Parallel Identity Testing for Skew Circuits with Big Powers and Applications

Daniel König and Markus Lohrey

Department für Elektrotechnik und Informatik, Universität Siegen, Germany
{koenig,lohrey}@informatik.uni-siegen.de

Received (Day Month Year)
Accepted (Day Month Year)

Communicated by [editor]

Powerful skew arithmetic circuits are introduced. These are skew arithmetic circuits with variables, where input gates can be labeled with powers x^n for binary encoded numbers n. It is shown that polynomial identity testing for powerful skew arithmetic circuits belongs to coRNC^2, which generalizes a corresponding result for (standard) skew circuits. Two applications of this result are presented: (i) Equivalence of higher-dimensional straight-line programs can be tested in coRNC^2; this result is even new in the one-dimensional case, where the straight-line programs produce words. (ii) The compressed word problem (or circuit evaluation problem) for certain wreath products of finitely generated abelian groups belongs to coRNC^2. Using the Magnus embedding, it follows that the compressed word problem for a free metabelian group belongs to coRNC^2.

Keywords: Arithmetic circuits; polynomial identity testing; straight-line programs; wreath products.

Mathematics Subject Classification 2010: 68W30; 68W20; 20F10

1. Introduction

Polynomial identity testing is the following computational problem: The input is a circuit, whose internal gates are labeled with either addition or multiplication and its input gates are labeled 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 $\mathbb{Z}_n$ for $n \geq 2$). Based on the Schwartz-Zippel-DeMillo-Lipton Lemma, Ibarra and Moran [13] proved that polynomial identity testing over $\mathbb{Z}$ or $\mathbb{Z}_n$ belongs to the class coRP (the complements of problems in randomized polynomial time). Whether there is a deterministic polynomial time algorithm for polynomial identity testing is an important problem. In [14] 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 [16] proved that if polynomial identity testing belongs to P, then (i) there is a language in 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 problems in complexity theory. Hence, although it is quite plausible that polynomial identity testing belongs to P (by [14]), it will be probably very hard to prove (by [16]).

It is known that for algebraic formulas (where the circuit is a tree) and more generally, skew circuits (where for every multiplication gate, one of the two input gates is a constant or a variable), polynomial identity testing belongs to coRNC (but it is still not known to be in P), see [16, Corollary 2.1]. This holds, since algebraic formulas and skew circuits can be evaluated in NC if the variables are substituted by concrete (binary coded) numbers. Then, as for general polynomial identity testing, the Schwartz-Zippel-DeMillo-Lipton Lemma yields a coRNC-algorithm.

In this paper, we identify a larger class of circuits, for which polynomial identity testing is still in coRNC; we call these circuits *powerful skew circuits*. In such a circuit, we require that for every multiplication gate, one of the two input gates is either a constant or a power x^N of a variable x, where the exponent N is given in binary notation. One can replace this power x^N by a subcircuit of size $\log N$ using iterated squaring, but the resulting circuit is no longer skew. The main result of this paper states that polynomial identity testing for powerful skew circuits over the rings $\mathbb{Z}[x]$ and $\mathbb{F}_p[x]$ is still in coRNC (in fact, coRNC^2). For this, we use an identity testing algorithm of Agrawal and Biswas [1], which computes the output polynomial of the circuit modulo a polynomial $p(x)$ of polynomially bounded degree, which is randomly chosen from a certain sample space. Moreover, in our application, all computations can be done in the ring $\mathbb{F}_p[x]$ for a prime number p of polynomial size. This allows us to compute the big powers x^N modulo $p(x)$ in NC^2 using an algorithm of Fich and Tompa [9]. It should be noted that the application of the Agrawal-Biswas algorithm is crucial in our situation. If, instead we would use the Schwartz-Zippel-DeMillo-Lipton Lemma, then we would be forced to compute $a^N \mod m$ for randomly chosen numbers a and m with polynomially many bits. Whether this problem (modular powering) belongs to NC is a famous open problem [10, Problem B.5.6].

We present two applications of our coRNC identity testing algorithm. The first one concerns the equivalence problem for straight-line programs. Here, a straight-line program (SLP) is a context-free grammar G that computes a single word $\mathrm{val}(G)$. In this context, SLPs are extensively used in data compression and algorithmics on compressed data, see [20] for an overview. It is known that equivalence for SLPs, i.e., the question whether $\mathrm{val}(G) = \mathrm{val}(H)$ for two given SLPs, can be decided in polynomial time. This result was independently discovered by Hirshfeld, Jerrum, and Moller [12], Mehlhorn, Sundar, and Uhrig [24], and Plandowski [25]. Another algorithm was recently proposed by Jeż [15]. All known algorithms for the equivalence test are sequential and it is not clear how to parallelize them. Here, we exhibit an NC^2-reduction from the equivalence problem for SLPs to identity testing for powerful skew circuits. Hence, equivalence for SLPs belongs to coRNC. Moreover, our reduction immediately generalizes to higher dimensional pictures for which SLPs can be defined in a fashion similar to the one-dimensional (word) case, using one concatenation operation in each dimension. For two-dimensional SLPs, Berman et al. [5] proved that equivalence belongs to coRP using a reduction to PIT. We can improve this result to coRNC. Whether equivalence of two-dimensional (resp., one-dimensional) SLPs belongs to P (resp., NC) is open.

Our second application concerns the compressed word problem for groups. Let G be a finitely generated (f.g.) group, and let Σ be a finite generating set for G. For the compressed word problem for G, briefly $\mathsf{CWP}(G)$, the input is an SLP (as described in the preceding paragraph) over the alphabet $\Sigma \cup \Sigma^{-1}$, and it is asked whether $\mathrm{val}(G)$ evaluates to the group identity. The compressed word problem is a succinct version of the classical word problem (does a given word over $\Sigma \cup \Sigma^{-1}$ evaluate to the group identity?). One of the main motivations for the compressed word problem is the fact that the classical word problem for certain groups (automorphism groups, group extensions) can be reduced to the compressed word problem for simpler groups [21, Section 4.2]. For finite groups (and monoids) the compressed word problem was studied in Beaudry et al. [3], and for infinite groups the problem was studied for the first time in [19]. Subsequently, several important classes of f.g. groups with polynomial time compressed word problems were found: 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 [21] for details. For the important class of f.g. linear groups, i.e., f.g. groups of matrices over a field, one can show that the compressed word problem reduces to polynomial identity testing (over $\mathbb{Z}$ or $\mathbb{Z}_p$, depending on the characteristic of the field) and hence belongs to coRP [21, Theorem 4.15]. Vice versa, it was shown that polynomial identity testing over $\mathbb{Z}$ can be reduced to the compressed word problem for the linear group $\mathsf{SL}_3(\mathbb{Z})$ [21, Theorem 4.16]. The proof is based on a construction of Ben-Or and Cleve [4]. This result indicates that derandomizing the compressed word problem for a f.g. linear group will be in general very difficult.

In this paper, we consider the compressed word problem for wreath products. If G is a f.g. non-abelian group, then the compressed word problem for the wreath product $G \wr \mathbb{Z}$ is coNP-hard [21, Theorem 4.21]. On the other hand, we prove that $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z})$ is equivalent w.r.t. NC^2-reductions to identity testing for powerful skew circuits. In particular, $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z})$ belongs to coRNC. The latter result generalizes to any wreath product $G \wr H$, where $H = \mathbb{Z}^n$ for some n and G is a finite direct product of copies of $\mathbb{Z}$ and $\mathbb{Z}_p$ for primes p. Using the Magnus embedding theorem, it follows that the compressed word problem for a f.g. free metabelian group belongs to coRNC.

A short version of this paper appeared in [17].

2. Background from complexity theory

We assume some basic knowledge in complexity theory; see e.g. [2] for a detailed introduction. Recall that 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. Thus, if $L \in \mathsf{coRP}$ then there exists a polynomial time bounded randomized Turing machine R such that: (i) if $x \in A$ then R accepts x with probability 1, and (ii) if $x \notin A$ then R accepts x with probability at most $1/2$. The classes RP and coRP are easily seen to be closed under polynomial time many-one reductions.

We use standard definitions concerning circuit complexity, see e.g. [26] for more details. In particular we will consider the class NC^i of all problems that can be solved by a circuit family $(\mathcal{C}_n)_{n\geq 1}$, where the size of $\mathcal{C}_n$ (the circuit for length-n inputs) is polynomially bounded in n, its depth is bounded by $O(\log^i n)$, and $\mathcal{C}_n$ is built from input gates, NOT-gates and AND-gates and OR-gates of fan-in two. The class NC is the union of all classes NC^i. All circuit families in this paper will be logspace-uniform, which means that the mapping $a^n \mapsto \mathcal{C}_n$ can be computed in logspace. A few times, we will mention the class $\mathsf{DLOGTIME}$-uniform TC^0, see [11] for details. Here, it is only important that $\mathsf{DLOGTIME}$-uniform TC^0 is contained in NC^1. The most important circuit complexity class in this paper is NC^2. It is well-known that logspace computations can be carried out in NC^2.

To define a randomized version of NC^i, one uses circuit families with additional inputs. So, let the n^{th} circuit $\mathcal{C}_n$ in the family have n normal input gates plus m random input gates, where m is polynomially bounded in n. For an input $x \in \{0,1\}^n$ one defines the acceptance probability as

$$\mathsf{Prob}[\mathcal{C}_n \text{ accepts } x] = \frac{|\{y \in \{0,1\}^m \mid \mathcal{C}_n(x,y) = 1\}|}{2^m}$$

Here, $\mathcal{C}_n(x,y) = 1$ means that the circuit $\mathcal{C}_n$ evaluates to 1 if the i^{th} normal input gate gets the i^{th} bit of the input word x, and the i^{th} random input gate gets the i^{th} bit of the random word y. Then, the class RNC^i is the class of all problems A for which there exists a polynomial size circuit family $(\mathcal{C}_n)_{n\geq 0}$ of depth $O(\log^i n)$ with random input gates that uses NOT-gates and AND-gates and OR-gates of fan-in two, such that for all inputs $x \in \{0,1\}^*$ of length n: (i) if $x \in A$, then $\mathsf{Prob}[\mathcal{C}_n \text{ accepts } x] \geq 1/2$, and (ii) if $x \notin A$, then $\mathsf{Prob}[\mathcal{C}_n \text{ accepts } x] = 0$. As usual, coRNC^i is the class of all complements of problems from RNC^i. Section B.9 in [10] contains several problems that are known to be in RNC, but which are not known to be in NC; the most prominent example is the existence of a perfect matching in a graph.

3. Polynomials and circuits

Input representations of polynomials. In this paper we deal with polynomial rings $R[x_1, \ldots, x_k]$ in several variables, where R is the ring of integers $\mathbb{Z}$ or the ring

$\mathbb{Z}_n$ of integers modulo $n \geq 2$. For computational problems, we have to distinguish between two representations of polynomials. Let

$$p(x_1, \ldots, x_k) = \sum_{i=1}^{l} a_i x_1^{e_{i,1}} \cdots x_k^{e_{i,k}}$$

be a multivariate polynomial.

- The *standard representation* of $p(x)$ is the sequence of tuples $(a_i, e_{i,1}, \ldots, e_{i,k})$, where the coefficient a_i is represented in binary notation and the exponents $e_{i,j}$ are represented in unary notation. Let
$$|p| = \sum_{i=1}^{n} (\lceil \log |a_i| \rceil + e_{i,1} + \cdots + e_{i,k}).$$
- The *succinct representation* of $p(x)$ is the sequence of tuples $(a_i, e_{i,1}, \ldots, e_{i,k})$, where both the coefficient a_i and the exponents $e_{i,j}$ are represented in binary notation. Let
$$\|p\| = \sum_{i=1}^{n} (\lceil \log |a_i| \rceil + \lceil \log e_{i,1} \rceil + \cdots + \lceil \log e_{i,k} \rceil).$$

We use the following result of Eberly [8] (see also [11]).

Proposition 3.1. *Iterated addition, iterated multiplication, and division with remainder of polynomials from $\mathbb{Z}[x]$ or $\mathbb{F}_p[x]$ (p is a prime that can be part of the input in binary encoding) that are given in standard representation belong to* NC^1 *(in fact,* $\mathsf{DLOGTIME}$*-uniform* TC^0*).*

Circuits and branching programs over semirings. Consider a commutative semiring $\mathcal{S} = (S, \oplus, \otimes)$. An arithmetic circuit (or just circuit) over $\mathcal{S}$ is a triple $\mathcal{C} = (V, \mathsf{rhs}, S)$, where V is a finite set of *gates* or *variables*, $S \in V$ is the output gate, and rhs (for *right-hand side*) maps every $A \in V$ to an expression (the right-hand side of A) of one of the following three forms:

- a semiring element $s \in S$ (such a gate is an *input gate*),
- $B \oplus C$ with $B, C \in V$ (such a gate is an *addition gate*),
- $B \otimes C$ with $B, C \in V$ (such a gate is a *multiplication gate*).

Moreover, we require that the directed graph

$$\mathsf{graph}(\mathcal{C}) = (V, \{(A, B) \in V \times V \mid B \text{ occurs in } \mathsf{rhs}(A)\})$$

is acyclic. Every gate $A \in V$ evaluates to an element $\mathrm{val}_{\mathcal{C}}(A) \in S$ in the natural way and we set $\mathrm{val}(\mathcal{C}) = \mathrm{val}_{\mathcal{C}}(S)$. A circuit over $\mathcal{S}$ is called skew if for every multiplication gate A one of the two gates (or both of them) in $\mathsf{rhs}(A)$ is an input gate.

Circuits in the way we defined them are also known as straight-line programs. Often, it is convenient, to allow in right-hand side more complex arithmetic expressions, like for instance $(s \otimes B \otimes C) \oplus D$. By adding additional gates, we can split such right-hand sides into the basic forms $s \in S$, $B \oplus C$, and $B \otimes C$. In particular, we sometimes allow so-called *copy gates*, where the right-hand side is another gate. Such copy gates can be eliminated by a logspace computation, see [18, Lemma 1].

A branching program over $\mathcal{S}$ is a tuple $\mathcal{A} = (V, E, \lambda, s, t)$, where (V, E) is a directed acyclic graph, $\lambda : E \to S$ assigns to each edge a semiring element, and $s, t \in V$. Let $\mathcal{P}$ be the set of all paths from s to t. For a path $p = (v_0, v_1, \ldots, v_n) \in \mathcal{P}$ ($v_0 = s$, $v_n = t$) we define $\lambda(p) = \prod_{i=1}^{n} \lambda(v_{i-1}, v_i)$ as the product (w.r.t. $\otimes$) of all edge labels along the path. Finally, the value defined by $\mathcal{A}$ is

$$\mathrm{val}(\mathcal{A}) = \sum_{p \in \mathcal{P}} \lambda(p).$$

It is well known that skew circuits and branching programs are basically the same objects (note that $\mathcal{S}$ is assumed to be commutative).

NC^2-evaluation of branching programs. It is well known that the value defined by a branching program $\mathcal{A}$ can be computed using matrix powers. W.l.o.g. assume that $\mathcal{A} = (\{1, \ldots, n\}, E, \lambda, 1, n)$ and consider the adjacency matrix M of the edge-labeled graph $(\{1, \ldots, n\}, E, \lambda)$, i.e., the $(n \times n)$-matrix M with $M[i, j] = \lambda(i, j)$. Then

$$\mathrm{val}(\mathcal{A}) = \left(\sum_{i=0}^{n} M^i \right) [1, n].$$

For many semirings $\mathcal{S}$, this simple fact can be used to get an NC^2-algorithm for computing $\mathrm{val}(\mathcal{A})$. The $n + 1$ matrix powers M^i ($0 \le i \le n$) can be computed in parallel, and every power can be computed by a balanced tree of height $\log i \le \log n$, where every tree node computes a matrix product. Hence, we obtain an NC^2-algorithm, if

(i) the number of bits needed to represent a matrix entry in M^n is polynomially bounded in n and the number of bits of the entries in M, and
(ii) the product of two matrices over the semiring $\mathcal{S}$ can be computed in NC^1.

Point (ii) holds if products of two elements and iterated sums in $\mathcal{S}$ can be computed in NC^1. For the following important semirings these facts are well known (see also Proposition 3.1): $(\mathbb{Z}[x], +, \cdot)$, $(\mathbb{Z}_n[x], +, \cdot)$ for $n \ge 2$, $(\mathbb{Z} \cup \{\infty\}, \min, +)$, and $(\mathbb{Z} \cup \{-\infty\}, \max, +)$. Here, we assume that polynomials are given in the *standard representation*. For the polynomial ring $\mathbb{Z}[x]$ also note that every entry $p(x)$ of the matrix power M^n is a polynomial of degree $n \cdot m$, where m is the maximal degree of a polynomial in M, and all coefficients are bounded by $(n \cdot m \cdot a)^n$ (and hence need at most $n \cdot (\log n + \log m + \log a)$ bits), where a is the maximal absolute value of a coefficient in M. Hence point (i) above holds. The following lemma sums up the above discussion.

Lemma 3.2. *The output value of a given skew circuit (or branching program) over one of the following semirings can be computed in NC^2:*

(i) $(\mathbb{Z}[x], +, \cdot)$ *and* $(\mathbb{Z}_n[x], +, \cdot)$ *for* $n \geq 2$ *(polynomials are given in the standard representation, and* n *can be part of the input in binary representation)*

(ii) $(\mathbb{Z} \cup \{\infty\}, \min, +)$ *and* $(\mathbb{Z} \cup \{-\infty\}, \max, +)$ *(integers are given in binary representation)*

Actually, one can calculate the output value of the skew circuits in point (i) in the complexity class $\mathsf{DET} \subseteq \mathsf{NC}^2$ (see [7]), but in the following we only need the NC^2-bound. Point (i) of Lemma 3.2 also holds for the polynomial rings $(\mathbb{Z}[x_1, \ldots, x_k], +, \cdot)$ and $(\mathbb{Z}_n[x_1, \ldots, x_k], +, \cdot)$ as long as the number k of variables is not part of the input: The polynomial $p(x_1, \ldots, x_k) = \prod_{i=1}^{k}(x_i + 1)$ can be defined by a branching program with $O(k)$ edges labeled by the polynomials $x_i + 1$, but the product of these polynomials has 2^k monomials. Also note that it is important that we use the standard representation for polynomials in (i): The polynomial $p(x) = \prod_{i=1}^{n}(x^{2^i} + 1)$ can be represented by a branching program with $O(n)$ edges labeled by the polynomials $x^{2^i} + 1$ but $p(x)$ has 2^n monomials.

Powerful skew circuits. In this paper, we will mainly deal with circuits over a polynomial ring $R[x_1, \ldots, x_k]$, where the ring R is either $(\mathbb{Z}, +, \cdot)$ or $(\mathbb{Z}_n, +, \cdot)$. Let R be one of these rings. By definition, in such a circuit every input gate is labeled with a polynomial from $R[x_1, \ldots, x_k]$. Usually, one considers circuits where the right-hand side of an input gate is a polynomial given in standard representation (or, equivalently, a constant $a \in R$ or variable x_i); we will also use the term "standard circuits" in this case. For succinctness reasons, we will also consider circuits over $R[x_1, \ldots, x_k]$, where the right-hand sides of input gates are polynomials given in succinct representation. For general circuits this makes no real difference (since a power x^N can be defined by a subcircuit of size $O(\log N)$ using iterated squaring), but for skew circuits we will gain additional succinctness. We will use the term "powerful skew circuits". Formally, a *powerful skew circuit* over the polynomial ring $R[x_1, \ldots, x_k]$ is a skew circuit over the ring $R[x_1, \ldots, x_k]$ as defined above, where the right-hand side of every input gate is a polynomial that is given in succinct representation (equivalently, we could require that the right-hand side is a constant $a \in R$ or a power x_i^N with N given in binary notation). We define the size of a powerful skew circuit $\mathcal{C}$ as follows: First, define the size $\mathsf{size}_{\mathcal{C}}(A)$ of a gate $A \in V$ as follows: If A is an addition gate or a multiplication gate, then $\mathsf{size}_{\mathcal{C}}(A) = 1$, and if A is an input gate with $\mathsf{rhs}(A) = p(x_1, \ldots, x_k)$, then $\mathsf{size}_{\mathcal{C}}(A) = \|p(x_1, \ldots, x_k)\|$. Finally, we define the size of $\mathcal{C}$ as $\sum_{A \in V} \mathsf{size}_{\mathcal{C}}(A)$.

A *powerful branching program* is a branching program (V, E, λ, s, t) over a polynomial ring $R[x_1, \ldots, x_k]$, where every edge label $\lambda(e)$ $(e \in E)$ is a polynomial that is given in succinct representation. The size of a powerful branching program is $\sum_{e \in E} \|\lambda(e)\|$. From a given powerful skew circuit one can compute in logspace an equivalent powerful branching program and vice versa.

Note that the transformation of a powerful skew circuit over $R[x_1, \ldots, x_k]$ into an equivalent standard skew circuit (where every input gate is labeled by a polynomial given in standard representation) requires an exponential blow-up. For instance, the smallest standard skew circuit for the polynomial x^N has size N, whereas x^N can be trivially obtained by a powerful skew circuit of size $\lceil \log N \rceil$.

Polynomial identity testing. A central computational problem in computational algebra is *polynomial identity testing*, briefly PIT. Let R be a ring that is effective in the sense that elements of R can be encoded by natural numbers in such a way that addition and multiplication in R become computable operations. Then, PIT for the ring R is the following problem:

Input: A number $k \geq 1$ and a circuit $\mathcal{C}$ over the ring $R[x_1, \ldots, x_k]$.
Question: Is $\mathsf{val}(\mathcal{C})$ the zero-polynomial?

For the rings $\mathbb{Z}$ and $\mathbb{Z}_p$ (p prime) the following result was shown in [13]; for $\mathbb{Z}_n$ with n composite, it was shown in [1].

Theorem 3.3. *For each of the rings $\mathbb{Z}$ and $\mathbb{Z}_n$ ($n \geq 2$), PIT belongs to the class* coRP.

Note that the number k of variables is part of the input in PIT. On the other hand, there is a well-known reduction from PIT to PIT restricted to univariate polynomials (polynomials with a single variable) [1]. For a multivariate polynomial $p(x_1, \ldots, x_k) \in R[x_1, \ldots, x_k]$ let $\deg_i(p)$ be the degree of p in the variable x_i. It is the largest number d such that x_i^d appears in a monomial of p. Let $p(x_1, \ldots, x_k)$ be a polynomial and let $d = 1 + \max\{\deg_i(p) \mid 1 \leq i \leq k\}$. We define the univariate polynomial $U(p)$ as

$$U(p) = p(y^1, y^d, \ldots, y^{d^{k-1}}).$$

Hence, the polynomial $U(p)$ is obtained from $p(x_1, \ldots, x_k)$ by replacing every monomial $a \cdot x_1^{n_1} \cdots x_1^{n_k}$ by $a \cdot y^N$, where $N = n_1 + n_2 d + \cdots n_k d^{k-1}$ is the number with base-d representation $(n_1, n_2, \ldots, n_k)$. The polynomial p is the zero-polynomial if and only if $U(p)$ is the zero-polynomial.

The following lemma can be also shown for arbitrary circuits, but we will only need it for powerful skew circuits.

Lemma 3.4. *Given a powerful skew circuit $\mathcal{C}$ for the polynomial $p(x_1, \ldots, x_k)$, the following can be be computed in* NC^2*:*

(i) The binary encoding of $d = 1 + \max\{\deg_i(p) \mid 1 \leq i \leq k\}$ and
(ii) a powerful skew circuit $\mathcal{C}'$ for $U(p)$.

Proof. Let $\mathcal{C}$ be a powerful skew circuit for the polynomial $p(x_1, \ldots, x_k)$. In order to compute $\deg_i(p)$, we construct a circuit over the max-plus semiring as follows: Take the circuit $\mathcal{C}$. If A is an input gate that is labeled with the polynomial $a(x_1, \ldots, x_k)$, then relabel A with the binary coded number $\deg_i(a)$. Moreover, for a gate A with $\mathsf{rhs}(A) = B + C$ (resp., $\mathsf{rhs}(A) = B \times C$) we set $\mathsf{rhs}(A) = \max(B, C)$ (resp.,

$\mathsf{rhs}(A) = B + C$). The resulting circuit is clearly skew. Therefore it can be evaluated in NC^2 by Lemma 3.2.

Once the number $d = 1 + \max\{\deg_i(p) \mid 1 \leq i \leq k\}$ is computed we simply replace every monomial $a \cdot x_1^{n_1} \cdots x_k^{n_k}$ in the circuit $\mathcal{C}$ by the monomial $a \cdot y^N$, where $N = n_1 + n_2 d + \cdots n_k d^{k-1}$. The binary encoding of N can be computed from the binary encodings of $n_1, \ldots, n_k$ even in $\mathsf{DLOGTIME}$-uniform TC^0. □

Note that the above reduction from multivariate to univariate circuits does not work for standard skew circuits: the output circuit will be powerful skew even if the input circuit is standard skew. For instance, the polynomial $\prod_{i=1}^{k} x_i$ (which can be produced by a standard skew circuit of size k) is transformed into the polynomial y^{2^k-1}, for which the smallest standard skew circuit has size $\Omega(2^k)$.

4. PIT for powerful skew circuits

Our main result for powerful skew circuits is:

Theorem 4.1. *For each of the rings $\mathbb{Z}$ and $\mathbb{F}_p$ (p is a prime that can be part of the input in unary encoding), PIT for powerful skew circuits belongs to the class coRNC^2.*

The proof of Theorem 4.1 has two main ingredients: The randomized identity testing algorithm of Agrawal and Biswas [1] and the modular polynomial powering algorithm of Fich and Tompa [9]. Let us start with the identity testing algorithm of Agrawal and Biswas. We will only need the version for the polynomial ring $\mathbb{F}_p[x]$, where p is a prime number.

Consider a polynomial $P(x) \in \mathbb{F}_p[x]$ of degree d. The algorithm of Agrawal and Biswas consists of the following steps (later we will apply this algorithm to the polynomial defined by a powerful skew circuit), where $0 < \epsilon < 1$ is an error parameter:

(1) Let ℓ be a number with $\ell \geq \log d$ and $t = \max\{\ell, \frac{1}{\epsilon}\}$
(2) Find the smallest prime number r such that $r \neq p$ and r does not divide any of $p-1$, $p^2-1, \ldots, p^{\ell-1}-1$. It is argued in [1] that $r \in O(\ell^2 \log p)$.
(3) Randomly choose a tuple $b = (b_0, \ldots, b_{\ell-1}) \in \{0,1\}^\ell$ and compute the polynomial $T_{r,b,t}(x) = Q_r(A_{b,t}(x))$, where $Q_r(x) = \sum_{i=0}^{r-1} x^i$ is the r^{th} cyclotomic polynomial and $A_{b,t}(x) = x^t + \sum_{i=0}^{\ell-1} b_i \cdot x^i$.
(4) Accept, if $P(x) \bmod T_{r,b,t}(x) = 0$, otherwise reject.

Clearly, if $P(x) = 0$, then the above algorithm accepts with probability 1. For a non-zero polynomial $P(x)$, Agrawal and Biswas proved:

Theorem 4.2 ([1]). *Let $P(x) \in \mathbb{F}_p[x]$ be a non-zero polynomial of degree d. The above algorithm rejects $P(x)$ with probability at least $1 - \varepsilon$.*

The second result we are using was shown by Fich and Tompa:

Theorem 4.3 ([9]). *The following computation can be done in* NC^2*:*

Input: A unary encoded prime number p*, polynomials* $a(x), q(x) \in \mathbb{F}_p[x]$ *such that* $\deg(a(x)) < \deg(q(x)) = d$*, and a binary encoded number* N*.*

Output: The polynomial $a(x)^N$ *mod* $q(x)$*.*

Remark 4.4. In [9], it is stated that the problem can be solved using circuits of depth $(\log n)^2 \log\log n$ for the more general case that the underlying field is $\mathbb{F}_{p^\ell}$, where p and ℓ are given in unary representation. The main bottleneck is the computation of an iterated matrix product $A_1 A_2 \cdots A_m$ (for m polynomial in the input length) of $(d \times d)$-matrices over the field $\mathbb{F}_{p^\ell}$. In our situation (where the field is $\mathbb{F}_p$) we easily obtain an NC^2-algorithm for this step: Two $(d \times d)$-matrices over $\mathbb{F}_p$ can be multiplied in NC^1 (actually in $\mathsf{DLOGTIME}$-uniform TC^0). Then we compute the product $A_1 A_2 \cdots A_m$ by a balanced binary tree of depth $\log m$. Also logspace-uniformity of the circuits is not stated explicitly in [9], but follows easily since only standard arithmetical operations on binary coded numbers are used.

Now we can prove Theorem 4.1:

Proof of Theorem 4.1. By Lemma 3.4 we can restrict to univariate polynomials. We first prove the theorem for the case of a powerful skew circuit $\mathcal{C}$ over the field $\mathbb{F}_p$, where the prime number p is part of the input but specified in unary notation.

Let p be a unary encoded prime number and $\mathcal{A} = (\{1, \ldots, n\}, 1, n, \lambda)$ be a powerful branching program with n nodes that is equivalent to $\mathcal{C}$. We can assume that every edge label $\lambda(e)$ is either an element of $\mathbb{F}_p$ or a power x^N, where N is given in binary representation. Let $P(x) = \mathsf{val}(\mathcal{A}) \in \mathbb{F}_p[x]$. Fix an error probability $0 < \varepsilon < 1$. Our randomized NC^2-algorithm is based on the identity testing algorithm of Agrawal and Biswas. It accepts with probability 1 if $\mathsf{val}(\mathcal{A}) = 0$ and accepts with probability at most ϵ if $P(x) \neq 0$. Let us go through the four steps of the Agrawal-Biswas algorithm to see that they can be implemented in NC^2.

Step 1. An upper bound on the degree of $P(x)$ can be computed in NC^2 as in the proof of Lemma 3.4. For the number ℓ we can take the number of bits of this degree bound, which is a polynomial bound in the input size. Let $t = \max\{\ell, \frac{1}{\epsilon}\}$.

Step 2. For the prime number r we know that $r \in O(\ell^2 \log p)$, which is a polynomial bound. Hence, we can test in parallel all possible candidates for r. For a certain candidate r, we check in parallel whether it is prime (recall that r is of polynomial value and of logarithmic bit length) and whether it divides any of the numbers $p-1$, $p^2-1, \ldots, p^{\ell-1}-1$. The whole computation is possible in NC^1.

Step 3. Let $b = (b_0, \ldots, b_{\ell-1}) \in \{0,1\}^\ell$ be the chosen tuple. We have to compute the polynomial $T_{r,b,t}(x) = Q_r(A_{b,t}(x))$, where $Q_r(x) = \sum_{i=0}^{r-1} x^i$ and $A_{b,t} = x^t + \sum_{i=0}^{\ell-1} b_i \cdot x^i$. This is an instance of iterated multiplication (for the powers $A_{b,t}(x)^i$) and iterated addition of polynomials. Hence, by Proposition 3.1 also this step can be carried out in NC^1. Note that the degree of $T_{r,b,t}(x)$ is $t \cdot (r-1)$, i.e., polynomial in the input size.

Step 4. For the last step, we have to compute $P(x)$ mod $T_{r,b,t}(x)$. For this, we consider in parallel all edge labels x^N in our powerful branching program $\mathcal{A}$. Recall that N is given in binary notation. Using the Fich-Tompa algorithm we compute x^N mod $T_{r,b,t}(x)$ (with $a(x) = x$) in NC^2. We then replace the edge label x^N by x^N mod $T_{r,b,t}(x)$. Let $\mathcal{B}$ be the resulting branching program. Every polynomial that appears as an edge label in $\mathcal{B}$ is now given in standard form. Hence, by Lemma 3.2 we can compute in NC^2 the output polynomial $\mathsf{val}(\mathcal{B})$. Clearly, $P(x)$ mod $T_{r,b,t}(x) = \mathsf{val}(\mathcal{B})$ mod $T_{r,b,t}(x)$. The latter polynomial can be computed in NC^1 by Proposition 3.1.

Let us now prove Theorem 4.1 for the ring $\mathbb{Z}$. Let $\mathcal{A} = (\{1,\ldots,n\},1,n,\lambda)$ be a powerful branching program over $\mathbb{Z}$ with n nodes and let $P(x) = \mathrm{val}(\mathcal{A})$. Let us first look at the coefficients of $P(x)$. Let m be the maximum absolute value of some edge label $a \in \mathbb{Z}$ in $\mathcal{A}$. Since there are at most n^n paths from s to t in $\mathcal{A}$, every coefficient of the polynomial $P(x)$ belongs to the interval $[-(mn)^n,(mn)^n]$. Let $k = n \cdot (\lceil \log m \rceil + \lceil \log n \rceil) + 1$ and $p_1,\ldots,p_k$ be the first k prime numbers. Each prime p_i is polynomially bounded in k (and hence the input size) and the list of these primes can be computed in NC^1 by doing in parallel all necessary divisibility checks on unary encoded numbers.

The Chinese remainder theorem implies that $P(x) = 0$ if and only if $P(x) \equiv 0$ mod p_i for all $1 \le i \le k$ (since $\prod_{i=1}^k p_i > 2 \cdot (mn)^n$). We can carry out the latter tests in parallel using the above algorithm for a unary encoded prime number. The overall algorithm accepts if we accept for every prime p_i. If $P(x) = 0$, then we will accept for every $1 \le i \le k$ with probability 1, hence the overall algorithm accepts with probability 1. On the other hand, if $P(x) \neq 0$, then there exists a prime p_i $(1 \le i \le k)$ such that the algorithm rejects with probability at least $1 - \varepsilon$. Hence, the overall algorithm will reject with probability at least $1 - \varepsilon$ as well. $\square$

5. Multi-dimensional pictures

Let Γ be a finite alphabet. For $l \in \mathbb{N}$ let $[l] = \{1,\ldots,l\}$. An *n-dimensional picture* over Γ is a mapping $p : \prod_{j=1}^n [l_j] \to \Gamma$ for some $l_j \in \mathbb{N}$, where $\prod$ denotes the Carteasian product. Let $\mathrm{dom}(p) = \prod_{j=1}^n [l_j]$. For $1 \le j \le n$ we define $|p|_j = l_j$ as the length of p in the j^{th} dimension. Note that one-dimensional pictures are simply finite words. Let Γ_n^* denote the set of n-dimensional pictures over Γ. On this set we can define partially defined concatenation operations $\circ_i$ $(1 \le i \le n)$ as follows: For pictures $p, q \in \Gamma_n^*$, the picture $p \circ_i q$ is defined if and only if $|p|_j = |q|_j$ for all $1 \le j \le n$ with $i \neq j$. In this case, we have $|p \circ_i q|_j = |p|_j$ $(= |q|_j)$ for $j \neq i$ and $|p \circ_i q|_i = |p|_i + |q|_i$. Let $l_j = |p \circ_i q|_j$. For a tuple $(k_1,\ldots,k_n) \in \prod_{j=1}^n [l_j]$ we finally set $(p \circ_i q)(k_1,\ldots,k_n) = p(k_1,\ldots,k_n)$ if $k_i \le |p|_i$ and $(p \circ_i q)(k_1,\ldots,k_n) = q(k_1,\ldots,k_{i-1},k_i - |p|_i,k_{i+1},\ldots,k_n)$ if $k_i > |p|_i$. These operations generalize the concatenation of finite words.

Example 5.1. Here is an example of the concatenation of two 2-dimensional pictures, which are drawn as rectangular arrays. Dimension one (resp., two) is the horizontal (resp., vertical) dimension. Notice that in contrast to matrices, the vertical coordinate in a picture increase from bottom to top in the following diagrams.

$$\left(\boxed{\begin{array}{ccc}1&0&0\\0&1&1\end{array}} \circ_2 \boxed{\begin{array}{ccc}1&0&1\end{array}}\right) \circ_1 \left(\boxed{\begin{array}{cc}1&0\\0&1\end{array}} \circ_2 \boxed{\begin{array}{cc}1&1\end{array}}\right)$$

$$= \boxed{\begin{array}{ccc}1&0&1\\1&0&0\\0&1&1\end{array}} \circ_1 \boxed{\begin{array}{cc}1&1\\1&0\\0&1\end{array}} = \boxed{\begin{array}{ccccc}1&0&1&1&1\\1&0&0&1&0\\0&1&1&0&1\end{array}}$$

5.1. *Multi-dimensional straight-line programs*

We now present a succinct representation of multi-dimensional pictures, that allows to compress pictures having repetitive subpatterns. An *n-dimensional straight-line program (SLP)* over the terminal alphabet Γ is a triple $\mathbb{A} = (V, \mathsf{rhs}, S)$, where V is a finite set of variables, $S \in V$ is the start variable, and rhs maps each variable A to its right-hand side $\mathsf{rhs}(A)$, which is either a terminal symbol $a \in \Gamma$ or an expression of the form $B \circ_i C$, where $B, C \in V$ and $1 \leq i \leq n$ such that the following additional conditions are satisfied:

- The relation $\{(A, B) \in V \times V \mid B \text{ occurs in } \mathsf{rhs}(A)\}$ is acyclic.
- One can assign to each $A \in V$ and $1 \leq i \leq n$ a number $|A|_i$ with the following properties: If $\mathsf{rhs}(A) \in \Gamma$ then $|A|_i = 1$ for all i. If $\mathsf{rhs}(A) = B \circ_i C$ then $|A|_i = |B|_i + |C|_i$ and $|A|_j = |B|_j = |C_j|$ for all $j \neq i$.

These conditions ensure that every variable A evaluates to a unique n-dimensional picture $\mathrm{val}_{\mathbb{A}}(A)$ such that $|\mathrm{val}_{\mathbb{A}}(A)|_i = |A|_i$ for all $1 \leq i \leq n$. Finally, $\mathrm{val}(\mathbb{A}) = \mathrm{val}_{\mathbb{A}}(S)$ is the picture defined by $\mathbb{A}$. We omit the index $\mathbb{A}$ if the underlying SLP is clear from the context. We define the size of the SLP $\mathbb{A} = (V, \Gamma, S, P)$ as $|\mathbb{A}| = |V|$.

A one-dimensional SLP is a context-free grammar that generates a single word. Two-dimensional SLPs were studied in [5].

For all dimensions i it is straightforward to define an SLP $\mathbb{A}$ of size m such that $|\mathrm{val}(\mathbb{A})|_i = 2^m$. Hence, an SLP can be seen as a compressed representation of the picture it generates, and an exponential compression ratio can be achieved in this way.

5.2. *Equality testing for compressed words and n-dimensional pictures*

Given two n-dimensional SLPs we want to know whether they evaluate to the same picture. In [5] it was shown that this problem belongs to coRP by translating it to polynomial identity testing. For a given n-dimensional picture $p : \mathrm{dom}(p) \to \{0, 1\}$

we define the polynomial

$$f_p(x_1, \ldots, x_n) = \sum_{(e_1,\ldots,e_n) \in \mathrm{dom}(p)} p(e_1, \ldots, e_n) \prod_{i=1}^{n} x_i^{e_i - 1}.$$

We consider f_p as a polynomial from $\mathbb{Z}_2[x_1, \ldots, x_n]$. For two n-dimensional pictures p and q such that $|p|_i = |q|_i$ for all $1 \le i \le n$ we clearly have $p = q$ if and only if $f_p + f_q = 0$ (recall that coefficients are from $\mathbb{Z}_2$). In [5], it was observed that from an SLP $\mathbb{A}$ for a picture P, one can easily construct an arithmetic circuit for the polynomial f_p, which leads to a coRP-algorithm for equality testing. Since the circuit for f_p is actually powerful skew, we get:

Theorem 5.2. *The question whether two n-dimensional SLPs $\mathcal{A}$ and $\mathcal{B}$ evaluate to the same n-dimensional picture is in* coRNC^2 *(here, n is part of the input).*

Proof. Let $\mathbb{A}_1 = (V_1, \mathsf{rhs}_1, S_1)$ and $\mathbb{A}_2 = (V_2, \mathsf{rhs}_2, S_2)$ be n-dimensional SLPs over the alphabet Γ. We can assume that $V_1 \cap V_2 = \emptyset$ and $\Gamma = \{0, 1\}$ (if $\Gamma = \{a_1, \ldots, a_k\}$ then we encode a_i by $0^i 1^{k-i}$).

First we calculate $|A|_i$ for every $1 \le i \le n$ and every $A \in V_1 \cup V_2$ in NC^2 by evaluating additive circuits over $\mathbb{N}$, see Lemma 3.2. If $|S_1|_i \ne |S_2|_i$ for some $1 \le i \le n$, then we have $\mathrm{val}(\mathbb{A}_1) \ne \mathrm{val}(\mathbb{A}_2)$. Otherwise, we construct the circuit

$$\mathcal{C} = (V_1 \cup V_2 \cup \{S\}, \mathsf{rhs}, S)$$

over $\mathbb{Z}_2[x_1, \ldots, x_n]$ with:

- $\mathsf{rhs}_{\mathcal{C}}(A) = B + x_k^{|B|_k} \cdot C$ if $A \in V_1$ and $\mathsf{rhs}_1(A) = B \circ_k C$ or $A \in V_2$ and $\mathsf{rhs}_2(A) = B \circ_k C$ (note that $V_1 \cap V_2 = \emptyset$),
- $\mathsf{rhs}_{\mathcal{C}}(A) = a$ if $\mathsf{rhs}_1(A) = a \in \{0, 1\}$ or $\mathsf{rhs}_2(A) = a \in \{0, 1\}$, and
- $\mathsf{rhs}(S) = S_1 + S_2$.

Then $\mathrm{val}(\mathcal{C}) = f_{\mathrm{val}(\mathbb{A}_1)} + f_{\mathrm{val}(\mathbb{A}_2)}$. Since we are working over the coefficient field $\mathbb{Z}_2$, we have $\mathrm{val}(\mathcal{C}) = 0$ if and only if $\mathrm{val}(\mathbb{A}_1) = \mathrm{val}(\mathbb{A}_2)$. Obviously $\mathcal{C}$ becomes a powerful skew circuit after splitting right-hand sides of the form $B + x_k^N \cdot C$. Hence, Theorem 4.1 implies that $\mathrm{val}(\mathcal{C}) = 0$ can be checked in coRNC^2. $\square$

It should be noted that even in the one-dimensional case (where equality testing for SLPs can be done in polynomial time [12,15,24,25]), no randomized NC-algorithm was known before.

Example 5.3. We consider the 2-dimensional SLP $\mathbb{A} = (V, \mathsf{rhs}, S)$ with

$$V = \{S, C, D, X_0, X_1\} \cup \{A_i, B_i \mid 0 \le i \le n-1\}$$

and the following right-hand side mapping:

- $\mathsf{rhs}(S) = A_{n-1} \circ_2 A_{n-1}$,
- $\mathsf{rhs}(A_i) = A_{i-1} \circ_2 A_{i-1}$ for $1 \le i \le n-1$,

- $\mathsf{rhs}(A_0) = B_{n-1} \circ_1 B_{n-1}$,
- $\mathsf{rhs}(B_i) = B_{i-1} \circ_1 B_{i-1}$ for $1 \leq j \leq n-1$,
- $\mathsf{rhs}(B_0) = C \circ_1 D$,
- $\mathsf{rhs}(C) = X_0 \circ_2 X_1$,
- $\mathsf{rhs}(D) = X_1 \circ_2 X_0$,
- $\mathsf{rhs}(X_0) = 0$.
- $\mathsf{rhs}(X_1) = 1$.

The picture $p = \mathrm{val}(\mathbb{A})$ is the $(2^{n+1} \times 2^{n+1})$-picture with $p(i,j) = (i+j) \bmod 2$ for all $i, j \in [1, 2^{n+1}]$. For $n = 2$ we obtain the following (8×8)-picture:

1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1

The horizontal and vertical dimensions of the pictures defined by the variables are the following:

- $|S|_1 = |S|_2 = 2^{n+1}$,
- $|A_i|_1 = 2^{n+1}$, $|A_i|_2 = 2^{i+1}$ for $0 \leq i \leq n-1$,
- $|B_i|_1 = 2^{i+1}$, $|B_i|_2 = 2$ for $0 \leq i \leq n-1$,
- $|C|_1 = |D|_1 = 1$, $|C|_2 = |D|_2 = 2$,
- $|X_0|_1 = |X_0|_2 = |X_1|_1 = |X_1|_2 = 1$.

A powerful skew circuit for this polynomial is obtained by the following right-hand side mapping:

- $\mathsf{rhs}(S) = A_{n-1} + x_2^{2^n} A_{n-1}$,
- $\mathsf{rhs}(A_i) = A_{i-1} + x_2^{2^i} A_{i-1}$ for $1 \leq i \leq n-1$,
- $\mathsf{rhs}(A_0) = B_{n-1} + x_1^{2^n} B_{n-1}$,
- $\mathsf{rhs}(B_i) = B_{i-1} x_1^{2^i} B_{i-1}$ for $1 \leq i \leq n-1$,
- $\mathsf{rhs}(B_0) = C + x_1 D$,
- $\mathsf{rhs}(C) = X_0 + x_2 X_1$,
- $\mathsf{rhs}(D) = X_1 + x_2 X_0$,
- $\mathsf{rhs}(X_0) = 0$,
- $\mathsf{rhs}(X_1) = 1$.

6. Circuits over wreath products

As a second application of identity testing for powerful skew circuits we will consider the circuit evaluation problem (also known as the compressed word problem) for wreath products of finitely generated abelian groups. The wreath product is an important operation in group theory. The next subsection briefly recalls the definition and some well-known results. We assume some basic familiarity with group theory.

6.1. *Wreath products*

Let G and H be groups. The restricted wreath product $H \wr G$ is defined as follows:

- Elements of $H \wr G$ are pairs (f, g), where $g \in G$ and $f : G \to H$ is a mapping such that $f(a) \neq 1_H$ for only finitely many $a \in G$ (1_H is the identity element of H).
- The multiplication in $H \wr G$ is defined as follows: Let $(f_1, g_1), (f_2, g_2) \in H \wr G$. Then $(f_1, g_1)(f_2, g_2) = (f, g_1 g_2)$, where $f(a) = f_1(a) f_2(g_1^{-1} a)$ for all $a \in G$.

For readers, who have not seen this definition before, the following intuition might be helpful: An element $(f, g) \in H \wr G$ can be thought as a finite collection of elements of H that are sitting in certain elements of G (the mapping f) together with a distinguished element of G (the element g), which can be thought as a cursor moving around G. If we want to compute the product $(f_1, g_1)(f_2, g_2)$, we do this as follows: First, we shift the finite collection of H-elements that corresponds to the mapping f_2 by g_1: If the element $h \in H \setminus \{1_H\}$ is sitting in $a \in G$ (i.e., $f_2(a) = h$), then we remove h from a and put it to the new location $g_1 a \in G$. This new collection corresponds to the mapping $f_2' : a \mapsto f_2(g_1^{-1} a)$. After this shift, we multiply the two collections of H-elements pointwise: If in $a \in G$ the elements h_1 and h_2 are sitting (i.e., $f_1(a) = h_1$ and $f_2'(a) = h_2$), then we put the product $h_1 h_2$ into the G-location a. Finally, the new distinguished G-element (the new cursor position) becomes $g_1 g_2$.

Example 6.1. The wreath product $\mathbb{Z} \wr \mathbb{Z}$ is generated by $\{a, t, a^{-1}, t^{-1}\}$, where $a = (f_0, 1)$, $t = (f_1, 0)$, $a^{-1} = (f_0, -1)$, and $t^{-1} = (f_{-1}, 0)$. Here, $f_0 : \mathbb{Z} \to \mathbb{Z}$ maps every integer to zero, and $f_1(0) = 1$, $f_{-1}(0) = -1$, and $f_1(z) = f_{-1}(z) = 0$ for $z \in \mathbb{Z} \setminus \{0\}$. An element of $\mathbb{Z} \wr \mathbb{Z}$ can be viewed as an assignment of integers to the positions of a two-way infinite line whose positions are identified with the integers. Only finitely many positions carry a non-zero integer. Moreover, the cursor is sitting on a certain position on the line. Multiplying with a (resp., a^{-1}) on the right moves the cursor to the right (resp., left). Multiplying with t (resp., t^{-1}) increments (resp., decrements) the integer at the current cursor position. At the beginning 0 assigned to every position and the cursor is at position 0. In this setting, the word $w = ataatata^{-1}t^{-1}$ evaluates to the element in Figure 1(e), where the position of the curser is marked grey.

The following lemma seems to be folklore.

Lemma 6.2. *The group $(A \times B) \wr G$ embeds into $(A \wr G) \times (B \wr G)$.*

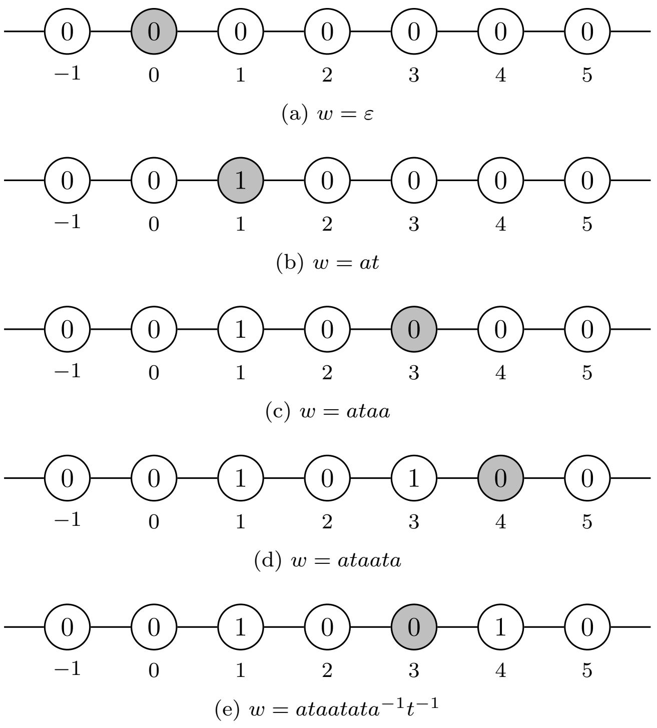

Fig. 1: The element in $\mathbb{Z} \wr \mathbb{Z}$ corresponding to $w = ataatata^{-1}t^{-1}$.

Proof. Let $\pi_A : A \times B \to A$ be the natural projection morphism and similarly for $\pi_B : A \times B \to B$. We define an embedding $\varphi : (A \times B) \wr G \to (A \wr G) \times (B \wr G)$ by

$$\varphi(f, g) = \Big((f \circ \pi_A, g), (f \circ \pi_B, g)\Big).$$

Clearly, φ is injective. Moreover, φ is a group homomorphism. □

A proof of the following simple lemma can be found for instance in [22].

Lemma 6.3. *Let K be a subgroup of H of finite index m and let G be a group. Then $G^m \wr K$ is isomorphic to a subgroup of index m in $G \wr H$.*

6.2. *Compressed word problems*

Let G be a finitely generated (f.g.) group and let Σ be a finite generating set for G, i.e., every element of G can be written as a finite product of elements from Σ and inverses of elements from Σ. Let $\Gamma = \Sigma \cup \{a^{-1} \mid a \in \Sigma\}$. For a word $w \in \Gamma^*$ we write $w = 1$ in G if and only if the word w evaluates to the identity of G. The *word problem* for G asks, whether $w = 1$ in G for a given input word. There exist finitely generated groups and in fact finitely presented groups (groups that are defined by finitely many defining relations) with an undecidable word problem. Here, we are interested in the *compressed word problem* for a finitely generated group. In this problem, the input word w is given in a compressed form by a one-dimensional SLP as defined in Section 5. Recall that a one-dimensional picture over an alphabet Γ is simply a finite word over Γ. Hence, $\mathrm{val}(\mathbb{A})$ is a word if $\mathbb{A}$ is a one-dimensional SLP. In the following we always mean one-dimensional SLPs when using the term SLP. A right-hand side $A \circ_1 B$ of such an SLP is simply written as AB in the following. The compressed word problem for G asks, whether $\mathrm{val}(\mathbb{A}) = 1$ in G for a given SLP $\mathbb{A}$. We use the abbreviation $\mathsf{CWP}(G)$ for the compressed word problem for G.

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 [21] for more details. Groups, for which the compressed word problem can be solved in polynomial time are [21]: 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 [3] 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 6.4 ([3]). *Let G be a finite group. If G is solvable, then $\mathsf{CWP}(G)$ belongs to the class NC^2. If G is not solvable, then $\mathsf{CWP}(G)$ is P-complete.*

The following two results are proven in [21].

Theorem 6.5 ([21, 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 6.5 is shown in [21]:

Theorem 6.6 ([21, Theorem 4.16]). *The problem $\mathsf{CWP}(\mathsf{SL}_3(\mathbb{Z}))$ 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 [4] for simulating arithmetic circuits by matrix products.

Finally, the following result was recently shown in [18]; it generalizes the NC^2-statement from Theorem 6.4.

Theorem 6.7 ([18]). *Let G be a f.g. group having a normal subgroup H such that H is f.g. nilpotent and the quotient group G/H is finite solvable. Then $\mathsf{CWP}(G) \in \mathsf{NC}^2$.*

To the knowledge of the authors, there is no example of a group G not having the properties from Theorem 6.7, for which $\mathsf{CWP}(G)$ belongs to NC.

In the rest of the paper, we study the compressed word problem for wreath products. For the case that the left factor of a wreath product is not abelian, the following hardness result was shown in [21]:

Theorem 6.8 ([21, Theorem 4.21]). *If G is a f.g. non-abelian group, then $\mathsf{CWP}(G \wr \mathbb{Z})$ is coNP-hard.*

Because of this result we will only consider wreath products $G \wr H$ with G f.g. abelian.

6.3. $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z})$ *and identity testing for powerful skew circuits*

In this section, we explore the relationship between the compressed word problem for the wreath product $\mathbb{Z} \wr \mathbb{Z}$ and polynomial identity testing for powerful skew circuits. We show that these two problems are equivalent w.r.t. NC^2-reductions.

Let $G = \mathbb{Z} \wr \mathbb{Z}$. We consider the generators a and t of G, where t (resp., a) generates the $\mathbb{Z}$-copy on the left (resp., right). So, with a (resp., a^{-1}) we move the cursor to the left (resp., right) and with t (resp., t^{-1}) we add one (resp., subtract one) from the value at the current cursor position. Let $\Gamma = \{a, t, a^{-1}, t^{-1}\}$.

For a word $w \in \Gamma^*$ we define $\Delta(w) = |w|_a - |w|_{a^{-1}} \in \mathbb{Z}$. The word w is *positive* if $\Delta(u) \geq 0$ for every prefix u of w that ends with t or t^{-1}. The word w is *well-formed*, if it is positive and $\Delta(w) = 0$. If w is positive and $(f, g) \in G$ is the group element represented by the word w, then $f(x) \neq 0$ implies that $x \in \mathbb{N}$ (intuitively, the $\mathbb{Z}$-generator t or its inverse is never added to a position outside of $\mathbb{N}$). If in addition w is well-formed then $g = 0$. For a given positive word $w \in \Gamma^*$ we define a polynomial $p_w(x) \in \mathbb{Z}[x]$ inductively as follows:

- $p_\varepsilon(x) = 0$.
- If $w = ua$ or $w = ua^{-1}$, then $p_w(x) = p_u(x)$.
- If $w = ut^\delta$ with $\delta \in \{1, -1\}$, then $p_w(x) = p_u(x) + \delta \cdot x^d$, where $d = \Delta(w) = \Delta(u)$.

If the positive word w represents the group element $(f, g) \in G$, then

$$p_w(x) = \sum_{e \geq 0} f(e) \cdot x^e.$$

In particular, the following equivalence holds for every positive word $w \in \Gamma^*$:

$$w = 1 \text{ in } G \quad \Leftrightarrow \quad (p_w(x) = 0 \text{ and } \Delta(w) = 0)$$

Lemma 6.9. *From a given SLP $\mathbb{A}$ over the alphabet Γ one can compute in logspace an SLP $\mathbb{A}'$ over the alphabet Γ such that (i) $\mathrm{val}(\mathbb{A}) = 1$ in G if and only if $\mathrm{val}(\mathbb{A}') = 1$ in G and (ii) for every variable A of $\mathbb{A}'$ the word $\mathrm{val}_{\mathbb{A}'}(A)$ is positive.*

Proof. Let $\mathbb{A} = (V, \mathsf{rhs}_{\mathbb{A}}, S)$ and let $k = |\mathrm{val}(\mathbb{A})|$. We can assume that V contains two unique variables T and T^{-1} such that $\mathsf{rhs}_{\mathbb{C}}(T) = t$ and $\mathsf{rhs}_{\mathbb{A}}(T^{-1}) = t^{-1}$. From the SLP $\mathbb{A}$ we can compute in logspace new SLPs $\mathbb{B}$ and $\mathbb{B}^{-1}$ such that $\mathrm{val}(\mathbb{B}) = a^k$ and $\mathrm{val}(\mathbb{B}^{-1}) = a^{-k}$. For this we replace in $\mathbb{A}$ all letters in Γ by the letter a (resp., a^{-1}). Let B be the start variable of $\mathbb{B}$ and let B^{-1} be the start variable of $\mathbb{B}^{-1}$. We now construct a third SLP $\mathbb{C}$ with the variable set $\{X, X^{-1}, Y, Y^{-1}\} \uplus V$, where $\uplus$ denotes the disjoint union. We define $\mathsf{rhs}_{\mathbb{C}}(Y) = BT$, $\mathsf{rhs}_{\mathbb{C}}(Y^{-1}) = BT^{-1}$, $\mathsf{rhs}_{\mathbb{C}}(X) = YB^{-1}$, $\mathsf{rhs}_{\mathbb{C}}(X^{-1}) = Y^{-1}B^{-1}$. For all variables $A \in V$ we obtain $\mathsf{rhs}_{\mathbb{C}}(A)$ from $\mathsf{rhs}_{\mathbb{A}}(A)$ by replacing every occurrence of the variable T^{δ} by X^{δ}. We then define $\mathbb{A}'$ as the disjoint union of the SLPs $\mathbb{B}$, $\mathbb{B}^{-1}$ and $\mathbb{C}$. Then, $\mathrm{val}_{\mathbb{A}'}(A) = a^k\, \mathrm{val}_{\mathbb{A}}(A)\, a^{-k}$ for every variable A of $\mathbb{A}$. $\square$

Lemma 6.10. *From a given SLP $\mathbb{A}$ over the alphabet Γ such that $\mathrm{val}_{\mathbb{A}}(A)$ is positive for every variable A of $\mathbb{A}$ one can compute in NC^2 a powerful skew circuit $\mathcal{C}$ such that $\mathsf{val}(\mathcal{C}) = p_{\mathrm{val}(\mathbb{A})}(x)$. In particular, $\mathrm{val}(\mathbb{A}) = 1$ in G if and only if ($\mathsf{val}(\mathcal{C}) = 0$ and $\Delta(\mathrm{val}(\mathbb{A})) = 0$).*

Proof. Let $\mathbb{A} = (V, \mathsf{rhs}_{\mathbb{A}}, S)$ for the further consideration. For every $A \in V$, the word $\mathrm{val}_{\mathbb{A}}(A)$ is positive. Hence, for every $A \in V$ we can define the polynomial $p_A(x) := p_{\mathrm{val}(A)}(x)$. Moreover, let

$$d(A) = \Delta(\mathrm{val}(A)) \in \mathbb{Z} \text{ and}$$
$$m(A) = \min(\{\Delta(u) \mid u \text{ is a prefix of } \mathrm{val}(A) \text{ that ends with } t \text{ or } t^{-1}\}),$$

where we set $\min(\emptyset) = \infty$. Since $\mathrm{val}(A)$ is positive, we have $m(A) \geq 0$. The numbers $d(A)$ can be computed by an additive circuit in NC^2, see Lemma 3.2. The numbers $m(A)$ can be computed in NC^2 as well, using the following identity:

$$m(A) = \begin{cases} \infty & \text{if } \mathsf{rhs}_{\mathbb{A}}(A) \in \{a, a^{-1}\}, \\ 0 & \text{if } \mathsf{rhs}_{\mathbb{A}}(A) \in \{t, t^{-1}\}, \\ \min\{m(B), d(B) + m(C)\} & \text{if } \mathsf{rhs}_{\mathbb{A}}(A) = BC. \end{cases}$$

Note that these rules define a skew circuit in the semiring $(\mathbb{Z} \cup \{\infty\}, \min, +)$ with binary encoded input values. Hence, by Lemma 3.2 the circuit can be evaluated in NC^2.

In case $m(A) < \infty$, we can write the polynomial $p_A(x)$ uniquely as

$$p_A(x) = x^{m(A)} \cdot q_A(x),$$

for a polynomial $q_A(x)$. Note that $q_A(x)$ can be a multiple of x due to cancellation of monomials in $p_A(x)$. For instance, if $\mathrm{val}(A) = att^{-1}at$ then $p_A(x) = x^2$, $m(A) = 1$, and $q_A(x) = x$.

We now construct a circuit $\mathcal{D}$ such that for every $A \in V$ we have:

$$\mathsf{val}_{\mathcal{D}}(A) = q_A(x).$$

We define the rules of the circuit $\mathcal{D}$ as follows, where A is a variable with $m(A) < \infty$:

- If $\mathsf{rhs}_{\mathbb{A}}(A) = t^{\delta}$ for $\delta \in \{-1, 1\}$, then we set $\mathsf{rhs}_{\mathcal{D}}(A) = \delta$.
- If $\mathsf{rhs}_{\mathbb{A}}(A) = BC$ and $m(C) = \infty$, then we set $\mathsf{rhs}_{\mathcal{D}}(A) = B$.
- If $\mathsf{rhs}_{\mathbb{A}}(A) = BC$ and $m(B) = \infty$, then we set $\mathsf{rhs}_{\mathcal{D}}(A) = C$.
- If $\mathsf{rhs}_{\mathbb{A}}(A) = BC$ and $m(B) < \infty$, $m(C) < \infty$, then we have $m(A) = \min\{m(B), d(B) + m(C)\}$ and we set $\mathsf{rhs}_{\mathcal{D}}(A) = (M_B \times B) + (M_C \times C)$, where

$$M_B = \begin{cases} 1 & \text{if } m(B) \leq d(B) + m(C) \\ x^{m(B)-d(B)-m(C)} & \text{if } m(B) > d(B) + m(C) \end{cases}$$

$$M_C = \begin{cases} 1 & \text{if } m(B) \geq d(B) + m(C) \\ x^{d(B)+m(C)-m(B)} & \text{if } m(B) < d(B) + m(C). \end{cases}$$

Note that the circuit $\mathcal{D}$ is powerful skew. The final circuit $\mathcal{C}$ can be easily obtained by multiplying the output gate of $\mathcal{D}$ with $x^{m(S)}$ (if $m(S) = \infty$ we set the output gate to zero). □

From Lemma 6.9 and 6.10 we get:

Lemma 6.11. *The compressed word problem for $\mathbb{Z} \wr \mathbb{Z}$ is NC^2-reducible to PIT for powerful skew circuits over the ring $\mathbb{Z}[x]$.*

Example 6.12. Consider the SLP with the following right-hand side mapping:

- $\mathsf{rhs}(A_0) = a$, $\mathsf{rhs}(B_0) = a^{-1}$,
- $\mathsf{rhs}(A_i) = A_{i-1}A_{i-1}$, $\mathsf{rhs}(B_i) = B_{i-1}B_{i-1}$ for $1 \leq i \leq 4$
- $\mathsf{rhs}(T) = t$,
- $\mathsf{rhs}(A) = A_4T$
- $\mathsf{rhs}(B) = AB_2$
- $\mathsf{rhs}(C) = BT$
- $\mathsf{rhs}(S) = CC$

All words $\mathrm{val}(X)$ for the above nonterminals X are positive, and we have

- $p_{A_i}(x) = 0$, $d(A_i) = 2^i$, $m(A_i) = \infty$ for $0 \leq i \leq 4$,
- $p_{B_i}(x) = 0$, $d(B_i) = -2^i$, $m(B_i) = \infty$ for $0 \leq i \leq 4$,
- $p_T(x) = 1$, $d(T) = m(T) = 0$,
- $p_A(x) = x^{16}$, $d(A) = m(A) = 16$,
- $p_B(x) = x^{16}$, $d(B) = 16 - 4 = 12$, $m(B) = 16$,
- $p_C(x) = x^{12} + x^{16}$, $d(C) = m(C) = 12$,
- $p_S(x) = x^{12} + x^{16} + x^{24} + x^{28}$, $d(S) = 28$, $m(S) = 12$.

The polynomials $q_X(s)$ for $X \in \{T, A, B, C, S\}$ are

- $q_T(x) = q_A(x) = q_B(x) = 1$,
- $q_C(x) = 1 + x^4$,
- $q_S(x) = 1 + x^4 + x^{12} + x^{16}$.

A powerful skew circuit for these polynomials is obtained by the construction from the proof of Lemma 6.10:

- $\mathsf{rhs}(T) = 1$,
- $\mathsf{rhs}(A) = T$
- $\mathsf{rhs}(B) = A$
- $\mathsf{rhs}(C) = x^4 \times B + 1 \times T$
- $\mathsf{rhs}(S) = 1 \times C + x^{12} \times C$

In the rest of this section we show that PIT for powerful skew circuits can be reduced in NC^2 to $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z})$. By Proposition 3.4, it suffices to consider the univariate case. The following two lemmas are obvious.

Lemma 6.13. *Let $u, v \in \Gamma^*$ be well-formed. Then $w = uv$ is well-formed too and $p_w(x) = p_u(x) + p_v(x)$.*

Lemma 6.14. *Let $u \in \Gamma^*$ be well-formed, $n, m \in \mathbb{N}$ and let $w = a^n u^m a^{-n}$. Then w is well-formed too and $p_w(x) = m \cdot x^n \cdot p_u(x)$.*

Lemma 6.15. *From a given powerful skew circuit $\mathcal{C}$ over the ring $\mathbb{Z}[x]$, one can compute in NC^2 an SLP $\mathbb{A}$ over the alphabet Γ such that the following holds:*

- *$\mathrm{val}(\mathbb{A})$ is well-formed and*
- *$p_{\mathrm{val}(\mathbb{A})}(x) = \mathsf{val}(\mathcal{C})$.*

Proof. Let $\mathcal{C} = (V, \mathsf{rhs}_{\mathcal{C}}, S)$. The set of variables of our SLP $\mathbb{A}$ contains V, a disjoint copy $V' = \{A' \mid A \in V\}$ of V, plus some auxiliary variables. The start variable is S. For every variable $A \in V$ we will have $p_{\mathrm{val}_{\mathbb{A}}(A)}(x) = \mathsf{val}_{\mathcal{C}}(A)$ and for every variable $A' \in V'$ we will have $p_{\mathrm{val}_{\mathbb{A}}(A')}(x) = -\mathsf{val}_{\mathcal{C}}(A)$. We now define the right-hand sides of the $\mathbb{A}$. Thereby we allow powers of the form α^n (for α a letter from Γ or a variable) in right-hand sides. Such powers can be produced using auxillary variables and iterated squaring.

- If $\mathsf{rhs}_{\mathcal{C}}(A) = b{\cdot}x^n$, then we set $\mathsf{rhs}_{\mathbb{A}}(A) = a^n t^b a^{-n}$ and $\mathsf{rhs}_{\mathbb{A}}(A') = a^n t^{-b} a^{-n}$.
- If $\mathsf{rhs}_{\mathcal{C}}(A) = B + C$, then we set $\mathsf{rhs}_{\mathbb{A}}(A) = BC$ and $\mathsf{rhs}_{\mathbb{A}}(A') = B'C'$. The correctness of this step follows from Lemma 6.13.
- If $\mathsf{rhs}_{\mathcal{C}}(A) = B \times C$, where w.l.o.g. C is an input gate with $\mathsf{rhs}_{\mathcal{C}}(C) = b \cdot x^n$, then we set $\mathsf{rhs}_{\mathbb{A}}(A) = a^n B^b a^{-n}$ and $\mathsf{rhs}_{\mathbb{A}}(A') = a^n B^{-b} a^{-n}$, where we set $B^{-c} = (B')^c$ for $c \geq 1$. The correctness of this step follows from Lemma 6.14.

It follows by a straightforward induction that for every $A \in V$, the word $\mathrm{val}_{\mathbb{A}}(A)$ and $\mathrm{val}_{\mathbb{A}}(A')$ are well-formed. □

Example 6.16. Consider the powerful skew circuit

$$\mathcal{C} = (\{A, B, C, D, E, S\}, \mathsf{rhs}_{\mathcal{C}}, S)$$

with the following right-hand side mapping:

- $\mathsf{rhs}_{\mathcal{C}}(A) = -3 \cdot x^{100}$,
- $\mathsf{rhs}_{\mathcal{C}}(B) = 5 \cdot x^{200}$,
- $\mathsf{rhs}_{\mathcal{C}}(C) = A + B$,
- $\mathsf{rhs}_{\mathcal{C}}(D) = C \times A$,
- $\mathsf{rhs}_{\mathcal{C}}(E) = D + D$,
- $\mathsf{rhs}_{\mathcal{C}}(S) = E \times B$.

The construction from the proof of Lemma 6.15 yields the following SLP $\mathbb{A}$ (where those variables that are not needed in order to produce $\mathrm{val}(\mathbb{A})$ are not shown):

- $\mathsf{rhs}_{\mathbb{A}}(A') = a^{100}t^3a^{-100}$,
- $\mathsf{rhs}_{\mathbb{A}}(B') = a^{100}t^{-5}a^{-100}$,
- $\mathsf{rhs}_{\mathbb{A}}(C') = A'B'$,
- $\mathsf{rhs}_{\mathbb{A}}(D) = a^{100}(C')^3a^{-100}$,
- $\mathsf{rhs}_{\mathbb{A}}(E) = DD$,
- $\mathsf{rhs}_{\mathbb{A}}(S) = a^{200}E^5a^{-200}$.

From Lemma 6.11 and 6.15 we directly obtain:

Corollary 6.17. *The compressed word problem for $\mathbb{Z} \wr \mathbb{Z}$ is equivalent w.r.t. NC^2-reductions to PIT for powerful skew circuits over the ring $\mathbb{Z}[x]$.*

In exactly the same way we can show:

Corollary 6.18. *The compressed word problem for $\mathbb{Z}_n \wr \mathbb{Z}$ ($n \geq 2$) is equivalent w.r.t. NC^2-reductions to PIT for powerful skew circuits over the ring $\mathbb{Z}_n[x]$.*

Let us mention that in [18] it is shown that there exists a polycyclic group G (of Hirsch length 3) such that PIT for powerful skew circuits over the ring $\mathbb{Z}[x]$ is polynomial time reducible to the compressed word problem for G.

6.4. *Compressed word problems in* coRNC^2

In this section, we apply the results from the last section to find groups for which the compressed word problem belongs to coRNC^2. Recall from Section 6.2 that the only known examples of groups with a word problem in NC are groups G having a normal subgroup H such that (i) H is f.g. nilpotent and (ii) G/H is finite solvable. For wreath products we use the following lemma:

Lemma 6.19. *For every $k \geq 1$ and every finitely generated group G, $\mathsf{CWP}(G \wr \mathbb{Z}^k)$ is NC^2-reducible to $\mathsf{CWP}(G \wr \mathbb{Z})$.*

Proof. The idea is similar to the proof of Proposition 3.4. Let G be generated by the finite set Σ. Fix the generating set $\{a_1, a_2, \ldots, a_k\}$ for $\mathbb{Z}^k$, where every a_i generates a $\mathbb{Z}$-copy. Then $G \wr \mathbb{Z}^k$ is generated by the set $\Gamma = \Sigma \cup \{a_1, a_2, \ldots, a_k\}$.

Let $\mathbb{A}$ be an SLP over the alphabet $\Gamma \cup \Gamma^{-1}$. First, we compute in NC^2 the number $d = 2(|\mathrm{val}(\mathbb{A})| + 1)$. Note that for all $a_i, b_i \in \mathbb{Z}$ $(1 \le i \le k)$ with $|a_i|, |b_i| \le |\mathrm{val}(\mathbb{A})|$ we have: $(a_1, \ldots, a_k) = (b_1, \ldots, b_k)$ if and only if $\sum_{i=1}^{k} a_i \cdot d^{i-1} = \sum_{i=1}^{k} b_i \cdot d^{i-1}$.

From our SLP $\mathbb{A}$ we construct a new SLP $\mathbb{B}$ by replacing every occurrence of a_i (resp., a_i^{-1}) in a right-hand side by a new variable that produces $a^{d^{i-1}}$ (resp., $a^{-d^{i-1}}$). This implies the following: If $(f, (z_1, \ldots, z_k))$ (resp., (h, z)) is the group element of $\mathsf{CWP}(G \wr \mathbb{Z}^k)$ (resp., $\mathsf{CWP}(G \wr \mathbb{Z})$) represented by $\mathrm{val}(\mathbb{A})$ (resp., $\mathrm{val}(\mathbb{B})$), then $z = \sum_{i=1}^{k} z_i \cdot d^{i-1}$ and for all $(x_1, \ldots, x_k) \in \mathbb{Z}^k$, $f(x_1, \ldots, x_k) = h(x)$, where $x = \sum_{i=1}^{k} x_i \cdot d^{i-1}$. It follows that $\mathrm{val}(\mathbb{A}) = 1$ in $G \wr \mathbb{Z}^k$ if and only if $\mathrm{val}(\mathbb{B}) = 1$ in $G \wr \mathbb{Z}$. □

Corollary 6.20. *Let G be a finite direct product of copies of $\mathbb{Z}$ and $\mathbb{Z}_p$ for primes p. Then, for every $n \ge 1$, $\mathsf{CWP}(G \wr \mathbb{Z}^n)$ belongs to coRNC^2.*

Proof. Assume that $G = \prod_{i=1}^{k} G_i$ where for every $1 \le i \le k$, either $G_i = \mathbb{Z}$ or $G_i = \mathbb{Z}_{p_i}$ for a prime p_i. By Lemma 6.2, $G \wr \mathbb{Z}^n$ is a subgroup of $\prod_{i=1}^{k} G_i \wr \mathbb{Z}^n$, which implies that the compressed word problem for $G \wr \mathbb{Z}^n$ is logspace-reducible to the compressed word problem for $\prod_{i=1}^{k} G_i \wr \mathbb{Z}^n$ [21, Proposition 4.3]. Hence, from an SLP $\mathbb{A}$ over the generators of $G \wr \mathbb{Z}^n$ we can compute in logspace (and hence NC^2) SLPs $\mathbb{A}_1, \ldots, \mathbb{A}_k$, where $\mathbb{A}_i$ is an SLP over the generators of $G_i \wr \mathbb{Z}^n$ such that $\mathrm{val}(\mathbb{A}) = 1$ in $G \wr \mathbb{Z}^n$ if and only if for all $1 \le i \le k$, $\mathrm{val}(\mathbb{A}_i) = 1$ in $G_i \wr \mathbb{Z}^n$. Moreover, from the SLP $\mathbb{A}_i$ we can compute by Lemma 6.19 in NC^2 an SLP $\mathbb{B}_i$ over the generators of $G_i \wr \mathbb{Z}$ such that $\mathrm{val}(\mathbb{A}_i) = 1$ in $G_i \wr \mathbb{Z}^n$ if and only if $\mathrm{val}(\mathbb{B}_i) = 1$ in $G_i \wr \mathbb{Z}$ (the SLPs $\mathbb{B}_i$ can be computed in parallel). By Theorem 4.1 and Corollary 6.17 and 6.18 the compressed word problem for every group $G_i \wr \mathbb{Z}$ belongs to coRNC^2. The corresponding circuits are put in parallel with independent random bits. The overall circuit accepts if and only if $\mathrm{val}(\mathbb{B}_i) = 1$ in $G_i \wr \mathbb{Z}$ for all $1 \le i \le k$. The error probability of the resulting circuit is bounded by the maximal error probability of the circuits for the groups $G_i \wr \mathbb{Z}$, $1 \le i \le k$; see also the argument at the end of the proof of Theorem 4.1. □

It is not clear, whether in Corollary 6.20 we can replace G by an arbitrary finitely generated abelian group. On the other hand, if we apply Theorem 3.3 instead of Theorem 4.1 we obtain:

Corollary 6.21. *Let G be f.g. abelian and let H be f.g. virtually abelian (i.e., H has a f.g. abelian subgroup of finite index). Then $\mathsf{CWP}(G \wr H)$ belongs to coRP.*

Proof. Let $K \le H$ be a f.g. abelian subgroup of finite index m in H. Moreover, either $K = 1$ or $K \cong \mathbb{Z}^k$ for some $k \ge 1$. By Lemma 6.3, $G^m \wr K$ is isomorphic to a subgroup of index m in $G \wr H$. If the group A is a finite index subgroup of the group B, then $\mathsf{CWP}(B)$ is polynomial-time many-one reducible to $\mathsf{CWP}(A)$ [21, Theorem 4.4]. Hence, it suffices to show that $\mathsf{CWP}(G^m \wr K)$ belongs to coRP. Since G^m is finitely generated abelian, it suffices to show that $\mathsf{CWP}(\mathbb{Z}_n \wr K)$ $(n \ge 2)$

and $\mathsf{CWP}(\mathbb{Z} \wr K)$ belong to coRP; the argument is the same as in the proof of Corollary 6.20. The case $K = 1$ is clear. So, assume that $K \cong \mathbb{Z}^k$ for some $k \geq 1$. By Corollary 6.20, $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z}^k)$ belongs to coRNC. Moreover, by Lemma 6.19, Theorem 3.3 and Corollary 6.18, $\mathsf{CWP}(\mathbb{Z}_n \wr \mathbb{Z}^k)$ belongs to coRP. □

Let us conclude the paper with an application of Corollary 6.20 to free metabelian groups. Recall that a group is metabelian if its commutator subgroup is abelian. It is known that every f.g. metabelian group can be embedded into a finite direct product of f.g. linear groups [27]. From this, one can deduce that the compressed word problem of a finitely generated metabelian group belongs to coRP. For f.g. *free* metabelian groups, we can sharpen this fact:

Theorem 6.22. *For every f.g. free metabelian group the compressed word problem belongs to* coRNC^2.

Proof. By the Magnus embedding theorem [23] the free metabelian group of rank r can be embedded into the wreath product $\mathbb{Z}^r \wr \mathbb{Z}^r$. The result follows from Corollary 6.20. □

7. Open problems

Our coRNC^2 identity testing algorithm for powerful skew circuits only works for the coefficient rings $\mathbb{Z}$ and $\mathbb{Z}_p$ with p prime. It is not clear how to extend it to $\mathbb{Z}_n$ with n composite. The Agrawal-Biswas identity testing algorithm also works for $\mathbb{Z}_n$ with n composite. But the problem is that the Fich-Tompa algorithm only works for polynomial rings over $\mathbb{Z}_p$ with p prime. For equality testing for multi-dimensional straight-line programs it remains open whether a polynomial time algorithm exists. For the one-dimensional (word) case, a polynomial time algorithm exists. Here, it remains open, whether the equality problem is in NC.

Acknowledgments

This work was supported by the DFG research project LO 748/12-1.

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] S. Arora and B.Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.
[3] 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.
[4] M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992.
[5] P. Berman, M. Karpinski, L. L. Larmore, W. Plandowski, and W. Rytter. On the complexity of pattern matching for highly compressed two-dimensional texts. *Journal of Computer and System Sciences*, 65(2):332–350, 2002.

[6] A. Chiu, G. Davida, and B. Litow. Division in logspace-uniform NC^1. *Theoretical Informatics and Applications. Informatique Théorique et Applications*, 35(3):259–275, 2001.

[7] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[8] W. Eberly. Very fast parallel polynomial arithmetic. *SIAM Journal on Computing*, 18(5):955–976, 1989.

[9] F. E. Fich and M. Tompa. The parallel complexity of exponentiating polynomials over finite fields. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 38–47. ACM, 1985.

[10] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, 1995.

[11] 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.

[12] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science*, 158(1&2):143–159, 1996.

[13] 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.

[14] 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.

[15] A. Jeż. Faster fully compressed pattern matching by recompression. *ACM Transactions on Algorithms*, 11(3):20:1–20:43, 2015.

[16] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[17] D. König and M. Lohrey. Parallel identity testing for skew circuits with big powers and applications. In *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science 2015, MFCS 2015, Part II*, volume 9235 of *Lecture Notes in Computer Science*, pages 445–458. Springer, 2015.

[18] D. König and M. Lohrey. Evaluation of circuits over nilpotent and polycyclic groups. *Algorithmica*, 80(5):1459–1492, 2018.

[19] M. Lohrey. Word problems and membership problems on compressed words. *SIAM Journal on Computing*, 35(5):1210 – 1240, 2006.

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

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

[22] M. Lohrey, B. Steinberg, and G. Zetzsche. Rational subsets and submonoids of wreath products. *Information and Computation*, 243:191–204, 2015.

[23] W. Magnus. On a theorem of Marshall Hall. *Annals of Mathematics. Second Series*, 40:764–768, 1939.

[24] K. Mehlhorn, R. Sundar, and C. Uhrig. Maintaining dynamic sequences under equality tests in polylogarithmic time. *Algorithmica*, 17(2):183–198, 1997.

[25] W. Plandowski. Testing equivalence of morphisms on context-free languages. In *Proceedings of the 2nd Annual European Symposium on Algorithms, ESA 1994*, volume 855 of *Lecture Notes in Computer Science*, pages 460–470. Springer, 1994.

[26] H. Vollmer. *Introduction to Circuit Complexity.* Springer, 1999.

[27] B. A. Wehrfritz. On finitely generated soluble linear groups. *Mathematische Zeitschrift*, 170:155–167, 1980.