Parallel Identity Testing for Skew Circuits with Big Powers and Applications

Daniel König and Markus Lohrey

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

Abstract. Powerful skew arithmetic circuits are introduced. These are skew arithmetic circuits with variables, where input gates can be labelled 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 strings. (ii) The compressed word problem (or circuit evaluation problem) for certain wreath products belongs to coRNC^2. Full proofs can be found in the long version [13].

1 Introduction

Polynomial identity testing (PIT) is the following computational problem: The input is an arithmetic circuit, built up from addition gates, multiplication gates, and input gates that 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 $\mathbb{Z}_n$ for $n \geq 2$). Based on the Schwartz-Zippel-DeMillo-Lipton Lemma, Ibarra and Moran [9] proved that PIT 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 PIT is an important problem. In [10] 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 [11] have shown that if PIT $\in \mathsf{P}$, 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 problems in complexity theory. This indicates that derandomizing PIT will be very difficult. On the other hand, for arithmetic formulas (where the circuit is a tree) and skew arithmetic circuits (where for every multiplication gate, one of the two input gates is a constant or a variable), PIT belongs to coRNC (but it is still not known to be in P), see [11, Cor. 2.1]. This holds, since arithmetic formulas and skew arithmetic circuits can be evaluated in NC if the variables are substituted by concrete (binary coded) numbers. Then, as for general PIT, the Schwartz-Zippel-DeMillo-Lipton Lemma yields a coRNC-algorithm.

In this paper, we identify a larger class of arithmetic 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 PIT for powerful skew circuits over the rings $\mathbb{Z}[x]$ and $\mathbb{Z}_p[x]$ (p prime) 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 $q(x)$ 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 $q(x)$ in NC^2 using an algorithm of Fich and Tompa [5]. It should be noted that the application of the Agrawal-Biswas algorithm is crucial. If, instead we would use the Schwartz-Zippel-DeMillo-Lipton Lemma, then we would be forced to compute $a^N \bmod m$ for randomly chosen numbers a and m with polynomially many bits. Whether this problem (modular powering) belongs to NC is an open problem [6, 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 [15] for an overview. Equivalence for SLPs, i.e., the question whether $\mathrm{val}(G) = \mathrm{val}(H)$ for given SLPs G, H, can be decided in polynomial time. This result was independently discovered in [8,19,21]. All known algorithms for SLP-equivalence are sequential and it is not clear how to parallelize them. Here, we exhibit an NC^2-reduction from SLP-equivalence to PIT for skew powerful 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 (string) case, using one concatenation operation in each dimension. For two-dimensional SLPs, Berman et al. [3] proved that equivalence belongs to coRP using a reduction to PIT. We 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 defined 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 [16, Section 4.2]. For finite groups (and monoids) the compressed word problem was studied in Beaudry et al. [2], and for infinite groups the problem was studied for the first time in [14]. Subsequently, several important classes of f.g. groups with polynomial time compressed word problems were found: f.g. nilpotent groups, graph groups (also known as right-angled Artin groups), and virtually special groups. The latter include all Coxeter groups, one-relator groups with torsion, fully residually free groups, and fundamental groups of hyperbolic 3-manifolds; see [16]. For f.g. linear groups, i.e., f.g. groups of matrices over a field, the compressed word problem reduces to PIT (over $\mathbb{Z}$ or $\mathbb{Z}_p$, depending on the characteristic of the field) and hence belongs to coRP [16, Thm. 4.15]. Recently it was shown in [12] that the compressed word problem for a f.g. nilpotent group belongs to NC^2 (the result can be extended to so called f.g. nilpotent by finite solvable groups). Here, we prove that the compressed word problem for the wreath product $\mathbb{Z} \wr \mathbb{Z}$ is equivalent w.r.t. NC^2-reductions to PIT for powerful skew circuits. In particular, $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z})$ belongs to coRNC. This result generalizes to 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. In contrast, it was shown in [16, Thm. 4.21] that $\mathsf{CWP}(G \wr \mathbb{Z})$ is coNP-hard for every non-abelian group G.

2 Background from complexity theory

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.

We use standard definitions concerning circuit complexity, see e.g. [22] for more details. In particular we will consider the class NC^i of all problems that can be solved by a polynomial size circuit family of depth $O(\log^i n)$ that uses 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 n-th circuit in the family can be computed in logspace from the unary encoding of n.

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 $x \in \{0,1\}^n$, its acceptance probability is $\mathsf{Prob}[\mathcal{C}_n \text{ accepts } x] = 2^{-m} \cdot |\{y \in \{0,1\}^m \mid \mathcal{C}_n(x,y) = 1\}|$. 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 string x, and the i-th random input gate gets the i-th bit of the random string 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 [6] contains several problems that are known to be in RNC, but not known to be in NC; the most prominent example is the existence of a perfect matching in a graph.

3 Polynomials and circuits

We deal with multivariate polynomial rings $R[x_1, \ldots, x_k]$, 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 distinguish between two representations of polynomials. Consider the polynomial

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

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 (of course this is only important for the coefficient ring $\mathbb{Z}$) 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 $p(x)$ coincides with the standard version, except that 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 [4] (see also [7]).

Proposition 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.

Consider a commutative semiring $\mathcal{S} = (S, \oplus, \otimes)$. An arithmetic circuit (or just circuit) over $\mathcal{S}$ is a triple $\mathcal{C} = (V, \mathsf{rhs}, A_0)$, where V is a finite set of *gates* or *variables*, $A_0 \in V$ 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: (i) a semiring element $s \in S$ (such a gate is an *input gate*), (ii) $B \oplus C$ with $B, C \in V$ (such a gate is an *addition gate*), (iii) $B \otimes C$ with $B, C \in V$ (such a gate is a *multiplication gate*). Moreover, the directed graph $(V, \{(A, B) \in V \times V \mid B \text{ occurs in } \mathsf{rhs}(A)\})$ (the graph of $\mathcal{C}$) has to be acyclic. Every gate $A \in V$ evaluates to an element $\mathrm{val}_{\mathcal{C}}(A) \in S$ in the natural way and we set $\mathrm{val}(\mathcal{C}) = \mathrm{val}_{\mathcal{C}}(A_0)$. A circuit over $\mathcal{S}$ is called skew if for every multiplication gate A one of the two gates in $\mathsf{rhs}(A)$ is an input gate.

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 $\mathrm{val}(\mathcal{A}) = \sum_{p \in \mathcal{P}} \lambda(p)$. Skew circuits and branching programs are basically the same objects (the edge labels of the branching program correspond to the constant inputs of multiplication gates in the skew circuit).

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 $\mathcal{A}$, i.e., the $(n \times n)$-matrix M with $M[i, j] = \lambda(i, j)$. Then $\mathrm{val}(\mathcal{A})$ is the $(1, n)$-entry of the matrix $\sum_{i=0}^{n} M^i$. For many semirings $\mathcal{S}$, this fact can be used to get an NC^2-algorithm for computing $\mathrm{val}(\mathcal{A})$. The $n + 1$ matrix powers M^i ($0 \leq i \leq n$) can be computed in parallel, and every power can be computed by a balanced tree of height $\log i \leq \log n$, 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 e.g. [22]: $(\mathbb{Z}[x], +, \cdot)$, $(\mathbb{Z}_n[x], +, \cdot)$ for $n \geq 2$, $(\mathbb{Z} \cup \{\infty\}, \min, +)$, 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 a^n (and hence need at most $n \log a$ bits), where a is the maximal absolute value of a coefficient in M. Hence point (i) above holds, and we get:

Lemma 1. *The output value of a given skew circuit (or branching program) over one of the following semirings can be computed in* NC^2*:*

(a) $(\mathbb{Z}[x], +, \cdot)$ *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)*
(b) $(\mathbb{Z} \cup \{\infty\}, \min, +)$ *and* $(\mathbb{Z} \cup \{-\infty\}, \max, +)$ *(integers are given in binary representation)*

Point (a) of Lemma 1 also holds for the polynomial rings $(\mathbb{Z}[x_1, \dots, x_k], +, \cdot)$ and $(\mathbb{Z}_n[x_1, \dots, x_k], +, \cdot)$ as long as the number k of variables is not part of the input: The polynomial $\prod_{i=1}^{k}(x_i + 1)$ 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 (a): The polynomial $\prod_{i=1}^{n}(x^{2^i} + 1)$ has 2^n monomials but can be represented by a branching program with $O(n)$ edges labeled by the polynomials $x^{2^i} + 1$.

In this paper, we will deal with circuits over a polynomial ring $R[x_1, \dots, x_k]$, where R is $(\mathbb{Z}, +, \cdot)$ or $(\mathbb{Z}_n, +, \cdot)$. By definition, in such a circuit every input gate is labelled with a polynomial from $R[x_1, \dots, x_k]$. 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 $R[x_1, \dots, x_k]$, where the right-hand sides of input gates are polynomials given in *succinct representation*. For general circuits this makes no real difference (since a big power $x_i^{n_i}$ can be defined by a subcircuit of size $O(\log n_i)$ 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 $R[x_1, \dots, x_k]$ is a skew circuit over the ring $R[x_1, \dots, x_k]$ 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 $\mathsf{rhs}(A) = p(x_1, \dots, x_k)$, then $\mathsf{size}_{\mathcal{C}}(A) = \|p(x_1, \dots, x_k)\|$. 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 $R[x_1, \dots, x_k]$, where every edge label $\lambda(e)$ 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 $R[x_1, \dots, x_k]$ into an equivalent standard skew circuit (where every input gate is labelled 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 $\lceil \log n \rceil$.

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: Given a number $k \geq 1$ and a circuit $\mathcal{C}$ over the ring $R[x_1, \ldots, x_k]$, 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 [9]; for $\mathbb{Z}_n$ with n composite, it was shown in [1].

Theorem 1. *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 $\deg(p, x_i)$ 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 $d \in \mathbb{N}$ such that $\deg(p, x_i) < d$ for all $1 \leq i \leq k$. We define the univariate polynomial $U(p, d) = p(y^1, y^d, \ldots, y^{d^{k-1}})$. It is obtained from $p(x_1, \ldots, x_k)$ by replacing every monomial $a \cdot x_1^{n_1} \cdots x_k^{n_k}$ 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 $U(p, d)$ is the zero-polynomial. The following lemma can be shown for arbitrary circuits, but we will only need it for powerful skew circuits.

Lemma 2. *Given a powerful skew circuit $\mathcal{C}$ for the polynomial $p(x_1, \ldots, x_k)$, one can compute in NC^2 (i) the binary encoding of a number d with $\deg(p, x_i) < d$ for all $1 \leq i \leq k$ and (ii) a powerful skew circuit $\mathcal{C}'$ for $U(p, d)$.*

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

The main result of this paper is:

Theorem 2. *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 Thm. 2 has two main ingredients: The randomized PIT algorithm of Agrawal and Biswas [1] and the modular polynomial powering algorithm of Fich and Tompa [5]. Let us start with the Agrawal-Biswas algorithm. We 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 $A_{b,t} = x^t + \sum_{i=0}^{\ell-1} b_i \cdot x^i$.
4. Accept, if $P(x) \bmod T_{r,b,t} = 0$, 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 3 ([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 ([5]). *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 m.
Output: The polynomial $a(x)^m$ mod $q(x)$.

Remark 1. In [5], 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 $A_1 A_2 \cdots A_k$ (for k polynomial in n) 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. Then we compute the product $A_1 A_2 \cdots A_k$ by a balanced binary tree of depth $\log k$. Also logspace-uniformity of the circuits is not stated explicitly in [5], but follows easily since only standard arithmetical operations on binary coded numbers are used.

Proof of Thm. 2. By Lemma 2 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 $\mathcal{A} = (\{1, \ldots, n\}, 1, n, \lambda)$ a powerful branching program with n nodes that is equivalent to $\mathcal{C}$. 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 Agrawal-Biswas identity test. 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 2. For the number ℓ we can take the number of bits of this degree bound, which is polynomial in the input size.

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 size) 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. Computing 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 $A_{b,t} = x^t + \sum_{i=0}^{\ell-1} b_i \cdot x^i$, is an instance of iterated multiplication (for the powers $A_{b,t}(x)^i$) and iterated addition of polynomials. Hence, by Prop. 1 also this step can be carried out in NC^1. Note that the degree of $T_{r,b,t}(x)$ is $\ell \cdot (r-1) \in O(\ell^3 \log p)$, i.e., polynomial in the input size.

Step 4. For the last step, we have to compute $P(x) \bmod T_{r,b,t}(x)$. For this, we consider in parallel all monomials $a \cdot x^k$ that occur in an edge label of our powerful branching program $\mathcal{A}$. Recall that $a \in \mathbb{F}_p$ and k is given in binary notation. Using the Fich-Tompa algorithm we compute $x^k \bmod T_{r,b,t}(x)$ in NC^2. We then replace the edge label $a \cdot x^k$ by $a \cdot (x^k \bmod T_{r,b,t}(x))$. Let $\mathcal{B}$ the resulting branching program. Every polynomial that appears as an edge label in $\mathcal{B}$ is now given in standard form. Hence, by Lemma 1 we can compute in NC^2 the output polynomial $\mathsf{val}(\mathcal{B})$. Clearly, $P(x) \bmod T_{r,b,t}(x) = \mathsf{val}(\mathcal{B}) \bmod T_{r,b,t}(x)$, which can be computed in NC^1 by Prop. 1.

Let us now prove Thm. 2 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 $|a|$, where $a \cdot x^k$ is an edge label of $\mathcal{A}$. Since there are at most 2^n many paths from s to t in $\mathcal{A}$, every coefficient of $P(x)$ belongs to the interval $[-(2m)^n, (2m)^n]$. Let $k = (\lceil \log(m) \rceil + 1) \cdot (n+1)$ 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 primes can be computed in logspace by doing all necessary divisibility checks on binary encoded numbers having only $O(\log k)$ bits. The Chinese remainder theorem implies that $P(x) = 0$ if and only if $P(x) \equiv 0 \bmod p_i$ for all $1 \le i \le k$. 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. □

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 finite fields.

5 Multi-dimensional straight-line programs

Let Γ be a finite alphabet. For $l \in \mathbb{N}$ let $[0, l] = \{0, 1, \ldots, l\}$. An *$n$-dimensional picture* over Γ is a mapping $p : \prod_{j=1}^n [0, l_j - 1] \to \Gamma$ for some $l_j \in \mathbb{N}$. Let $\mathrm{dom}(p) = \prod_{j=1}^n [0, l_j - 1]$. 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 $(k_1, \ldots, k_n) \in \prod_{j=1}^n [0, l_j - 1]$ we finally set $(p \circ_i q)(k_1, \ldots, k_n) = p(k_1, \ldots, k_n)$

if $k_i < |p|_i$ 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 $k_i \geq |p|_i$. These operations generalize the concatenation of finite words.

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 (i) the relation $\{(A, B) \in V \times V \mid B \text{ occurs in } \mathsf{rhs}(A)\}$ is acyclic, and (ii) 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 define the size of the SLP $\mathbb{A} = (V, \Gamma, S, P)$ as $|\mathbb{A}| = |V|$. Note that the length of the picture $\mathrm{val}(\mathbb{A})$ (in each dimension) can be exponential in $|\mathbb{A}|$. A one-dimensional SLP is a context-free grammar that generates a single word [15]. Two-dimensional SLPs were studied in [3].

Given two n-dimensional SLPs we want to know whether they evaluate to the same picture. In [3] it was shown that this problem belongs to coRP by translating it to PIT. For an n-dimensional picture $p : \mathrm{dom}(p) \to \{0, 1\}$ we define the polynomial

$$f_p(x_1, ..., x_n) = \sum_{(k_1,...,k_n) \in \mathrm{dom}(p)} p(k_1, ..., k_n) \prod_{i=1}^{n} x_i^{k_i}.$$

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 $1 \leq i \leq n$ we clearly have $p = q$ if and only if $f_p + f_q = 0$ (recall that coefficients are from $\mathbb{Z}_2$). In [3], it was observed that from an SLP $\mathbb{A}$ for a picture P, one can easily construct an arithmetic circuit $\mathcal{C}$ for the polynomial f_p, which leads to a coRP-algorithm for equality testing. For instance, $\mathsf{rhs}_{\mathbb{A}}(A) = B \circ_k C$ is translated into $\mathsf{rhs}_{\mathcal{C}}(A) = B + x_k^N \cdot C$, where $N = |B|_k$ can be precomputed in NC^2 (using Lemma 1). This shows that the circuit $\mathcal{C}$ is actually powerful skew and can be constructed in NC^2 (see the appendix for details). Hence, we get:

Theorem 5. *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).*

It should be noted that even in the one-dimensional case (where SLP-equality can be tested in polynomial time [8,19,21]), no randomized NC-algorithm was known before. For equality testing for SLPs of dimension $n \geq 2$ it remains open whether a polynomial time algorithm exists. For the one-dimensional case, a polynomial time algorithm exists [8,19,21], but no NC-algorithm is known.

6 Circuits over wreath products

As a second application of PIT for powerful skew circuits we consider the circuit evaluation problem, also known as the compressed word problem, for wreath products of finitely generated abelian groups. We assume some basic familiarity with group theory.

6.1 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 [20]. Here, we are interested in the *compressed word problem* for a f.g. group. For this, the input word w is given in 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. In the following we always mean one-dimensional SLPs when using the term SLP. The compressed word problem for G asks, whether $\mathrm{val}(\mathbb{A}) = 1$ in G for a given SLP $\mathbb{A}$.

As mentioned in the introduction, there are important classes of groups with a polynomial time compressed word problem. Moreover, for f.g. linear groups the compressed word problem belongs to coRP. Concerning the parallel complexity, it was recently shown in [12] (using results from [2]) that $\mathsf{CWP}(G) \in \mathsf{NC}^2$, whenever G has a f.g. nilpotent normal subgroup H such that the quotient G/H is finite and solvable (a so called f.g. nilpotent by finite solvable group). To the knowledge of the authors, there are no other known examples of groups with a compressed word problem in NC.

6.2 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). 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 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 at $a \in G$ (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 the elements h_1 and h_2 are sitting at $a \in G$ (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 $g_1 g_2$.

Proofs of the following lemmas can be found in the long version [13].

Lemma 3. *The group $(A \times B) \wr G$ embeds into $(A \wr G) \times (B \wr G)$.*

Lemma 4. *For every $k \geq 1$ and every finitely generated group G,* $\mathsf{CWP}(G \wr \mathbb{Z}^k)$ *is* NC^2*-reducible to* $\mathsf{CWP}(G \wr \mathbb{Z})$.

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

In this section, we show that $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z})$ (resp., $\mathbb{Z}_n \wr \mathbb{Z}$) and PIT for powerful skew circuits over $\mathbb{Z}[x]$ (resp., $\mathbb{Z}_n[x]$) are equivalent w.r.t. NC^2-reductions.

We consider the generators a and t of $\mathbb{Z} \wr \mathbb{Z}$, where $a = (0, 1)$ and $t = (f, 0)$ with $f(0) = 1$ and $f(x) = 0$ for $x \neq 0$. So, multiplying with a (resp., a^{-1}) on the right corresponds to moving the cursor to the left (resp., right) and multiplying with t (resp., t^{-1}) on the right corresponds to adding (resp., subtacting) one from the value at the current cursor position. Let $\Gamma = \{a, t, a^{-1}, t^{-1}\}$. The main result of this section is:

Theorem 6. *The compressed word problem for $\mathbb{Z} \wr \mathbb{Z}$ (resp., $\mathbb{Z}_n \wr \mathbb{Z}$) is equivalent w.r.t. NC^2-reductions to PIT for powerful skew circuits over the ring $\mathbb{Z}[x]$ (resp., $\mathbb{Z}_n[x]$).*

Going from PIT for powerful skew circuits over the ring $\mathbb{Z}[x]$ to $\mathsf{CWP}(\mathbb{Z}\wr\mathbb{Z})$ is relatively easy (and the same reduction works for $\mathbb{Z}_n$ instead of $\mathbb{Z}$): We encode a polynomial $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ by the group element $g(p(x)) = (f, 0) \in \mathbb{Z} \wr \mathbb{Z}$, where $f(k) = a_k$ for all $0 \leq k \leq n$ and $f(z) = 0$ for all other integers z. The idea now is to construct from a given powerful skew circuit $\mathcal{C}$ over the ring $\mathbb{Z}[x]$ an SLP $\mathbb{A}$ over the alphabet Γ such that $\mathrm{val}(\mathbb{A})$ evaluates (in $\mathbb{Z} \wr \mathbb{Z}$) to the group element $g(\mathrm{val}(\mathcal{C}))$. Note that $g(p_1(x) + p_2(x)) = g(p_1(x))g(p_2(x))$ in the group $\mathbb{Z} \wr \mathbb{Z}$. This allows to deal with addition gates in $\mathcal{C}$. Multiplication gates in general circuits cannot be handeled in this way. But fortunately, $\mathcal{C}$ is powerful skew, and we can make use of the following identity, where $m, n \in \mathbb{N}$: $g(m \cdot x^n \cdot p(x)) = a^n g(p(x))^m a^{-n}$ (conjugation by a^n corresponds to multiplication with the monomial x^n).

Going from $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z})$ back to PIT for powerful skew circuits over the ring $\mathbb{Z}[x]$ is based on the same correspondence between polynomials and elements of $\mathbb{Z} \wr \mathbb{Z}$, but is slightly more technical. The problem is that for a group element $(f, z) \in \mathbb{Z} \wr \mathbb{Z}$ there might be a negative $a \in \mathbb{Z}$ with $f(a) \neq 0$. Hence, encoding f by a polynomial would in fact lead to a Laurent polynomial. But we can avoid this problem by conjugating all elements of $\mathbb{Z} \wr \mathbb{Z}$ with a large enough power a^n such that the domain of the above function f is contained in the non-negative integers; see [13] for details.

By Lemma 3 and Lemma 4, $\mathsf{CWP}((G \times H) \wr \mathbb{Z}^n)$ is NC^2-reducible to $\mathsf{CWP}(G \wr \mathbb{Z})$ and $\mathsf{CWP}(H \wr \mathbb{Z})$. Together with Thm. 2 and Thm. 6 we obtain the following result:

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

It is not clear, whether in Cor. 1 we can replace G by an arbitrary finitely generated abelian group. On the other hand, if we apply Thm. 1 instead of Thm. 2 we obtain:

Corollary 2. *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 $K \leq H$ be a f.g. abelian subgroup of finite index m in H. Since K has the form $A \times \mathbb{Z}^k$ for some $k \geq 0$ with A finite abelian, we can (by increasing the finite index m) assume that $K = \mathbb{Z}^k$ for some $k \geq 0$. It is shown in [17] that $G^m \wr \mathbb{Z}^k \cong G^m \wr K$ 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 $\mathsf{CWP}(A)$ [16, Thm. 4.4]. Since $G^m \wr \mathbb{Z}^k$ is a finite index subgroup of $G \wr H$, it suffices to show that $\mathsf{CWP}(G^m \wr \mathbb{Z}^k) \in \mathsf{coRP}$. Since G^m is finitely generated abelian, it suffices to consider $\mathsf{CWP}(\mathbb{Z}_n \wr \mathbb{Z}^k)$ $(n \geq 2)$ and $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z}^k)$. The case $k = 1$ is clear, so assume that $k \geq 1$. By Cor. 1, $\mathsf{CWP}(\mathbb{Z} \wr \mathbb{Z}^k) \in \mathsf{coRNC}$ and by Thm. 1 and 6, $\mathsf{CWP}(\mathbb{Z}_n \wr \mathbb{Z}^k) \in \mathsf{coRP}$. $\square$

In the full version [13] of this paper, Cor. 1 is further applied to the compressed word problem for quotients of free groups with respect to commutator subgroups.

References

1. M. Agrawal and S. Biswas. Primality and identity testing via Chinese remaindering. *J. Assoc. Comput. Mach.*, 50(4):429–443, 2003.
2. M. Beaudry, P. McKenzie, P. Péladeau, and D. Thérien. Finite monoids: From word to circuit evaluation. *SIAM J. Comput.*, 26(1):138–152, 1997.
3. P. Berman, M. Karpinski, L. L. Larmore, W. Plandowski, and W. Rytter. On the complexity of pattern matching for highly compressed two-dimensional texts. *J. Comput. System Sci.*, 65(2):332–350, 2002.
4. W. Eberly. Very fast parallel polynomial arithmetic. *SIAM J. Comput.*, 18(5):955–976, 1989.
5. F. E. Fich and M. Tompa. The parallel complexity of exponentiating polynomials over finite fields. *J. Assoc. Comput. Mach.*, 35(3): 651-667, 1988.
6. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995.
7. W. Hesse, E. Allender, and D. A. M. Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. System Sci.*, 65:695–716, 2002.
8. Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theor. Comput. Sci.*, 158(1&2):143–159, 1996.
9. 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.
10. R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proc. STOC'97*, pages 220–229. ACM Press, 1997.
11. V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.*, 13(1-2):1–46, 2004.
12. D. König M. Lohrey. Evaluating matrix circuits. to appear in *Proc. COCOON 2015*.
13. D. König M. Lohrey. Parallel identity testing for algebraic branching programs with big powers and applications. arXiv.org, 2015. http://arxiv.org/abs/1502.04545
14. M. Lohrey. Word problems and membership problems on compressed words. *SIAM J. Comput.*, 35(5):1210 – 1240, 2006.
15. M. Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.
16. M. Lohrey. *The Compressed Word Problem for Groups*. SpringerBriefs in Mathematics. Springer, 2014.
17. M. Lohrey, B. Steinberg, and G. Zetzsche. Rational subsets and submonoids of wreath products. *Information and Computation*, 2014. doi:10.1016/j.ic.2014.12.014.
18. W. Magnus. On a theorem of Marshall Hall. *Ann. of Math. (2)*, 40:764–768, 1939.
19. K. Mehlhorn, R. Sundar, and C. Uhrig. Maintaining dynamic sequences under equality tests in polylogarithmic time. *Algorithmica*, 17(2):183–198, 1997.
20. P. S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Amer. Math. Soc. Transl. Ser. 2*, 9:1–122, 1958.
21. W. Plandowski. Testing equivalence of morphisms on context-free languages. In *Proc. ESA 1994*, volume 855 of *Lecture Notes in Computer Science*, pages 460–470. Springer, 1994.
22. H. Vollmer. *Introduction to Circuit Complexity*. Springer, 1999.