Electronic Colloquium on Computational Complexity, Report No. 74 (2017)

Efficient Identity Testing and Polynomial Factorization over Non-associative Free Rings

V. Arvind[*] Rajit Datta [†] Partha Mukhopadhyay [‡] S. Raja[§]

April 29, 2017

Abstract

In this paper we study arithmetic computations over non-associative, and non-commutative free polynomials ring $\mathbb{F}\{x_1, x_2, \ldots, x_n\}$. Prior to this work, the non-associative arithmetic model of computation was considered by Hrubes, Wigderson, and Yehudayoff [HWY10]. They were interested in completeness and explicit lower bound results.

We focus on two main problems in algebraic complexity theory: Polynomial Identity Testing (PIT) and polynomial factorization over $\mathbb{F}\{x_1, x_2, \ldots, x_n\}$. We show the following results.

1. Given an arithmetic circuit C of size s computing a polynomial $f \in \mathbb{F}\{x_1, x_2, \ldots, x_n\}$ of degree d, we give a deterministic poly(n, s, d) algorithm to decide if f is identically zero polynomial or not. Our result is obtained by a suitable adaptation of the PIT algorithm of Raz-Shpilka [RS05] for non-commutative ABPs.
2. Given an arithmetic circuit C of size s computing a polynomial $f \in \mathbb{F}\{x_1, x_2, \ldots, x_n\}$ of degree d, we give an efficient deterministic algorithm to compute the circuits for the irreducible factors of f in time poly(n, s, d) when $\mathbb{F} = \mathbb{Q}$. Over finite fields of characteristic p, our algorithm runs in time poly(n, s, d, p).

1 Introduction

Non-commutative computation, introduced in complexity theory by Hyafil [Hya77] and Nisan [Nis91], is a central field of algebraic complexity theory. The main algebraic structure of interest is the free non-commutative ring $\mathbb{F}\langle X\rangle$ over a field $\mathbb{F}$, where $X = \{x_1, x_2, \cdots, x_n\}$ is a set of free non-commuting variables. One of the main problems in the subject is non-commutative Polynomial Identity Testing. The problem can be stated as follows:

Let $f \in \mathbb{F}\langle X\rangle$ be a polynomial represented by a non-commutative arithmetic circuit C. The polynomial f can be either given by a black-box for C (using which we can evaluate C on matrices with entries from $\mathbb{F}$ or an extension field), or the circuit C may be explicitly given. The algorithmic problem is to check if the polynomial computed by C is identically zero.

We recall the formal definition of a non-commutative arithmetic circuit.

[*]Institute of Mathematical Sciences (HBNI), Chennai, India, email: arvind@imsc.res.in
[†]Chennai Mathematical Institute, Chennai, India, email: rajit@cmi.ac.in
[‡]Chennai Mathematical Institute, Chennai, India, email: partham@cmi.ac.in
[§]Chennai Mathematical Institute, Chennai, India, email: sraja@cmi.ac.in

ISSN 1433-8092

Definition 1. *A* non-commutative arithmetic circuit C *over a field* $\mathbb{F}$ *and indeterminates* $x_1, x_2, \cdots, x_n$ *is a directed acyclic graph (DAG) with each node of indegree zero labeled by a variable or a scalar constant from* $\mathbb{F}$*: the indegree* 0 *nodes are the input nodes of the circuit. Each internal node of the DAG is of indegree two and is labeled by either a* $+$ *or a* $\times$ *(indicating that it is a plus gate or multiply gate, respectively). Furthermore, the two inputs to each* $\times$ *gate are designated as left and right inputs which is the order in which the gate multiplication is done. A gate of* C *is designated as* output. *Each internal gate computes a polynomial (by adding or multiplying its input polynomials), where the polynomial computed at an input node is just its label. The* polynomial computed *by the circuit is the polynomial computed at its output gate.*

Since the cancellation of terms are restricted by non-commutativity, it is generally believed that the polynomial identity question could be easier in non-commutative setting than identity testing problem in commutative setting. This is partially supported by the deterministic polynomial-time white-box PIT algorithm for non-commutative ABP [RS05]. Such a result in commutative setting will be a huge breakthrough [1]. Yet, the progress towards a deterministic PIT result for general non-commutative arithmetic circuits is very slow. For example, such an algorithm is missing even for non-commutative *skew* circuits. In this work, we show that it is the property of *associativity* that makes the problem hard. In particular consider the non-commutative and non-associative ring of polynomials $\mathbb{F}\{x_1, x_2, \ldots, x_n\}$ [2] where every monomial comes with a bracketing order of multiplication. For example, in this model $(x_1(x_2x_1))$ is different from $((x_1x_2)x_1)$ where as a non-commutative monomial they are the same. Previously, the non-associative arithmetic model of computation was considered by Hrubes, Wigderson, and Yehudayoff [HWY10]. They showed completeness and explicit lower bound results for this model. We show the following result about PIT.

Theorem 1. *Let* $f(x_1, x_2, \ldots, x_n) \in \mathbb{F}\{X\}$ *be a degree* d *polynomial given by an arithmetic circuit of size* s*. Then in deterministic* $\text{poly}(s, n, d)$ *time we can test if* f *is an identically zero polynomial in* $\mathbb{F}\{X\}$*.*

Remark 1. *We note that our algorithm in Theorem 1 does not depend on the choice of the field* $\mathbb{F}$*. A recent result of Lagarde et al. [LMP16], among other results, show an exponential lower bound, and a deterministic polynomial-time* PIT *algorithm over* $\mathbb{R}$ *for non-commutative circuits where all parse trees in the circuit are isomorphic. We also note that Arvind et al. [AR16] show an exponential lower bound for set-multilinear arithmetic circuits where for each monomial, all its parse trees are unique, but two distinct monomials can have distinct parse tree structures.*

Next, we consider the polynomial factorization problem over $\mathbb{F}\{\text{X}\}$. Over usual commutative setting the polynomial factorization problem is defined as follows: Given an arithmetic circuit C computing a multivariate polynomial $f(x_1, x_2, \ldots, x_n) \in \mathbb{F}[\text{X}]$ of degree d, find circuits for the irreducible factors of f efficiently. A remarkable result of Kaltofen [Kal] solves the problem in randomized $\text{poly}(n, s, d)$ time and finding a deterministic solution is an outstanding open problem. Recently, it has been proved that the complexity of deterministic polynomial factorization problem and the PIT problem are polynomially equivalent [KSS15]. A natural question is to study the problem over $\mathbb{F}\langle X\rangle$. A key feature of non-commutative ring of polynomial is that $\mathbb{F}\langle X\rangle$ is not

[1] The situation is similar even in the lower bound case where Nisan proved that non-commutative determinant or permanent polynomial would require exponential-size algebraic branching program [Nis91].

[2] Sometime we denote it by $\mathbb{F}\{X\}$.

even an *unique factorization domain.* A recent result of Arvind et al. [AJR15] shows that with additional promise that the factors are variable disjoint, one can solve the problem efficiently and such a restriction ensures that the irreducible factorization is unique. In this paper we observe that $\mathbb{F}\{X\}$ is an *unique factorization domain* and we can solve the problem of finding irreducible factors (by circuits) in deterministic polynomial time.

Theorem 2. *Let $f(x_1, x_2, \ldots, x_n) \in \mathbb{F}\{X\}$ be a degree d polynomial given by an arithmetic circuit of size s. Then if $\mathbb{F} = \mathbb{Q}$, in deterministic $\mathrm{poly}(s, n, d)$ time we can output the circuits for the irreducible factors of f. Over positive characteristic fields $\mathbb{F}$ where char$(\mathbb{F}) = p$, we get a deterministic $\mathrm{poly}(s, n, d, p)$ time algorithm.*

Outline of the proofs

- **Identity Testing Result:** The algorithm is based on a suitable adaptation of white-box PIT algorithm for non-commutative ABPs [RS05]. We note that like Raz-Shpilka [RS05] PIT algorithm, if given circuit computes a nonzero polynomial $f \in \mathbb{F}\{X\}$, then this algorithm output a *certificate monomial* m such that coefficient of m in f is nonzero. We first sketch the main steps of the algorithm in [RS05] in a way that fits to our purpose. This view of their algorithm was also used in designing non-commutative ABP interpolation algorithm [AMS10].

  The algorithm processes the ABP layer by layer. At layer i of the ABP, algorithm maintains a set $\mathbb{B}_i$ of linearly independent coefficients vector of monomials and their size is bounded by width of the layer. The set $\mathbb{B}_i$ has the property that all the coefficients vector for monomials at the layer i can be written as a linear combination of the vectors in $\mathbb{B}_i$. The construction of $\mathbb{B}_{i+1}$ from $\mathbb{B}_i$ can be done efficiently. Clearly the identity testing problem can be solved by observing the set $\mathbb{B}_d$ where d is the depth of the ABP.

  Now we describe the main steps of our PIT algorithm for polynomials over $\mathbb{F}\{\mathrm{X}\}$ given by circuits. Let f be the input polynomial given by the circuit C. It is an easy observation that we can think the monomials of f are encoded over $\mathbb{F}\langle \mathrm{X}, (,) \rangle$ preserving the multiplication structure (Observation 1). Also, w.l.o.g assume that the $\times$ gates are of fan-in two and the $+$ gates are of unbounded fan-in. We can also compute different homogeneous degree parts $\{C_j : 1 \leq j \leq d\}$ from the given circuit efficiently and it is enough to test each of the circuits C_j for identity. So we only sketch the identity testing steps for a homogeneous degree d polynomial given by a circuit C.

  For each degree $j \leq d$ such that there is a degree j gate in the circuit C, algorithm maintains a set $\mathbb{B}_j$ of linearly independent coefficients vector of monomials such that for all other monomial of degree j computed by the circuit C, its coefficients vector can be written as a linear combination of basis vectors $\mathbb{B}_j$. Clearly the size of $\mathbb{B}_j$ is bounded by the size of the circuit. Then we show how to obtain the set $\mathbb{B}_{j+1}$ from all the sets $\{\mathbb{B}_i : 1 \leq i \leq j\}$. For each of the basis vectors we also keep track of the indexing monomials. In the non-associative model a degree d monomial $m = (m_1 m_2)$ can be generated in *unique way.* So to prove that the coefficient vector for such a monomial is in the span of $\mathbb{B}_d$ it is enough to look at the vectors generated from the span of $\mathbb{B}_{d_1}$ and $\mathbb{B}_{d_2}$ where $d_1 = \deg(m_1)$ and $d_2 = \deg(m_2)$. This is a crucial difference from a general non-commutative circuit.

- **Polynomial Factorization Over Non-associative Free Rings:**

  The factoring algorithm builds on the PIT algorithm outlined above. In an earlier paper, Arvind et al. observed that given a monomial m and a homogeneous non-commutative circuit C, one can compute the formal left and right derivatives of C with respect to m efficiently [AJR15]. We need this result too in our algorithm. To sketch the algorithm, consider an easy case when the given polynomial f can be factored as $f = gh$ with the additional promise that the constant terms of f, g, h are zero. Let the degrees of f, g, h are d, d_1, d_2 respectively. Clearly $f_d = g_{d_1} h_{d_2}$ where these are homogeneous degree d, d_1, d_2 parts. Invoking the PIT algorithm over C_d we recover any nonzero monomial $m = (m_1\ m_2)$ of degree d in f_d along with its coefficient $c_m(f)$[3]. Then clearly, for a nontrivial factorization m_1 is a nonzero monomial in g and m_2 is a nonzero monomial in h. Notice that the left derivative of C_d with respect to m_1 gives $c_{m_1}(g)\ h_{d_2}$ and the right derivative of C_d with respect to m_2 gives $c_{m_2}(h)\ g_{d_1}$. We use these derivative circuits and the non-associative structure of the circuit, to gradually build circuits for different homogeneous parts of g and h. The case where f, g, and h may have nonzero constant terms, is technically more complicated but the essential ideas are similar.

Organization

The paper is organized as follows. In Section 2, we state and prove some simple properties of non-associative and non-commutative polynomials useful for our proofs. In Section 3 we prove Theorem 1. In Section 4 we prove Theorem 2. We state a few open problems in Section 5.

2 Preliminaries

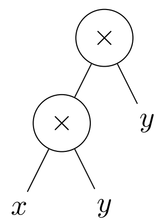
For an arithmetic circuit C, a *parse tree* for a monomial m is a multiplicative sub-circuit of C rooted at the output gate defined by the following process starting from the output gate:

- At each $+$ gate retain exactly one of its input gates.
- At each $\times$ gate retain both its input gates.
- Retain all inputs that are reached by this process.
- The resulting subcircuit is multiplicative and computes a monomial m (with some coefficient).

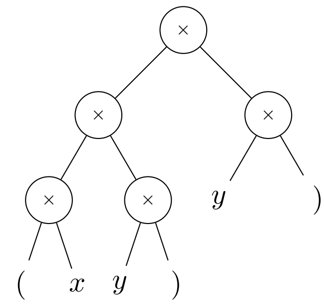
Over the non-associative model $\mathbb{F}\{X\}$, the same definition for the parse tree of a monomial applies and as already mentioned in the introduction that each such parse tree (or the generation of the monomial) comes with a bracketing order of multiplication. It is convenient to view a polynomial in $\mathbb{F}\{x_1, \ldots, x_n\}$ as an element in the non-commutative ring $\mathbb{F}\langle x_1, \ldots, x_n, (,)\rangle$ where we introduce two auxiliary variables (and) (for left and right bracketing) to encode the parse tree structure of any monomial. We illustrate the encoding by the following example.

[3]For any polynomial f and a monomial m we use the notation $c_m(f)$ to denote the coefficient of m in f.

Consider the following example of a monomial in non-associative model whose parse tree is shown in Figure 1a. The encoding of the monomial in $\mathbb{F}\langle x_1, \ldots, x_n, (,)\rangle$ is simply $((\ x\ y\ )\ y\ )$ and shown in Figure 1b.



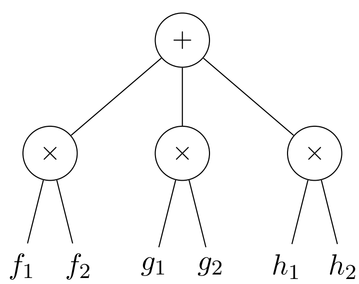
(a) A non-associative and non-commutative monomial xyy



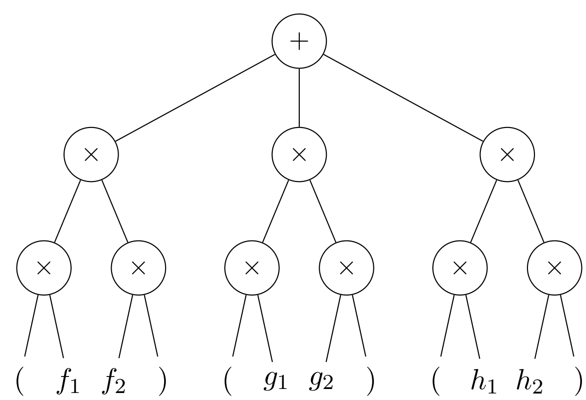
(b) Corresponding non-commutative bracketed monomial $((xy)\ y)$.

Figure 1: Non-associative & non-commutative monomial and its corresponding non-commutative bracketed monomial

Consider a polynomial $f \in \mathbb{F}\{X\}$ given by an arithmetic circuit C. One can easily get an equivalent polynomial $\tilde{f} \in \mathbb{F}\langle X, (,)\rangle$ computed by a circuit $\tilde{C}$ by just introducing the bracketing structure in each multiplication gate in C from the bottom of the circuit. For example, see Figure 2a and Figure 2b where f_i, g_i, h_i's are polynomials computed by sub-circuits. Clearly the bracket variables preserve the parse tree structure and does not trigger any new cancellations. The following fact is immediate.



(a) C computing a non-associative, non-commutative polynomial.



(b) $\tilde{C}$ computing the non commutative polynomial corresponding C.

Figure 2: Non-associative circuit and its corresponding non-commutative bracketed circuit

Observation 1. *$f \in \mathbb{F}\{X\} \neq 0$ if and only if $\tilde{f} \in \mathbb{F}\langle X, (,)\rangle \neq 0$.*

A free non-commutative ring $\mathbb{F}\langle \mathrm{X}\rangle$ is not a *unique factorization domain* (U.F.D). Consider the following standard example : $xyx + x = x(yx+1) = (xy+1)x$. In contrast, the non-associative free ring $\mathbb{F}\{\mathrm{X}\}$ is U.F.D. This property is crucial to even define the polynomial factorization problem studied in this paper.

Proposition 1. *Over any field $\mathbb{F}$ the non associative free ring $\mathbb{F}\{\mathrm{X}\}$ is a* unique factorization domain.

The proof of the proposition follows simply from the fact that each monomial in $\mathbb{F}\{\mathrm{X}\}$ can be generated in a *unique way*. Given a non-commutative circuit C computing a homogeneous polynomial in $\mathbb{F}\langle \mathrm{X}\rangle$ and a monomial m over X, one can talk of the left and right derivatives of C w.r.t m. This notion was first considered by Arvind et al. in [AJR15]. The polynomial f computed by C can be expressed as follows:

$$f = \sum_{\alpha} a_\alpha m \cdot m_\alpha + R_m.$$

The first part contains all the monomials of the form $m \cdot m_\alpha$ where $a_\alpha \in \mathbb{F}$ and the other monomials form the polynomial R_m. Then the left derivative

$$\frac{\partial^L C}{\partial_m} = \sum_{\alpha} a_\alpha m_\alpha.$$

Similarly we can define the right derivative. We note that circuits for left and right derivatives can be efficiently computed from the given circuit C. In [AM08, AMS10, AJR15], ideas similar to this were implicit. We briefly recall this in the following lemma.

Lemma 1. *Given a non-commutative circuit C of size s computing a homogeneous polynomial f of degree d in $\mathbb{F}\langle \mathrm{X}\rangle$ and monomial m, the left and right derivatives $\frac{\partial^L C}{\partial m}$ and $\frac{\partial^R C}{\partial m}$ can be computed deterministically in time* $\mathrm{poly}(n, d, s)$ *and the algorithm outputs the circuits $C_{m,L}$ and $C_{m,R}$ representing such derivatives.*

Proof. We explain the ideas only for left derivative, as similar ideas can be used for right derivative. Let m be a degree d' monomial and the homogeneous polynomial f of degree d computed by the circuit C as before. Now we can construct a small substitution deterministic finite automaton (DFA) A with $d'+2$ states, which recognizes set of all strings of length d with prefix m (i.e., strings of the form $m \cdot X^{d-d'}$) and substitutes 1 for prefix m. The transition matrices of the DFA can be represented by $(d'+2) \times (d'+2)$ matrices. Now evaluating the circuit C on the transition matrices we recover the circuit for $\frac{\partial^L C}{\partial m}$ in the $(1, d^{'}+1)^{th}$ entry of the final output matrix. □

Similarly we can define derivative of non-homogeneous polynomial f. The same matrix substitution works for non-homogeneous polynomials as well. We first remove constant term from f if any (e.g., by homogenization) and then evaluating the resulting circuit C on the transition matrices we recover the circuit for $\frac{\partial^L C}{\partial m}$ in the $(1, d^{'}+1)^{th}$ entry of the final output matrix.

As discussed above, the non-associative circuits can be encoded as non-commutative circuits over the variables $\{\mathrm{X}, (,)\}$, the left and right derivatives with respect to a given monomial can be efficiently computed. We use this in Section 4. We also note the following simple fact (see e.g., [SY10]) that the homogeneous parts of any non-commutative polynomial f given by a circuit C can be computed efficiently.

Lemma 2. *Given a non-commutative circuit C of size s computing a non-commuative polynomial f of degree d in $\mathbb{F}\langle \mathrm{X} \rangle$, one can compute homogeneous circuits C_j (where each gate computes a homogeneous polynomial) for j^{th} homogeneous part f_j of f, where $0 \leq j \leq d$, deterministically in time* $\mathrm{poly}(n, d, s)$.

3 Identity Testing Over Non-associative Free Rings

In this section we describe our identity testing algorithm. By Lemma 2, we assume that the given circuit C is homogenized (i.e. every gate computes a homogeneous polynomial). Also, the $\times$ gates have fan-in 2, $+$ gates have unbounded fan-in and the top gate is a sum gate. We also assume w.l.o.g that $+$ and $\times$ gates alternate in the circuit (otherwise we introduce sum gates with fan-in 1).

We define the j^{th}-layer to be the set of sum gates in the circuit computing degree j homogeneous polynomials. Let s^+ be the total number of sum gates in C. The idea is to consider a vector of coefficients $V_m \in \mathbb{F}^{s^+}$ corresponding to a monomial m indexed by sum gates of g in C.

$$V_m[g] = \mathrm{coeff}_m(f_g)$$

where f_g is the polynomial computed at the sum gate g.

We maintain a set of linearly independent basis vectors $\mathbb{B}_j$ for each $j \in [d]$-layer. The sets of vectors we construct inductively. Note that $|\mathbb{B}_j| \leq s$. Obviously the set $\mathbb{B}_1$ can be easily constructed. Inductively we assume that all the sets $\mathbb{B}_j : 1 \leq j \leq d-1$ are already constructed. We show the construction of $\mathbb{B}_d$. It is clear that the identity testing follows directly once we can construct $\mathbb{B}_d$.

We now describe the construction for the d-layer assuming we have basis $\mathbb{B}_j$ for every $j < d$. Consider a $\times$ gate with its children computing homogeneous polynomials of degree d_1 and d_2 respectively. Notice that $d = d_1 + d_2$ and $0 < d_1, d_2 < d$. Then consider the monomial set

$$M = \{m_1 m_2 \mid V_{m_1} \in \mathbb{B}_{d_1} \text{ and } V_{m_2} \in \mathbb{B}_{d_2}\}$$

We construct vectors $\{V_m \mid m \in M\}$ as follows.

$$V_{m_1 m_2}[g] = \sum_{(g_{d_1}, g_{d_2})} V_{m_1}[g_{d_1}] V_{m_2}[g_{d_2}]$$

where g is a $+$ gate in the d-layer, g_{d_1} is a $+$ gate in the d_1-layer and g_{d_2} is a $+$ gate in the d_2-layer and there is a $\times$ gate which is input to g and computes the product of g_{d_1} and g_{d_2}. Let $\mathbb{B}_{d_1,d_2}$ be a maximal linearly independent subset of $\{V_m \mid m \in M\}$. Then we let $\mathbb{B}_d$ be a maximal linearly independent subset of

$$\bigcup_{d_1+d_2=d} \mathbb{B}_{d_1,d_2}.$$

Claim 1. *For every monomial m of degree d, V_m is in the span of $\mathbb{B}_d$.*

Proof. Let $m = m_1 m_2$ and the degree of m_1 is d_1 and the degree of m_2 is d_2 [4]. By *Induction Hypothesis* V_{m_1} and V_{m_2} are in the span of $\mathbb{B}_{d_1}$ and $\mathbb{B}_{d_2}$ respectively and hence

$$V_{m_1} = \sum_{i=1}^{D_1} \alpha_i V_{m_i} \quad V_{m_i} \in \mathbb{B}_{d_1}$$

$$V_{m_2} = \sum_{j=1}^{D_2} \beta_j V_{m'_j} \quad V_{m'_j} \in \mathbb{B}_{d_2}$$

where D_i is the size of $\mathbb{B}_{d_i}$. Now, for a gate g in the d-layer,

$$\begin{aligned}
V_m[g] &= \sum_{(g_{d_1}, g_{d_2})} V_{m_1}[g_{d_1}] V_{m_2}[g_{d_2}] \\
&= \sum_{g_{d_1}, g_{d_2}} \Big(\sum_{i=1}^{D_1} \alpha_i V_{m_i}[g_{d_1}]\Big)\Big(\sum_{j=1}^{D_2} \beta_j V_{m'_j}[g_{d_2}]\Big) \qquad \textit{Induction Hypothesis} \\
&= \sum_{i=1}^{D_1} \sum_{j=1}^{D_2} \alpha_i \beta_j \sum_{g_{d_1}, g_{d_2}} V_{m_i}[g_{d_1}] V_{m'_j}[g_{d_2}] \\
&= \sum_{i=1}^{D_1} \sum_{j=1}^{D_2} \alpha_i \beta_j V_{m_i m'_j}[g] \qquad \text{by construction}
\end{aligned}$$

Thus V_m is in the span of $\mathbb{B}_{d_1,d_2}$ and hence in the span of $\mathbb{B}_d$.

□

Now given a circuit C of size s computing a polynomial $f \in \mathbb{F}\{\mathrm{X}\}$ of degree $\leq d$, we compute the homogeneous circuits $C_j \; : \; 0 \leq j \leq d$ (Lemma 2) efficiently and run the above algorithm on each of the circuits C_j to check whether f is identically zero. This completes the proof of Theorem 1.

4 Polynomial Factorization Over Non-associative Free Rings

In this section we give an efficient factorization algorithm for polynomials in $\mathbb{F}\{x_1, x_2, \ldots, x_n\}$ which outputs the factors of the input polynomial upto a scalar multiple. The algorithm uses the PIT algorithm over $\mathbb{F}\{x_1, x_2, \ldots, x_n\}$ as a main ingredient. It also crucially use the fact that the free ring $\mathbb{F}\{\mathrm{X}\}$ is U.F.D by Proposition 1. Before giving the complete proof we discuss an easy case first.

Lemma 3. *Let $f \in \mathbb{F}\{\mathrm{X}\}$ be given by a circuit C of size s and $f = g \cdot h$ where the degree of f, g, h are d, d_1, d_2 respectively. Additionally, it is promised that the constant terms in f, g, h are all zero. Then in deterministic* $\mathrm{poly}(n, d, s)$ *time we can compute the circuits for g and h.*

[4] Here a crucial point is that for a non-associative monomial of degree d, such a choice for d_1 and d_2 is *unique*. This is a place where a general non-commutative circuit behaves very differently.

Proof. Using Lemma 2, we first compute the circuits $C_j : 1 \leq j \leq d$ for the homogeneous j^{th} part of the polynomial f which is f_j. Clearly $f_d = g_{d_1} h_{d_2}$. We run the PIT algorithm on the circuit C_d to extract a monomial m of degree d along with its coefficient $c_m(f_d)$ in f_d. Notice that the monomial m is of the form $m = (m_1\, m_2)$. If g and h are nontrivial factors of f then it must be the case that m_1 and m_2 are monomials in g and h respectively. Compute the circuits for the left and right derivatives with respect to m_1 and m_2.

$$\frac{\partial^L C_d}{\partial m_1} = c_{m_1}(g_{d_1}) \cdot h_{d_2}$$

$$\frac{\partial^R C_d}{\partial m_2} C_d = c_{m_2}(h_{d_2}) \cdot g_{d_1}$$

In general the $(i + d_2)^{th} : i \leq d - d_2$ homogeneous part of f can be expressed as follows.

$$f_{i+d_2} = g_i h_{d_2} + \sum_{t=i+1}^{i+d_2-1} g_t h_{d_2\ -\ (t-i)}$$

We depict the circuit C_{i+d_2} for the polynomial f_{i+d_2} in Figure 3. The top gate of the circuit is a sum of polynomial fan-in under which there are product gates. From C_{i+d_2}, we construct another circuit C'_{i+d_2} by just keeping the product gates whose left degree is i and right degree is d_2. The resulting circuit is shown in Figure 4. The circuit C'_{i+d_2} must compute $g_i h_{d_2}$. By taking the right partial of C'_{i+d_2} with respect to m_2, we obtain the circuit for $c_{m_2}(h_{d_2})\ g_i$.

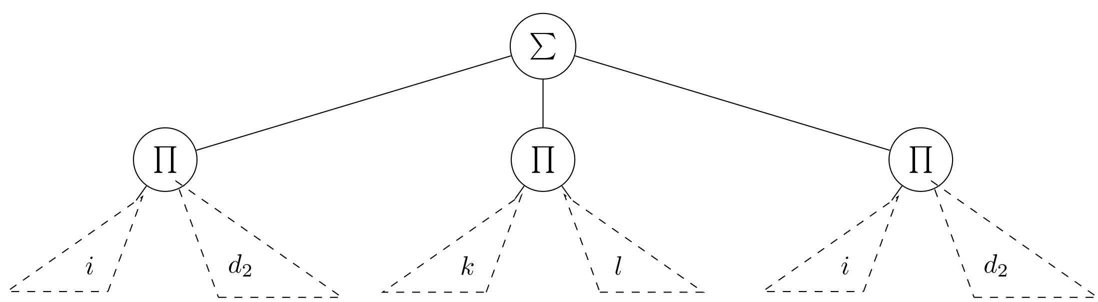


Figure 3: Circuit C_{i+d_2} of $i + d_2$th homogeneous part of f

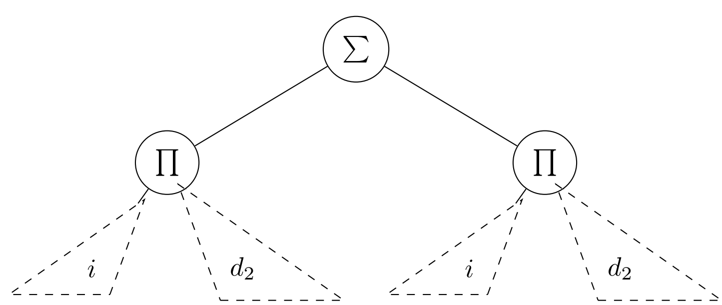


Figure 4: C'_{i+d_2} obtained by keeping only degree (i, d_2) type product gates, this computes $g_i h_{d_2}$

We repeat the above construction for each $i \in [d_1]$ to obtain circuits for $c_{m_2}(h_{d_2})g_i$ for $1 \leq i \leq d_1$. Similarly we can get the circuits for $c_{m_1}(g_{d_1})h_i$ for each $i \in [d_2]$ using the left derivatives with respect to the monomial m_1.

By adding the above circuits we get the circuits C_g and C_h for $c_{m_2}(h_{d_2})g$ and $c_{m_1}(g_{d_1})h$ respectively. We set $C_g = \frac{c_{m_2}(h_{d_2})}{c_m(f)}g$ so that $C_g C_h = f$. Using PIT algorithm one can easily check whether g and h are nontrivial factors. In that case we further recurse on g and h to obtain their irreducible factors.

□

Now we tackle the general case where in f, g, h the constant terms can be arbitrary. In the subsequent proofs we assume $\deg(g) \geq \deg(h)$ for clarity and recover the circuits for g and h. The case when $\deg(g) < \deg(h)$ can be handled in an analogous way by changing the role of left derivatives to right derivatives. We handle the case $\deg(g) = \deg(h)$ separately as follows.

Lemma 4. *Let $f = (g+\alpha)(h+\beta)$ and $\deg(g) = \deg(h)$. Given f by a circuit C we can efficiently compute $\gamma_1 g$ and $\gamma_2 h$, where $\gamma_1 = c_{m_2}(h)$ and $\gamma_2 = c_{m_1}(g)$.*

Proof. We note that $f = (g+\alpha)(h+\beta) = g\ h + \beta\ g + \alpha\ h + \alpha\ \beta$. Using the PIT algorithm on the highest degree homogeneous part of f, we get a maximum degree monomial m in f. Let m be of the structure $m = (m_1\ m_2)$. Left deriving the circuit w.r.t the monomial m_1 we get $c_{m_1}(g)h + \beta c_{m_1}(g) + \alpha c_{m_1}(h)$, removing the constant term we get a circuit for $c_{m_1}(g)h = \gamma_2 h$. Similarly right deriving w.r.t m_2 we get $c_{m_2}(h)g + \beta c_{m_2}(g) + \alpha c_{m_2}(h)$, removing the constant term we get a circuit for $c_{m_2}(h)g = \gamma_1 g$. □

When $\deg(g) > \deg(h)$ we can recover $h + \beta$ entirely (upto a scalar factor) and we need to obtain the homogeneous parts of g separately.

Lemma 5. *Let $f = (g+\alpha)\ (h+\beta)$ be a polynomial of degree d in $\mathbb{F}\{\mathrm{X}\}$ given by a circuit C of size s. Let $\deg(g) > \deg(h)$. Then we can efficiently generate the circuit C' for $\gamma_2(h+\beta)$ where $\gamma_2 = c_{m_1}(g)$.*

Proof. Using PIT algorithm on the highest degree homogeneous part of f we get a monomial $m \in f$ of degree d such that $m = (m_1 m_2)$. Notice that $f = g\ h + \alpha\ h + \beta\ g + \alpha\ \beta$. Since $\deg(g) > \deg(h)$, if we take the left partial derivative of f with respect to m_1, we will get $c_{m_1}(g)\ (h+\beta)$. So, C' computes $c_{m_1}(g)\ (h+\beta)$ and it can be efficiently constructed in time $\mathrm{poly}(n, s, d)$. □

By extracting homogeneous parts from the circuit C' obtained above we get circuits for $\{\gamma_2 h_i : i \in [d_2]\}$. We also get the constant term $\gamma_2\beta$. Now we obtain the homogeneous components of g as follows.

Lemma 6. *Let $f = (g+\alpha)\ (h+\beta)$ of degree d, and $\alpha, \beta \in \mathbb{F}$, degree of g is d_1 and degree of h is d_2 . Let the polynomial f be given by a circuit and also assume that $d_1 > d_2$. Let m be a degree d monomial present in f such that $m = (m_1\ m_2)$. Then one can efficiently compute circuits for $\{\gamma_1 g_i : i \in [d_1 - d_2 + 1, d_1]\}$, where $\gamma_1 = c_{m_2}(h)$.*

Proof. Fix any $i \in [d_1 - d_2 + 1, d_1]$, and compute the homogeneous $(i + d_2)^{th}$ part of f by a circuit C_{i+d_2}. Similar to Lemma 3, we focus on the sub-circuits of C_{i+d_2} formed by $\times$ gate of the degree type (i, d_2). Since i is at least $d_1 - d_2 + 1$, such gates can compute the multiplication of a degree i polynomial with a degree d_2 polynomial. Then, by taking the right partial derivative with respect to m_2 we recover the circuits for $c_{m_2}(h_{d_2})\ g_i$ for any $i \in [d_1 - d_2 + 1, d_1]$. □

Now the goal is to recover the circuits for g_i where $1 \le i \le d_1 - d_2$ (upto a scalar multiple) and also the constant terms α and β. When $i \le d_1 - d_2$ a product gate of type (i, d_2) can entirely come from g and so the above technique does not work directly.

Claim 2. *The $(d_2+i)^{th}$ homogeneous part of f is given by $f_{d_2+i} = \sum_{j=0}^{d_2-1} g_{d_2+i-j}\ h_j + g_i\ h_{d_2}$ for $1 \le i \le d_1 - d_2$. From the circuit C_{d_2+i} of f_{d_2+i}, we can efficiently compute the circuits for $\{\gamma_1 g_i : 1 \le i \le d_1 - d_2\}$, where $\gamma_1 = c_{m_2}(h_{d_2})$, and $h_0 = \beta$.*

Proof. For simplicity we describe the case when $i = d_1 - d_2$. Then

$$f_{d_1} = \beta g_{d_1} + \sum_{j=1}^{d_2-1} g_{d_1-j}\ h_j + g_{d_1-d_2}\ h_{d_2}.$$

From Lemma 5 we have a circuit C' for $\gamma_2(h+\beta)$, extracting the constant term we get $\gamma_2\beta$. From Lemma 6 we have a circuit $\tilde{C}$ for $\gamma_1 g_{d_1}$, multiplying these we get a circuit C^* for $\gamma_1\gamma_2\beta g_{d_1}$. Since $\gamma_1\gamma_2 = c_m(f)$ we can divide by $c_m(f)$ and get a circuit for βg_{d_1}. Note that we have circuits for every term appearing in the sum (only degree more that $d_1 - d_2$ homogeneous parts of g appear in the sum) except $g_{d_1-d_2}$. Subtracting out $\beta g_{d_1} + \sum_{j=1}^{d_2-1} g_{d_1-j}\ h_j$ from the circuit C_{d_1} we are left with the circuit for the polynomial $g_{d_1-d_2} h_{d_2}$. Right deriving the resulting circuit w.r.t m_2 (obtained from PIT algorithm as $m = m_1 m_2$) we get a circuit for $c_{m_2}(h) g_{d_1-d_2}$.

For g_i , $1 \le i < d_1 - d_2$, note that we will have circuits for all the terms appearing in the sum. Again subtracting and deriving we will get a circuit for $\gamma_1 g_i$ ☐

Now we have circuits for $\gamma_1(\sum_{i=1}^{d_1} g_i)$ and $\gamma_2(\sum_{i=1}^{d_2} h_i)$ in the case $\deg(g) > \deg(h)$. If $\deg(g) < \deg(h)$ then one can simply interchange left and right partial derivatives in the above lemmas and get circuits for $\gamma_1(\sum_{i=1}^{d_1} g_i)$ and $\gamma_2(\sum_{i=1}^{d_2} h_i)$. When $\deg(g) = \deg(h)$ cases we have circuits for $\gamma_1 g$ and $\gamma_2 h$ by Lemma 4. Now we explain how to compute the constant terms of the individual factors.

First we recall that given a monomial m and a non-commutative circuit C, the coefficient of m in C can be computed in deterministic polynomial time [AMS10]. We know that $f_0 = \alpha \cdot \beta$. We compute the coefficient of the monomial m_1 in the circuits $\gamma_1\gamma_2 gh, \gamma_1 g$, and in $\gamma_2 h$. Let those coefficients be a, b, c respectively. Moreover, we know that $\delta = \gamma_1\gamma_2$ is the coefficient of $(m_1\ m_2)$ in f which we can compute. Also, let the coefficient of m_1 in f be γ.

Now equating the coefficient of m_1 from both side of the equation $f = (g+\alpha)(h+\beta)$, we get the following expression

$$f_0 \cdot \frac{b}{\gamma_1} + \alpha \cdot (\frac{a}{\delta} - \gamma) + \alpha^2 \cdot \frac{c}{\gamma_2} = 0$$

On further simplification we obtain that

$$f_0 b\delta + (\alpha\gamma_1) \cdot (a - \gamma\delta) + c \cdot (\alpha\gamma_1)^2 = 0$$

One can think the above as a quadratic equation in A where $A = \alpha\gamma_1$ as

$$c \cdot A^2 + (a - \gamma\delta) \cdot A + f_0 b\delta = 0$$

By solving the above quadratic equation we get two solutions A_1 and A_2 for $\alpha\gamma_1$. Notice that $\gamma_2\beta = \frac{\delta f_0}{A}$. This clearly provides circuits for $(g+\alpha)$ and $(h+\beta)$ and the circuits depend on the value of A. Then we can run the PIT algorithm to decide the correct value of A that to be used.

Over $\mathbb{Q}$ we can just solve the quadratic equation in deterministic polynomial time using standard method. Over finite field $\mathbb{F} = \mathbb{F}_q$ were $q = p^r$, one can solve this problem deterministically in time $\mathrm{poly}(p, r)$ [vzGS92]. Using randomness, one can solve this problem in time $\mathrm{poly}(\log q)$ using Berlekamp's factoring algorithm [Ber71]. This also completes the proof of Theorem 2.

5 Conclusion

The result of Hrubes, Wigderson, and Yehudayoff [HWY10] shows an exponential circuit-size lower bound for an explicit polynomial in non-associative and commutative arithmetic circuit model. It will be very interesting to complement their result in PIT domain, i.e. design an efficient *white-box* polynomial identity testing algorithm for non-associative but commutative circuit model. To the best of our understanding, even a randomized polynomial-time algorithm is not known.

The PIT algorithm presented in this paper is *white-box*. Can one design an efficient *black-box* (even randomized) identity testing algorithm for non-associative, and non-commutative circuit model? Of course, for such an algorithm one should be allowed to evaluate the circuit over any algebra even non-associative. In particular, there is no known Amitsur-Levitzki type theorem [AL50] for this model which can be algorithmically useful, although the theory of non-associative algebra is well-studied in mathematics.

References

[AJR15] Vikraman Arvind, Pushkar S. Joglekar, and Gaurav Rattan, *On the complexity of non-commutative polynomial factorization*, Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II, 2015, pp. 38–49.

[AL50] Avraham Shimshon Amitsur and Jacob Levitzki, *Minimal identities for algebras*, Proceedings of the American Mathematical Society (1950), no. 4, 449–463.

[AM08] Vikraman Arvind and Partha Mukhopadhyay, *Derandomizing the isolation lemma and lower bounds for circuit size*, Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings, 2008, pp. 276–289.

[AMS10] Vikraman Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan, *New results on non-commutative and commutative polynomial identity testing*, Computational Complexity **19** (2010), no. 4, 521–558.

[AR16] Vikraman Arvind and S. Raja, *Some lower bound results for set-multilinear arithmetic computations*, Chicago J. Theor. Comput. Sci. **2016 (6)** (2016).

[Ber71] E. R. Berlekamp, *Factoring polynomials over large finite fields**, Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation (New York, NY, USA), SYMSAC '71, ACM, 1971, pp. 223–.

[HWY10] Pavel Hrubes, Avi Wigderson, and Amir Yehudayoff, *Relationless completeness and separations*, Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010, 2010, pp. 280–290.

[Hya77] Laurent Hyafil, *The power of commutativity*, 18th Annual Symposium on Foundations of Computer Science (FOCS), Providence, Rhode Island, USA, 31 October - 1 November 1977, 1977, pp. 171–174.

[Kal] Erich Kaltofen, *Factorization of polynomials given by straight-line programs*, Randomness in Computation **vol. 5 of Advances in Computing Research, pages 375–412. 1989**.

[KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka, *Equivalence of polynomial identity testing and polynomial factorization*, Computational Complexity **24** (2015), no. 2, 295–331.

[LMP16] Guillaume Lagarde, Guillaume Malod, and Sylvain Perifel, *Non-commutative computations: lower bounds and polynomial identity testing*, Electronic Colloquium on Computational Complexity (ECCC) **23** (2016), 94.

[Nis91] Noam Nisan, *Lower bounds for non-commutative computation (extended abstract)*, STOC, 1991, pp. 410–418.

[RS05] Ran Raz and Amir Shpilka, *Deterministic polynomial identity testing in non-commutative models*, Computational Complexity **14** (2005), no. 1, 1–19.

[SY10] Amir Shpilka and Amir Yehudayoff, *Arithmetic circuits: A survey of recent results and open questions*, Foundations and Trends in Theoretical Computer Science **5** (2010), no. 3-4, 207–388.

[vzGS92] Joachim von zur Gathen and Victor Shoup, *Computing frobenius maps and factoring polynomials*, Computational Complexity **2** (1992), 187–224.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il