathbb{R})$. Let us remark that, for instance, the group G_2 is not polycyclic, see e.g. [47, p. 56]. On the other hand, we have:

Proposition 28 *The group $G = G_{1+\sqrt{2}}$ is polycyclic and metabelian.*[3]

Proof We show that the commutator subgroup of G is isomorphic to $\mathbb{Z} \times \mathbb{Z}$, which implies the theorem. First we calculate the commutator subgroup of G. It is known that the commutator subgroup of a group generated by two elements g_1, g_2 is generated by all commutators $g_1^s g_2^t g_1^{-s} g_2^{-t}$ for $s, t \in \mathbb{Z}$ [32]. Hence,

$$[G, G] = \langle M_{s,t} \mid s, t \in \mathbb{Z} \rangle$$

where for $s, t \in \mathbb{Z}$ we set

$$\begin{aligned} M_{s,t} &= \begin{pmatrix} 1+\sqrt{2} & 0 \\ 0 & 1 \end{pmatrix}^s \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^t \begin{pmatrix} 1+\sqrt{2} & 0 \\ 0 & 1 \end{pmatrix}^{-s} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{-t} \\ &= \begin{pmatrix} (1+\sqrt{2})^s & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} (1+\sqrt{2})^{-s} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -t \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} (1+\sqrt{2})^s & t(1+\sqrt{2})^s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} (1+\sqrt{2})^{-s} & -t(1+\sqrt{2})^{-s} \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & -t + t(1+\sqrt{2})^s \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & t((1+\sqrt{2})^s - 1) \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

With the setting

$$u = \begin{pmatrix} 1 & \sqrt{2} \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad v = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$

we show that $\langle M_{s,t} \mid s, t \in \mathbb{Z} \rangle = \langle u, v \rangle$. Moreover, it is easy to see that u and v generate a copy of $\mathbb{Z} \times \mathbb{Z}$.

[3] It is probably known to experts that G is polycyclic. Since we could not find an explicit proof, we present the arguments for completeness.

We have $M_{1,1} = u$ and

$$M_{2,1}M_{1,1}^{-2} = \begin{pmatrix} 1 & 2+2\sqrt{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -2\sqrt{2} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} = v.$$

This shows that $\langle u, v\rangle \subseteq \langle M_{s,t} \mid s,t \in \mathbb{Z}\rangle$. For the other inclusion assume first that $s \geq 0$. Then

$$\begin{aligned} t\left(\left(1+\sqrt{2}\right)^s - 1\right) &= t\left(\left(\sum_{i=0}^{s} \binom{s}{i} \sqrt{2}^i\right) - 1\right) \\ &= t\left(\sum_{i=1}^{s} \binom{s}{i} \sqrt{2}^i\right) \\ &= t\left(\sum_{i=1}^{\lfloor \frac{s}{2} \rfloor} \binom{s}{2i} \sqrt{2}^{2i} + \sum_{i=1}^{\lceil \frac{s}{2} \rceil} \binom{s}{2i-1} \sqrt{2}^{2i-1}\right) \\ &= 2\left(\sum_{i=1}^{\lfloor \frac{s}{2} \rfloor} t\binom{s}{2i} 2^{i-1}\right) + \sqrt{2}\left(\sum_{i=1}^{\lceil \frac{s}{2} \rceil} t\binom{s}{2i-1} 2^{i-1}\right). \end{aligned}$$

So with

$$c_1 = \sum_{i=1}^{\lfloor \frac{s}{2} \rfloor} t\binom{s}{2i} 2^{i-1} \in \mathbb{Z} \quad \text{and} \quad c_2 = \sum_{i=1}^{\lceil \frac{s}{2} \rceil} t\binom{s}{2i-1} 2^{i-1} \in \mathbb{Z}$$

we get

$$M_{s,t} = \begin{pmatrix} 1 & t((1+\sqrt{2})^s - 1) \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2c_1 + \sqrt{2}c_2 \\ 0 & 1 \end{pmatrix} = v^{c_1}u^{c_2}.$$

For $s < 0$ we get with $a = -s \bmod 2$:

$$\begin{aligned} t\left(\left(1+\sqrt{2}\right)^s - 1\right) &= t\left(\left(\sqrt{2}-1\right)^{-s} - 1\right) \\ &= t\left(\left(\sum_{i=0}^{-s} \binom{-s}{i} (\sqrt{2})^i (-1)^{-s-i}\right) - 1\right) \\ &= t\left(-2a + \sum_{i=1}^{-s} \binom{-s}{i} (\sqrt{2})^i (-1)^{-s-i}\right) \\ &= t\left(-2a + \sum_{i=1}^{\lfloor \frac{-s}{2} \rfloor} \binom{-s}{2i} (\sqrt{2})^{2i} (-1)^{-s-2i}\right) + \\ &\quad t\sum_{i=1}^{\lceil \frac{-s}{2} \rceil} \binom{-s}{2i-1} (\sqrt{2})^{2i-1} (-1)^{-s-(2i-1)} \end{aligned}$$

$$= 2\left(-at + \sum_{i=1}^{\lfloor \frac{-s}{2} \rfloor} t\binom{-s}{2i} 2^{i-1}(-1)^{-s-2i}\right) + \sqrt{2}\left(\sum_{i=1}^{\lceil \frac{-s}{2} \rceil} t\binom{-s}{2i-1} 2^{i-1}(-1)^{-s-(2i-1)}\right).$$

So with

$$c_1 = -at + \sum_{i=1}^{\lfloor \frac{-s}{2} \rfloor} t\binom{-s}{2i} 2^{i-1}(-1)^{-s-2i} \in \mathbb{Z}$$

and

$$c_2 = \sum_{i=1}^{\lceil \frac{-s}{2} \rceil} t\binom{-s}{2i-1} 2^{i-1}(-1)^{-s-(2i-1)} \in \mathbb{Z}$$

we get

$$M_{s,t} = \begin{pmatrix} 1 & t((1+\sqrt{2})^s - 1) \\ 0 & 1 \end{pmatrix} = v^{c_1}u^{c_2}.$$

This shows that $\langle M_{s,t} \mid s,t \in \mathbb{Z}\rangle \subseteq \langle u, v\rangle$. $\square$

The main result of this section is:

Theorem 29 *Let $a \geq 2$. Polynomial identity testing for skew arithmetic circuits over $(\mathbb{Z}, +, \cdot)$ is logspace-reducible to the compressed word problem for the group G_a.*

In particular, there exist polycyclic groups for which the compressed word problem is at least as hard as polynomial identity testing for skew circuits. Recall that it is not known, whether there exists a polynomial time algorithm for polynomial identity testing restricted to skew arithmetic circuits. It also follows from Theorem 29 that the compressed word problem for the Baumslag-Solitar group $\langle a, t \mid t^{-1}at = a^2\rangle \cong G_2$ is at least as hard as polynomial identity testing for skew circuits.

For the proof of Theorem 29, we will make use of the following lemma. It is a result from [3] (see the proof of Proposition 2.2 in [3], where the result is shown for $a = 2$, but the proof immediately generalizes to any $a \geq 2$):

Lemma 30 *Let $a \geq 2$ be a real number and let $\mathcal{C}$ be an arithmetic circuit over $(\mathbb{Z}, +, \cdot)$ of size n. Let $\mathsf{val}(\mathcal{C}) = p(x_1, \ldots, x_m)$. Then $p(x_1, \ldots, x_m)$ is the zero-polynomial if and only if $p(\alpha_1, \ldots, \alpha_m) = 0$ where $\alpha_i = a^{2^{i \cdot n^2}}$ for $1 \leq i \leq m$.*

Proof of Thereom 29. Let us fix a skew arithmetic circuit $\mathcal{C}$ of size n with m variables $x_1, \ldots, x_m$. Consider the two positive arithmetic circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ obtained from Lemma 16, Notice that $\mathcal{C}_1$ and $\mathcal{C}_2$ are skew. Let $\mathsf{val}(\mathcal{C}) = p(x_1, \ldots, x_m)$ and $\mathsf{val}(\mathcal{C}_j) = p_j(x_1, \ldots, x_m)$ for $j \in \{1, 2\}$. By Lemma 16 $p_1(x_1, \ldots, x_m) = p_2(x_1, \ldots, x_m)$ if and only if $p(x_1, \ldots, x_m) = 0$. Let $\alpha_1, \ldots, \alpha_m$ be as in Lemma 30. For $j \in \{1, 2\}$ we will define an SLP $\mathcal{G}_j$ over the alphabet $\{g_a, g_a^{-1}, h, h^{-1}\}$ such that $\mathsf{val}(\mathcal{G}_j) = h^{p_j(\alpha_1, \ldots, \alpha_m)}$.

First of all, using iterated squaring, we can construct an SLP $\mathcal{H}$ with variables $B_1, B_1^{-1}, \ldots, B_m, B_m^{-1}$ (and some other auxiliary variables) such that

$$\mathsf{val}_{\mathcal{H}}(B_i) = g_a^{2^{i \cdot n^2}} = \begin{pmatrix} a^{2^{i \cdot n^2}} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha_i & 0 \\ 0 & 1 \end{pmatrix} \text{ and}$$

$$\mathsf{val}_{\mathcal{H}}(B_i^{-1}) = g_a^{-2^{i \cdot n^2}} = \begin{pmatrix} a^{-2^{i \cdot n^2}} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha_i^{-1} & 0 \\ 0 & 1 \end{pmatrix}.$$

We now construct the SLP $\mathcal{G}_j$ as follows: The set of variables of $\mathcal{G}_j$ consists of the gates of $\mathcal{C}_j$ and the variables of $\mathcal{H}$. We copy the right-hand sides from $\mathcal{H}$ and define the right-hand side for a gate A of $\mathcal{C}_j$ as follows:

$$\mathsf{rhs}_{\mathcal{G}_j}(A) = \begin{cases} \mathsf{Id} & \text{if } \mathsf{rhs}_{\mathcal{C}_j}(A) = 0 \text{ or } \mathsf{rhs}_{\mathcal{C}_j}(A) = 0 \cdot B \\ h & \text{if } \mathsf{rhs}_{\mathcal{C}_j}(A) = 1 \\ BC & \text{if } \mathsf{rhs}_{\mathcal{C}_j}(A) = B + C \\ B & \text{if } \mathsf{rhs}_{\mathcal{C}_j}(A) = 1 \cdot B \\ B_i B B_i^{-1} & \text{if } \mathsf{rhs}_{\mathcal{C}_j}(A) = x_i \cdot B. \end{cases}$$

We claim that for every gate A of $\mathcal{C}_j$ we have the following, where we denote for better readability the polynomial $\mathsf{val}_{\mathcal{C}_j}(A)$ to which gate A evaluates with p_A:

$$\mathsf{val}_{\mathcal{G}_j}(A) = \begin{pmatrix} 1 & p_A(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix}.$$

The cases that $\mathsf{rhs}_{\mathcal{C}_j}(A)$ is a constant, $0 \cdot B$, resp., $1 \cdot B$ are obvious. If $\mathsf{rhs}_{\mathcal{C}_j}(A) = B + C$ then we obtain by induction

$$\begin{aligned} \mathsf{val}_{\mathcal{G}_j}(A) &= \mathsf{val}_{\mathcal{G}_j}(B)\mathsf{val}_{\mathcal{G}_j}(C) \\ &= \begin{pmatrix} 1 & p_B(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & p_C(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & p_B(\alpha_1, \ldots, \alpha_m) + p_C(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & p_A(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

Finally, if $\mathsf{rhs}_{\mathcal{C}_j}(A) = x_i \cdot B$ then we obtain by induction

$$\begin{aligned} \mathsf{val}_{\mathcal{G}_j}(A) &= \begin{pmatrix} \alpha_i & 0 \\ 0 & 1 \end{pmatrix} \mathsf{val}_{\mathcal{G}_j}(B) \begin{pmatrix} \alpha_i^{-1} & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \alpha_i & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & p_B(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_i^{-1} & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \alpha_i & \alpha_i \cdot p_B(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_i^{-1} & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & \alpha_i \cdot p_B(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & p_A(\alpha_1, \ldots, \alpha_m) \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

Now we define the SLP $\mathcal{G}_2'$ by replacing in $\mathcal{G}_2$ every right-hand side $A \cdot B$ by $B \cdot A$ and replacing every input value by its inverse. This directly yields $\mathsf{val}(\mathcal{G}_2') = \mathsf{val}(\mathcal{G}_2)^{-1} = h^{-p_2(\alpha_1,\ldots,\alpha_m)}$. Finally we define the SLP $\mathcal{G}$ as the concatenation of $\mathcal{G}_1$ and $\mathcal{G}_2'$. We get

$$\begin{aligned}\mathsf{val}(\mathcal{G}) &= \mathsf{val}(\mathcal{G}_1) \cdot \mathsf{val}(\mathcal{G}_2') \\ &= h^{p_1(\alpha_1,\ldots,\alpha_m)} \cdot h^{-p_2(\alpha_1,\ldots,\alpha_m)} \\ &= h^{p_1(\alpha_1,\ldots,\alpha_m)-p_2(\alpha_1,\ldots,\alpha_m)},\end{aligned}$$

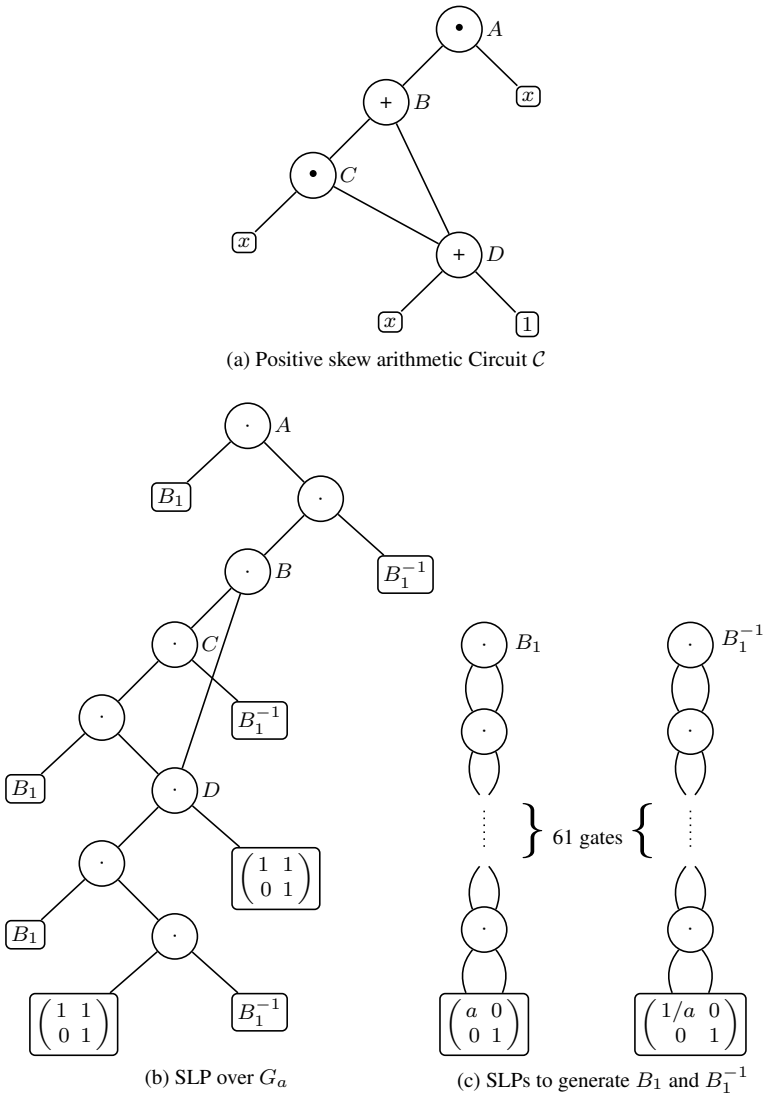
which is the group identity if and only if $p_1(\alpha_1, \ldots, \alpha_m) - p_2(\alpha_1, \ldots, \alpha_m) = p(\alpha_1, \ldots, \alpha_m) = 0$. By Lemma 30 this is equivalent to $\mathsf{val}(\mathcal{C}) = p(x_1, \ldots, x_m) = 0$. □

Example 31 In Figure 3 we start with a positive skew arithmetic circuit $\mathcal{C}$ with $|\mathcal{C}| = 8$ and transform it into an SLP $\mathcal{G}$ over G_a ($a \geq 2$), as explained in the proof of Thereom 29. Here $\alpha = a^{2^{64}}$, so we need 64 multiplication gates plus one input gate in the subSLP beneath gate B_1 (resp. B_1^{-1}) in order to get $\mathsf{val}(B_1) = g_a^{2^{64}}$ (resp. $\mathsf{val}(B_1^{-1}) = a^{-2^{64}}$). We split the figure in two parts: on the left there is the SLP $\mathcal{G}$, where B_1 and B_1^{-1} are left as inputs and on the right there are the two subSLPs for B_1 and B_1^{-1}.

Actually, we can carry out the above reduction for a class of arithmetic circuits that is slightly larger than the class of skew arithmetic circuits. Let us define a *powerful skew circuit* as an arithmetic circuit where for every multiplication gate A, $\mathsf{rhs}(A)$ is of the form $\alpha \cdot \prod_{i=1}^m x_i^{e_i} \cdot B$ for a gate B, binary coded integers $\alpha, e_1, \ldots, e_m$ ($e_i \geq 0$), and variables $x_1, \ldots, x_m$. Such a circuit can be converted into an ordinary arithmetic circuit (the big powers $x_i^{e_i}$ can be produced by iterated squaring), which, however is no longer skew. To extend the reduction from the proof of Thereom 29 to powerful skew circuits, first note that in a right-hand side $\alpha \cdot \prod_{i=1}^m x_i^{e_i} \cdot B$ we can assume that $\alpha = 1$, since we can obtain $\alpha \cdot \prod_{i=1}^m x_i^{e_i} \cdot B$ from $\prod_{i=1}^m x_i^{e_i} \cdot B$ using additional addition gates. For a gate A with $\mathsf{rhs}_{\mathcal{C}}(A) = \prod_{i=1}^m x_i^{e_i} \cdot B$ we set $\mathsf{rhs}_{\mathcal{G}}(A) = \prod_{i=1}^m B_i^{e_i} B \prod_{i=1}^m B_i^{-e_i}$. The powers $B_i^{e_i}$ and $B_i^{-e_i}$ can be defined using additional multiplication gates. In [25], we introduced powerful skew circuits, and proved that for this class, polynomial identity testing can be solved in coRNC^2. Using this result, we showed that the compressed word problem for a wreath product $G \wr \mathbb{Z}^n$ where G is a direct product of copies of $\mathbb{Z}$ and groups $\mathbb{Z}_p$ for primes p, belongs to coRNC^2.

Let us look again at the group $G = G_{1+\sqrt{2}}$ from Proposition 28. Its commutator subgroup is isomorphic to $\mathbb{Z} \times \mathbb{Z}$. Moreover, the quotient $G/[G, G]$ is isomorphic to $\mathbb{Z} \times \mathbb{Z}_2$: The G-generator h from (4) satisfies $h^2 \in [G, G]$, whereas the generator $g_{1+\sqrt{2}}$ has infinite order in the quotient. Hence, G has a subnormal series of the form $G \rhd H \rhd \mathbb{Z} \times \mathbb{Z} \rhd \mathbb{Z} \rhd 1$ where H has index two in G and $H/(\mathbb{Z} \times \mathbb{Z}) \cong \mathbb{Z}$. The group H is strongly polycyclic and has Hirsch length three. By Theorem 24 we obtain:

Corollary 32 *There is a strongly polycyclic group H of Hirsch length 3 such that polynomial identity testing for powerful skew circuits over $(\mathbb{Z}, +, \cdot)$ is polynomial time reducible to $\mathsf{CWP}(H)$.*



(a) Positive skew arithmetic Circuit $\mathcal{C}$

(b) SLP over G_a

(c) SLPs to generate B_1 and B_1^{-1}

Fig. 3: Transformation of a positive skew arithmetic circuit into an SLP over G_a.

One might try to extend Corollary 32 to certain infinite classes of polycyclic groups. Recently, Nikolaev and Ushakov [35] proved that the so called subset sum problem is NP-complete for every polycyclic group that is not virtually nilpotent. One might try to use their techniques to extend Corollary 32 to every polycyclic group that is not virtually nilpotent.

References

1. M. Agrawal and S. Biswas. Primality and identity testing via chinese remaindering. *Journal of the Association for Computing Machinery*, 50(4):429–443, 2003.
2. E. Allender, R. Beals, and M. Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8(2):99–126, 1999.
3. E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009.
4. E. Allender, J. Jiao, M. Mahajan, and V. Vinay. Non-commutative arithmetic circuits: Depth reduction and size lower bounds. *Theoretical Computer Science*, 209(1-2):47–86, 1998.
5. C. Àlvarez and B. Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107(1):3–30, 1993.
6. S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
7. V. Arvind and P. S. Joglekar. Arithmetic circuit size, identity testing, and finite automata. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:26, 2009.
8. L. Auslander. On a problem of Philip Hall. *Annals of Mathematics*, 86(2):112–116, 1967.
9. D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.
10. D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the Association for Computing Machinery*, 35(4):941–952, 1988.
11. M. Beaudry, P. McKenzie, P. Péladeau, and D. Thérien. Finite monoids: From word to circuit evaluation. *SIAM Journal on Computing*, 26(1):138–152, 1997.
12. M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992.
13. D. K. Biss and S. Dasgupta. A presentation for the unipotent group over rings with identity. *Journal of Algebra*, 237(2):691–707, 2001.
14. S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.
15. S. A. Cook and L. Fontes. Formal theories for linear algebra. *Logical Methods in Computer Science*, 8(1), 2012.
16. V. Diekert, A. G. Myasnikov, and A. Weiß. Conjugacy in Baumslag's group, generic case complexity, and division in power circuits. In *Proceedings of the 11th Symposium on Latin American Theoretical Informatics, LATIN 2014*, volume 8392 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2014.
17. W. Eberly. Very fast parallel polynomial arithmetic. *SIAM Journal on Computing*, 18(5):955–976, 1989.
18. W. Hesse, E. Allender, and D. A. M. Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65:695–716, 2002.
19. O. H. Ibarra and S. Moran. Probabilistic algorithms for deciding equivalence of straight-line programs. *Journal of the Association for Computing Machinery*, 30(1):217–228, 1983.
20. R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, STOC 1997*, pages 220–229. ACM Press, 1997.
21. V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.
22. M. I. Kargapolov and J. I. Merzljakov. *Fundamentals of the Theory of Groups*, volume 62 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1979.
23. O. G. Kharlampovič. A finitely presented solvable group with unsolvable word problem (Russian). *Izv. Akad. Nauk SSSR Ser. Mat. 45* no. 4, 852–873, 928, 1981.
24. D. E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithmus (third edition)*. Addison-Wesley, 1998.
25. D. König M. Lohrey. Parallel identity testing for algebraic branching programs with big powers and applications. In *Proceedings of Mathematical Foundations of Computer Science, MFCS 2015*, Lecture Notes in Computer Science 9235, pages 445–458 . Springer, 2015.
26. R. J. Lipton and Y. Zalcstein. Word problems solvable in logspace. *Journal of the Association for Computing Machinery*, 24(3):522–526, 1977.

27. M. Lohrey. Word problems and membership problems on compressed words. *SIAM Journal on Computing*, 35(5):1210 – 1240, 2006.
28. M. Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.
29. M. Lohrey. *The Compressed Word Problem for Groups*. SpringerBriefs in Mathematics. Springer, 2014.
30. M. Lohrey. Rational subsets of unitriangluar groups. *International Journal of Algebra and Computation*, 25(1-2):113-121, 2015.
31. A. I. Mal'cev. On certain classes of infinite soluble groups. *American Mathematical Society Translations, Series 2*, 2:1–21, 1956.
32. G. Miller. The commutator subgroup of a group generated by two operators. *Proceedings of the National Academy of Sciences of the United States of America*, 18:665–668, 1932.
33. C. Moore. Predicting nonlinear cellular automata quickly by decomposing them into linear ones. *Physica D: Nonlinear Phenomena*, 111:27–41, 1998.
34. A. G. Myasnikov and A. Weiß. TC^0 circuits for algorithmic problems in nilpotent groups. *CoRR*, abs/1702.06616, 2017.
35. A. Nikolaev and A. Ushakov. Subset sum problem in polycyclic groups. *CoRR*, abs/1703.07406, 2017.
36. D. Robinson. *Parallel Algorithms for Group Word Problems*. PhD thesis, University of California, San Diego, 1993.
37. J. J. Rotman. *An Introduction to the Theory of Groups (fourth edition)*. Springer, 1995.
38. W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1981.
39. H. U. Simon. Word problems for groups and contextfree recognition. In *Proceedings of Fundamentals of Computation Theory, FCT 1979*, pages 417–422. Akademie-Verlag, 1979.
40. R. Swan. Representations of polycyclic groups. *Proceedings of the American Mathematical Society*, 18:573–574, 1967.
41. J. Tits. Free subgroups in linear groups. *Journal of Algebra*, 20:250–270, 1972.
42. S. Toda. Counting problems computationally equivalent to computing the determinant. Technical Report CSIM 91-07, University of Electro-Communications, Tokyo, 1991.
43. S. D. Travers. The complexity of membership problems for circuits over sets of integers. *Theor. Comput. Sci.*, 369(1-3):211–229, 2006.
44. V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 270–284. IEEE Computer Society, 1991.
45. H. Vollmer. *Introduction to Circuit Complexity*. Springer, 1999.
46. S. Waack. On the parallel complexity of linear groups. *R.A.I.R.O. — Informatique Théorique et Applications*, 25(4):265–281, 1991.
47. B. A. F.Wehrfritz. *Infinite Linear Groups*. Springer, 1977.
48. A. Weiß. *On the Complexity of Conjugacy in Amalgamated Products and HNN Extensions*. PhD thesis, Universität Stuttgart, 2015.
49. O. Zariski and P. Samuel. *Commutative Algebra, Volume I*, volume 28 of *Graduate Texts in Mathematics*. Springer, 1958.