Factorization of Polynomials Given By Arithmetic Branching Programs

Amit Sinhababu
Aalen University, Germany
amitks@cse.iitk.ac.in

Thomas Thierauf
Aalen University, Germany
thomas.thierauf@uni-ulm.de

Abstract

Given a multivariate polynomial computed by an arithmetic branching program (ABP) of size s, we show that all its factors can be computed by arithmetic branching programs of size $\text{poly}(s)$. Kaltofen gave a similar result for polynomials computed by arithmetic circuits. The previously known best upper bound for ABP-factors was $\text{poly}(s^{\log s})$.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Arithmetic Branching Program, Multivariate Polynomial Factorization, Hensel Lifting, Newton Iteration, Hardness vs Randomness

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.33

Funding Research supported by DFG grant TH 472/5-1.

Acknowledgements We thank Nitin Saxena, Pranjal Dutta, Arpita Korwar, Sumanta Ghosh, Zeyu Guo and Mrinal Kumar for helpful discussions. A.S would like to thank the Institute of Theoretical Computer Science at Ulm University for the hospitality.

1 Introduction

Polynomial factoring is a classical question in algebra. For factoring multivariate polynomials, we have to specify a model for representing polynomials. A standard model in algebraic complexity to represent polynomials are *arithmetic circuits* (aka *straight-line programs*). Other well known models are *arithmetic branching programs* (ABP), *arithmetic formulas*, *dense representations*, where the coefficients of *all* n-variate monomials of degree $\leq d$ are listed, or *sparse representations*, where only the non-zero coefficients are listed. Given a polynomial in some model, one can ask for efficient algorithms for computing its factors represented in the same model. That leads to the following question.

▶ **Question** (Factor size upper bound). *Given a polynomial of degree d and size s in a representation, do all of its factors have size* $\text{poly}(s, d)$ *in the same representation?*

For example in the dense representation the size of the input polynomial and the output factors is the same, namely $\binom{n+d}{d}$, for n-variate polynomials of degree d. But for other representations, the factor of a polynomial may take *larger* size than the polynomial itself. For example, in the sparse representation the polynomial $x^d - 1$ has size 2, but its factor $1 + x + \cdots + x^{d-1}$ has size d.

Arithmetic circuits. The algebraic complexity class VP contains all families of polynomials $\{f_n\}_n$ that have degree $\text{poly}(n)$ and arithmetic circuits of size $\text{poly}(n)$. Kaltofen [11] showed that VP is closed under factoring: Given a polynomial $f \in \text{VP}$ of degree d computed by an arithmetic circuit of size s, all its factors can be computed by an arithmetic circuit of size $\text{poly}(s, d)$.

Arithmetic branching programs. Kaltofen's [11] proof technique for circuit factoring does not directly extend to formulas or ABPs. The construction there results in a circuit, even if the input polynomial is given as a formula or an ABP. Converting a circuit to an arithmetic formula or an ABP *may* cause super-polynomial blow-up of size.

Analogous to VP, classes VF and VBP contain families of polynomials that can be computed by polynomial-size arithmetic formulas and branching programs, respectively. Note that the size also bounds the degree of the polynomials in these models. Arithmetic branching programs are an intermediate model in terms of computational power, between arithmetic formulas and arithmetic circuits,

$$\mathrm{VF} \subseteq \mathrm{VBP} \subseteq \mathrm{VP}\,.$$

ABPs are interesting in algebraic complexity as they essentially capture the power of linear algebra, for example they can efficiently compute determinants. ABPs have several equivalent characterizations. They can be captured via iterated matrix multiplication, weakly-skew circuits, skew circuits, and determinants of a symbolic matrices. See [14] for an overview of these connections.

Proof technique. A standard technique to factor multivariate polynomials has typically two main steps. The first step uses Hensel lifting to lift a factorization to high enough precision, starting from two coprime univariate factors. The second step, sometimes called the *jump step* or *reconstruction step*, consists of reconstructing a factor from a corresponding lifted factor by solving a system of linear equations.

The earlier works for polynomial factorization use a version of Hensel lifting, where in each iteration the lifted factors remain *monic*. It seems as this version is not efficient for ABPs. We observe that monicness of the lifted factor is not necessary for the jump step. This allows us to use a simple version of Hensel lifting that is efficient for ABPs.

Another point in some earlier works is that, in a pre-processing step, the input polynomial is transformed into a square-free polynomial. It is not clear how to achieve this transformation with small ABPs. We get around this problem by observing that square-freeness is not necessary. It suffices to have one irreducible factor of multiplicity one. This weaker transformation can be computed by small ABPs.

Finally, we use the fact that the determinant can be computed efficiently by ABPs.

▶ Remark 1. Whatever ABP we construct, the same can be done for circuits. Hence, as a by-product, we also literally provide another proof for the classical circuit factoring result of Kaltofen.

Comparison with prior works. There are several proofs of the closure of VP under factors [9, 10, 11, 1, 2, 12, 16, 6, 3]. None of the previous proofs directly extends to the closure of VBP, i.e. branching programs, under factors.

Recently, Dutta, Saxena, and Sinhababu [6] and also Oliveira [16] considered factoring in restricted models like formulas, ABPs and small depth circuits. They reduce polynomial factoring to approximating power series roots of the polynomial to be factored. Then they use versions of Newton iteration for approximating the roots. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$. If $p(\boldsymbol{x}, y)$ is the given polynomial and $q(\boldsymbol{x})$ is a root w.r.t. y, i.e. $p(\boldsymbol{x}, q(\boldsymbol{x})) = 0$, Newton iteration repeatedly uses the following recursive formula to approximate q:

$$y_{t+1} = y_t - \frac{p(\boldsymbol{x}, y_t)}{p'(\boldsymbol{x}, y_t)}.$$

If p is given as a circuit, the circuit for y_{t+1} is constructed from the circuit of y_t. For the circuit model, we can assume that $p(\boldsymbol{x}, y)$ has a single leaf node y where we feed y_t. But for formula and branching programs, we may have d many leaves labeled by y, where d is the degree of p in terms of y. As we cannot reuse computations in formula or branching programs, we have to make d copies of y_t in each round. This leads to $d^{\log d}$ blow-up in size.

Oliveira [16] used the idea of approximating roots via an approximator polynomial function of the coefficients of a polynomial. This gives good upper bound on the size of factors of ABPs, formulas, and bounded depth circuits under the assumption that the individual degrees of the variables in the input polynomial are bounded by a constant.

Recently, Chou, Kumar, and Solomon [3] proved closure of VP under factoring using Newton iteration for several variables for a system of polynomial equations. This approach also faces the same problem for the restricted models.

Instead of lifting roots, another classical technique for multivariate factoring is *Hensel lifting*, where factors modulo an ideal are lifted. Hensel lifting has a slow version, where the power of the ideal increases by one in each round. The other version due to Zassenhaus [21] is fast, the power of the ideal gets doubled in each round.

Kaltofen's [11, 10] proofs uses slow versions of Hensel lifting iteratively for d rounds, where d is the degree of the given polynomial. That leads to an exponential blow-up of size in models where the previous computations cannot be reused, as using previous lifts twice would need two copies each time. .

Kopparty, Saraf, and Shpilka [12] use the standard way of doing fast Hensel lifting for $\log d$ rounds, where in each round the lifted factors are kept monic. To achieve this, one has to compute a polynomial division with remainder. Implementing this version of Hensel lifting for ABPs or formulas seems to require to make d copies of previous computations in each round. Thus, that way would lead to a $d^{\log d}$ size blow-up. Also, they compute the gcd of polynomials, for which a priori no size upper bound was known for ABP or formulas.

Here, we use a classic version of fast Hensel lifting, that needs $\log d$ rounds and additionally in each round we have to make copies of previous computations only constantly many times. As we mentioned earlier, we avoid to maintain the monicness, and also gcd-computations.

Though various versions of Hensel lifting (factorization lifting) and Newton iteration techniques (root lifting) are equivalent in a mathematical sense [19], it is interesting that the former gives a better factor size upper bound for the model of ABP.

Application in hardness vs. randomness. Closure under factoring is used in the hardness vs. randomness trade-off results in algebraic complexity. See for example the excellent survey of Kumar and Saptharishi [13] for details on this topic. The celebrated result of Kabanets and Impagliazzo [8, Theorem 7.7] showed that a sufficiently strong lower bound for arithmetic circuits would *derandomize* polynomial identity testing (PIT). The proof of derandomization uses a hard polynomial as well as the upper bound on the size of factors of a polynomial computed by the circuit [11]. As a corollary of our result, we get a similar statement in terms of ABPs: An exponential (or super-polynomial) lower bound for ABPs for an explicit multilinear polynomial yields quasi-poly (or sub-exponential) black-box derandomization of PIT for polynomials in VBP.

Closure under factoring is relevant in the connection between algebraic complexity and proof complexity [7]. If a class $\mathcal{C}$ is closed under factoring, then the following holds. If a polynomial is *hard* for the class $\mathcal{C}$, then all its nonzero multiples are hard for $\mathcal{C}$. Lower bounds for all the nonzero multiples of an explicit hard polynomial may lead to lower bounds for ideal proof systems [7].

2 Preliminaries

We consider multivariate polynomials over a field $\mathbb{F}$ of characteristic 0. A polynomial p is called *square-free*, if for any non-constant irreducible factor q, the polynomial q^2 is not a factor of p.

By $\deg(p)$ we denote the total degree of p. Let x and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be variables and $p(x, \boldsymbol{z})$ be a $(n+1)$-variate polynomial. Then we can view p as a univariate polynomial $p = \sum_i a_i(\boldsymbol{z})\, x^i$ over $\mathbb{K}[x]$, where $\mathbb{K} = \mathbb{F}[\boldsymbol{z}]$. The *$x$-degree of p* is denoted by $\deg_x(p)$. It is the highest degree of x in p. Polynomial p is called *monic in x*, if the coefficient $a_{d_x}(\boldsymbol{z})$ is a nonzero constant, where $d_x = \deg_x(p)$.

By $\text{poly}(n)$ we denote the class of polynomials in $n \in \mathbb{N}$.

We denote by $\mathcal{I} = \langle x \rangle$ the ideal of polynomials generated by x over the ring $\mathbb{F}[x, \boldsymbol{z}]$. The *$k$-th power of the ideal $\mathcal{I}$* is the ideal $\mathcal{I}^k = \langle x^k \rangle$.

Computational models. An *arithmetic circuit* is a directed acyclic graph, whose leaf nodes are labeled by the variables $x_1, \ldots, x_n$ and various constants from the underlying field. The other nodes are labeled by sum gates or product gates. A node labeled by a variable or constant computes the same. A node labeled by sum or product compute the sum or product of the polynomials computed by nodes connected by incoming edges. The *size of an arithmetic circuit* is the total number of its edges.

An *arithmetic formula* is a special kind of arithmetic circuit. A formula has the structure of a directed acyclic tree. Every node in a formula has out-degree at most one. As we can not *reuse* computations in a formula, it is considered to be weaker than circuits.

An *arithmetic branching program* (ABP) is a layered directed acyclic graph with a single source node and a single sink node. An edge of an ABP is labeled by a variable or a constant from the field. The weight corresponding to a path from the source to the sink is the product of the polynomials labeling the edges on the path. The polynomial $f(x_1, \ldots, x_n)$ computed by the ABP is the sum of the weights of the all possible paths from source to sink.

The *size of an ABP* is the number of its edges. The size of the smallest ABP computing f is denoted by $\text{size}_{\text{ABP}}(f)$. The degree of a polynomial computed by an ABP of size s is at most $\text{poly}(s)$.

Properties of ABPs. *Univariate* polynomials have small ABPs. Let $p(x)$ be a univariate polynomial of degree d. It can be computed by an ABP of size $2d+1$, actually even by a formula of that size.

For univariate polynomials $p(x), q(x)$, the *extended Euclidian algorithm* computes the gcd $h = \gcd(p, q)$ and also the *Bézout-coefficents*, polynomials a, b such that $ap + bq = h$, where $\deg(a) < \deg(q)$ and $\deg(b) < \deg(p)$. Let p have the larger degree, $d = \deg(p) \geq \deg(q)$. Then clearly also $\deg(h), \deg(a), \deg(b) \leq d$, and consequently, all these polynomials, p, q, h, a, b have ABP-size at most $2d+1$.

Let $p(\boldsymbol{x}), q(\boldsymbol{x})$ be multivariate polynomials in $\boldsymbol{x} = (x_1, \ldots, x_n)$. For the ABP-size with respect to *addition* and *multiplication*, we have

1. $\text{size}_{\text{ABP}}(p+q) \leq \text{size}_{\text{ABP}}(p) + \text{size}_{\text{ABP}}(q)$,
2. $\text{size}_{\text{ABP}}(pq) \leq \text{size}_{\text{ABP}}(p) + \text{size}_{\text{ABP}}(q)$.

For the sum of two ABPs B_p, B_q one can put B_p and B_q in parallel by merging the two source nodes of B_p and B_q into one new source node, and similar for the two sink nodes. For the product, one can put B_p and B_q in series by merging the sink of B_p with the source of B_q.

Another operation is *substitution.* Let $p(x_1, \ldots, x_n)$ and $q_1(\boldsymbol{x}), \ldots, q_n(\boldsymbol{x})$ be polynomials. Let $\text{size}_{\text{ABP}}(q_i) \le s$, for $i = 1, \ldots, n$. Then we have

$$\text{size}_{\text{ABP}}(p(q_1, \ldots, q_n)) \le s \cdot \text{size}_{\text{ABP}}(p).$$

To get an ABP for $p(q_1(\boldsymbol{x}), \ldots, q_n(\boldsymbol{x}))$, replace an edge labeled x_i in the ABP B_p for p by the ABP B_{q_i} for q_i.

It is known that the *determinant of a symbolic matrix* of dimension n can be computed by an ABP of size $\text{poly}(n)$ [15]. By substitution, the entries of the matrix can itself be polynomials computed by ABPs.

Resultant. Given two polynomials $p(x, \boldsymbol{y})$ and $q(x, \boldsymbol{y})$ in variables x and $\boldsymbol{y} = (y_1, \ldots, y_n)$, consider them as polynomials in x with coefficients in $\mathbb{F}[\boldsymbol{y}]$. The *resultant of* p *and* q *w.r.t.* x, denoted by $\text{Res}_x(p, q)$, is the determinant of the Sylvester matrix of p and q. For the definition of the Sylvester matrix, see [20]. Note that $\text{Res}_x(p, q)$ is a polynomial in $\mathbb{F}[\boldsymbol{y}]$.

Basic properties of the resultant are that it can be represented as a linear combination of p and q, and that it provides information about the gcd of p and q.

▶ **Lemma 2** (See [20]). *Let $p(x, \boldsymbol{y})$ and $q(x, \boldsymbol{y})$ be polynomials of degree $\le d$ and $h = \gcd(p, q)$.*
1. $\deg(\text{Res}_x(p, q)) \le 4d^2$,
2. $\exists u, v \in \mathbb{F}[x, \boldsymbol{y}] \quad up + vq = \text{Res}_x(p, q)$,
3. $\text{Res}_x(p, q) = 0 \iff \deg_x(h) > 0$.

Note that the problem whether $\text{Res}_x(p, q) = 0$ is a polynomial identity test (PIT), because $\text{Res}_x(p, q) \in \mathbb{F}[\boldsymbol{y}]$. It can be solved in a randomized way by the DeMillo-Lipton-Schwartz-Zippel Theorem (see [4] and the references therein for more details and history of this theorem).

▶ **Theorem 3** (Polynomial Identity Test). *Let $p(\boldsymbol{x})$ be an n-variate nonzero polynomial of total degree d. Let $S \subseteq \mathbb{F}$ be a finite set. For $\boldsymbol{\alpha} \in S^n$ picked independently and uniformly at random,*

$$\Pr[\, p(\boldsymbol{\alpha}) = 0 \,] \;\le\; \frac{d}{|S|}\,.$$

3 Pre-processing Steps and Algebraic Tool Kit

Before we start the Hensel lifting process, a polynomial should fulfill certain properties that the input polynomial might not have. In this section, we describe transformations of a polynomial that achieve these properties such that ABPs can compute the transformation and its inverse, and factors of the polynomials are maintained.

We also explain how to compute homogeneous components and how to solve linear systems via ABPs. We show how handle the special case when the given polynomial is just a power of an irreducible polynomial.

3.1 Computing homogeneous components and coefficients of a polynomial

Let $p(x, \boldsymbol{z})$ be polynomial of degree d in variables x and $\boldsymbol{z} = (z_1, \ldots, z_n)$. Let B_p be an ABP of size s that computes a polynomial p. Write p as a polynomial in x, with coefficients from $\mathbb{F}[\boldsymbol{z}]$,

$$p(x, \boldsymbol{z}) = \sum_{i=0}^{d} p_i(\boldsymbol{z})\, x^i \,.$$

We show that all the coefficients $p_i(\boldsymbol{z})$ have ABPs of size $\mathrm{poly}(s, d)$.

The argument is similar to Strassen's *homogenization* technique for arithmetic circuits, an efficient way to compute all the homogeneous components of a polynomial. The same technique can be used for ABPs (see [17, Lemma 5.2 and Remark]). Here we sketch the proof idea.

Each node v of B_p we split into $d+1$ nodes $v_0, \ldots, v_d$, such that node v_i computes the degree i part of the polynomial computed by node v, for $i = 0, 1, \ldots, d$. Consider an edge e between node u and v in B_p.

- If e is labeled with a constant $c \in \mathbb{F}$ or a variable z_i, then we put an edge between u_i and v_i with label c or z_i, respectively.
- If e is labeled with variable x, then we put an edge between u_i and v_{i+1} with label 1.

The resulting ABP has one source node and $d+1$ sink nodes. The i-th sink node computes $p_i(\boldsymbol{z})$.

For each edge of B_p we get either d or $d+1$ edges in the new ABP. Hence, its size is bounded by $s(d+1)$.

▶ **Lemma 4** (Coefficient extraction)**.** *Let $p(x, \boldsymbol{z}) = \sum_{i=0}^{d} p_i(\boldsymbol{z})\, x^i$ be a polynomial. Then $\mathrm{size}_{\mathrm{ABP}}(p_i) \leq (d+1)\,\mathrm{size}_{\mathrm{ABP}}(p)$, for $i = 0, 1, \ldots, d$.*

The technique can easily be extended to constantly many variables. For two variables, consider $p(x, y, \boldsymbol{z}) = \sum_{i,j} p_{i,j}(\boldsymbol{z})\, x^i y^j$. Then, from an ABP of size s for p we get ABPs for the coefficients $p_{i,j}(\boldsymbol{z})$ of size $s(d+1)^2$ similarly as above.

In *homogenization*, we want to compute the homogeneous components of p. That is, write $p(\boldsymbol{z}) = \sum_{i=0}^{d} p_i(\boldsymbol{z})$, where $\deg(p_i) = i$. From an ABP B_p for p we get ABPs for the p_i's similarly as above: In the definition of the new edges, only for constant label, we put the edge from u_i to v_i. In case of any variable label z_j, we put the edge from u_i to v_{i+1} with label z_j. Then the i-th sink node computes $p_i(\boldsymbol{z})$. The size is bounded by $s(d+1)$.

▶ **Lemma 5** (Homogenization)**.** *Let $p(\boldsymbol{z}) = \sum_{i=0}^{d} p_i(\boldsymbol{z})$ be a polynomial with $\deg(p_i) = i$, for $i = 0, 1, \ldots, d$. Then $\mathrm{size}_{\mathrm{ABP}}(p_i) \leq (d+1)\,\mathrm{size}_{\mathrm{ABP}}(p)$, for $i = 0, 1, \ldots, d$.*

3.2 Computing q from $p = q^e$

A special case is when the given polynomial $p(\boldsymbol{x})$ is a power of one irreducible polynomial $q(\boldsymbol{x})$, i.e., $p = q^e$, for some $e > 1$. This case is handled separately. Kaltofen [10] showed how to compute q for circuits, ABPs, and arithmetic formulas. Here, we give a short proof from Dutta [5].

▶ **Lemma 6.** *Let $p = q^e$, for polynomials $p(\boldsymbol{x}), q(\boldsymbol{x})$. Then $\mathrm{size}_{\mathrm{ABP}}(q) \leq \mathrm{poly}(\mathrm{size}_{\mathrm{ABP}}(p))$.*

Proof. We may assume that p is nonzero; otherwise the claim is trivial. We want to apply Newton's binomial theorem to compute $q = p^{1/e}$. For this we need that $p(0, \ldots, 0) = 1$. If this is not the case, we first transform p as follows.

1. If $p(0, \ldots, 0) = 0$, let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ be a point where $p(\boldsymbol{\alpha}) \neq 0$. By the PIT-Theorem, a random point $\boldsymbol{\alpha}$ will work, with high probability. Now we shift the variables and work with the shifted polynomial $\tilde{p}(\boldsymbol{x}) = p(\boldsymbol{x} + \boldsymbol{\alpha})$.

   Still, $\tilde{p}(0, \ldots, 0)$ might be different from 1. In this case, we also apply the next item to $\tilde{f}$.

2. If $p(0,\ldots,0) = a_0 \neq 0$, then we work with $\tilde{p}(\boldsymbol{x}) = p(\boldsymbol{x})/a_0$. Then $\tilde{p}(0,\ldots,0) = 1$.
Note that both transformations are easily reversible. Hence, in the following we simply assume that $p(0,\ldots,0) = 1$.

By Newton's binomial theorem, we have

$$q = p^{1/e} = (1 + (p-1))^{1/e} = \sum_{i=0}^{\infty} \binom{1/e}{i} (p-1)^i \,. \tag{1}$$

Note that $p^{1/e}$ is a polynomial of degree $d = \deg(q)$. Since $p - 1$ is constant free, the terms $(p-1)^j$ in the RHS of (1) have degree $> d$, for $j > d$. Thus (1) turns into a finite sum modulo the ideal $\langle \boldsymbol{x} \rangle^{d+1}$,

$$q = \sum_{i=0}^{d} \binom{1/e}{i} (p-1)^i \bmod \langle \boldsymbol{x} \rangle^{d+1} \,. \tag{2}$$

Let $\text{size}_{\text{ABP}}(p) = s$. For the polynomial $Q = \sum_{i=0}^{d} \binom{1/e}{i} (p-1)^i$ from (2), we clearly have $\text{size}_{\text{ABP}}(Q) \leq \text{poly}(s)$. Finally, to get $q = Q \bmod \langle \boldsymbol{x} \rangle^{d+1}$, we have to truncate the terms in Q with degree $> d$. This can be done by computing the homogeneous components of Q as described in Lemma 5. We conclude that $\text{size}_{\text{ABP}}(q) \leq \text{poly}(s)$. ◀

3.3 Reducing the multiplicity of a factor

In the earlier works on bivariate and multivariate polynomial factoring, typically the problem is reduced to factoring a *square-free* polynomial. This is convenient at various places in the Hensel lifting process. The technique to reduce to the square-free case is via taking the gcd of the input polynomial and its derivative. However, for getting upper bounds on the ABP-size of the factors, we want to avoid gcd-computations, because no polynomial size upper bound for the gcd of two ABPs is known.

We avoid this problem by observing that we do not need the polynomial to be square-free. As we will see, it suffices to have one irreducible factor with multiplicity one, and another coprime factor.

Let $p(\boldsymbol{x})$ be the given polynomial, for $\boldsymbol{x} = (x_1, \ldots, x_n)$. The special case that p is a power of one irreducible polynomial we just handled in Section 3.2. Hence, we may assume that p has at least two irreducible factors. So let $p = q^e \, p_0$, where q is irreducible and coprime to p_0.

Consider the derivative of p w.r.t. some variable, say x_1.

$$\frac{\partial p}{\partial x_1} = q^{e-1} \left((e-1) \frac{\partial q}{\partial x_1} p_0 + q \frac{\partial p_0}{\partial x_1} \right) . \tag{3}$$

Note that q does not divide the factor $\left((e-1) \frac{\partial q}{\partial x_1} p_0 + q \frac{\partial p_0}{\partial x_1} \right)$ in (3). Hence, the multiplicity of factor q in $\frac{\partial p}{\partial x_1}$ is reduced by one compared to p.

For the ABP-size, we write p as a polynomial in x_1, i.e. $p(\boldsymbol{x}) = \sum_{i=0}^{d} a_i x_1^i$, where the coefficients a_i are polynomials in $x_2, \ldots, x_n$. By Lemma 4, when p has an ABP of size s, then the coefficients a_i can be computed by ABPs of size $s' = s(d+1)$. We observe that then the coefficients of the derivative polynomial $\frac{\partial p}{\partial x_1} = \sum_{i=1}^{d} i a_i x_1^{i-1}$ have ABPs of size $s' + 1$.

We repeat taking derivatives $k = e - 1$ times and get $\frac{\partial^k p}{\partial x_1^k}$, which has the irreducible factor q with multiplicity one, as desired.

The coefficients of $\frac{\partial^k p}{\partial x_1^k}$ can be computed by ABPs of size $s' + 1$. This yields an ABP of size $\text{poly}(s)$ that computes $\frac{\partial^k p}{\partial x_1^k}$.

3.4 Transforming to a monic polynomial

Given any polynomial $p(\boldsymbol{z})$ in variables $\boldsymbol{z} = (z_1, \ldots, z_n)$, there is a standard trick to make it monic in a new variable x by applying a linear transformation on the variables: for $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$, let

$$\tau_{\boldsymbol{\alpha}} : \quad z_i \mapsto \alpha_i x + z_i,$$

for $i = 1, \ldots, n$. Let $p_{\boldsymbol{\alpha}}(x, \boldsymbol{z})$ be the resulting polynomial. Note that p and $p_{\boldsymbol{\alpha}}$ have the same degree. We show that $p_{\boldsymbol{\alpha}}(x, \boldsymbol{z})$ is monic in x, for a random transformation $\tau_{\boldsymbol{\alpha}}$.

▶ **Lemma 7** (Transformation to monic). *Let $p(\boldsymbol{z})$ be polynomial of total degree d. Let $S \subseteq \mathbb{F}$ be a finite set. For $\boldsymbol{\alpha} \in S^n$ picked independently and uniformly at random,*

$$\Pr[\, p_{\boldsymbol{\alpha}}(x, \boldsymbol{z}) \text{ is monic in } x\,] \geq 1 - \frac{d}{|S|}.$$

Proof. Consider the terms of degree d in p. Let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ such that $|\boldsymbol{\beta}| = \sum_{i=1}^n \beta_i = d$. We denote the term $\boldsymbol{z}^{\boldsymbol{\beta}} = z_1^{\beta_1} \cdots z_n^{\beta_n}$. Then the homogeneous component of degree d in p can be written as $a_d(\boldsymbol{z}) = \sum_{|\boldsymbol{\beta}|=d} c_{\boldsymbol{\beta}} \boldsymbol{z}^{\boldsymbol{\beta}}$. Note that a_d is a nonzero polynomial.

Now consider the transformed polynomial $p_{\boldsymbol{\alpha}}$. We have $\deg_x(p_{\boldsymbol{\alpha}}) = d$ and the coefficient of x^d in $p_{\boldsymbol{\alpha}}$ is $a_d(\alpha) = \sum_{|\boldsymbol{\beta}|=d} c_{\boldsymbol{\beta}} \boldsymbol{\alpha}^{\boldsymbol{\beta}}$. When we pick $\boldsymbol{\alpha}$ at random, $a_d(\alpha)$ will be a nonzero constant with probability $\geq 1 - \frac{d}{|S|}$ by the PIT-Theorem, and in this case $p_{\boldsymbol{\alpha}}(x, \boldsymbol{z})$ is monic in x. ◀

Given an ABP of size s that computes $p(\boldsymbol{z})$, we can construct another ABP of size $3s$ that computes $p_{\boldsymbol{\alpha}}(x, \boldsymbol{z})$. For the new ABP replace edge labeled by z_i by the ABP computing $\alpha_i x + z_i$. For each old edge, this requires adding two new edges with labels α_i and x.

3.5 Handling the starting point of Hensel lifting

After doing the above pre-processing steps on the given polynomial $p(\boldsymbol{z})$, we call the transformed polynomial $f(x, \boldsymbol{z})$. We can assume that f of degree d can be factorized as $f = gh$, where g and h are coprime and g is irreducible. In the first step of Hensel lifting, we factorize the univariate polynomial $f(x, 0, \ldots, 0) \equiv f(x, \boldsymbol{z}) \pmod{\boldsymbol{z}}$. Now, clearly we have the factorization $f(x, 0, \ldots, 0) = g(x, 0, \ldots, 0)\, h(x, 0, \ldots, 0)$, but these two factors might not be coprime. In this case we do another transformation.

▶ Remark. Although it would suffice for our purpose to start with two coprime factors, the transformation below produces one irreducible factor.

Let g_0 be an irreducible factor of $g(x, 0, \ldots, 0)$. Then we have for some univariate polynomial $h_0'(x)$ and for $h_0(x) = h_0'(x)\, h(x, 0, \ldots, 0)$,

$$g \equiv g_0\, h_0' \pmod{\boldsymbol{z}},$$
$$f \equiv g_0\, h_0 \pmod{\boldsymbol{z}}.$$

We want that g_0 is coprime to h_0' and h_0. Directly, this might *not* be the case because all factors of $f(x, 0, \ldots, 0)$ might have multiplicity > 1. However, we argue how to ensure this after a random shift $\boldsymbol{\alpha}$ of f. That is, we consider the function $f(x, \boldsymbol{z} + \boldsymbol{\alpha})$

1. First, we show how to achieve that g_0 is coprime to h_0'.
   Since g is irreducible, it is also square-free, and hence, $\gcd(g, \frac{\partial g}{\partial x}) = 1$. By Lemma 2, the resultant $r(\boldsymbol{z}) = \mathrm{Res}_x(g, \frac{\partial g}{\partial x})$ is a polynomial of degree $\leq 4d^2$ and $r(\boldsymbol{z}) \neq 0$. Hence, at a

random point $\boldsymbol{\alpha} \in [8d^2]^n$, we have $r(\boldsymbol{\alpha}) \neq 0$ with high probability. At such a point $\boldsymbol{\alpha}$, we have that $g(x, \boldsymbol{\alpha})$ is square-free. Therefore, $g(x, \boldsymbol{z})$ is square-free modulo $(\boldsymbol{z} - \boldsymbol{\alpha})$, or, equivalently, $g(x, \boldsymbol{z} + \boldsymbol{\alpha})$ is square-free modulo $\boldsymbol{z}$. Hence, when we define g_0 and h_0' from $g(x, \boldsymbol{z} + \boldsymbol{\alpha})$ instead of $g(x, \boldsymbol{z})$, they will be coprime.

2. Similarly, we can achieve that g_0 is coprime to h_0. By the first item, it now suffices to get g_0 coprime to $h(x, 0, \dots, 0)$.
   For showing this, we use that g_0 is coprime to h_0' and prove that $g(x, 0, \dots, 0)$ is coprime to $h(x, 0, \dots, 0)$. Consider the resultant of g and h w.r.t. x, the polynomial $r'(\boldsymbol{z}) = \operatorname{Res}_x(g, h)$ has degree $\leq 4d^2$. Since g and h are coprime, $r'(\boldsymbol{z}) \neq 0$. Hence, at a random point $\boldsymbol{\alpha} \in [8d^2]^n$, we have $r'(\boldsymbol{\alpha}) \neq 0$ with high probability, and hence $g(x, \boldsymbol{\alpha})$ and $h(x, \boldsymbol{\alpha})$ are coprime univariate polynomials. Therefore, $g(x, \boldsymbol{z})$ and $h(x, \boldsymbol{z})$ are coprime modulo $(\boldsymbol{z} - \boldsymbol{\alpha})$, or, equivalently, $g(x, \boldsymbol{z} + \boldsymbol{\alpha})$ and $h(x, \boldsymbol{z} + \boldsymbol{\alpha})$ are coprime modulo $\boldsymbol{z}$.

Combining the two items, a random point $\boldsymbol{\alpha} \in [8d^2]^n$ will fulfill both properties with high probability. So instead of factoring $f(x, \boldsymbol{z})$, we do a coordinate transformation $\boldsymbol{z} \mapsto \boldsymbol{z} + \boldsymbol{\alpha}$ and factor $f(x, \boldsymbol{z} + \boldsymbol{\alpha})$ instead. From these factors, we easily get the factors of $f(x, \boldsymbol{z})$ by inverting the transformation. Note also that when $f(x, \boldsymbol{z})$ is monic in x, the same holds for $f(x, \boldsymbol{z} + \boldsymbol{\alpha})$.

In the next section, we do another transformation on the input polynomial. We apply a map on the variables that maps x to x and z_i is mapped to yz_i, for a new variable y and $i = 1, \dots, n$. Then we factorize the transformed polynomial modulo y. Note that in this case, going modulo y has the same effect of going modulo $\boldsymbol{z}$. So we can use the above argument to ensure the starting condition for Hensel lifting is satisfied.

3.6 Reducing multivariate factoring to the bivariate case

Factoring multivariate polynomials can be reduced to the case of *bivariate* polynomials (see [12]). Let x, y and $\boldsymbol{z} = (z_1, \dots, z_n)$ be variables and let $f(x, \boldsymbol{z})$ be the given polynomial. With $f \in \mathbb{F}[x, \boldsymbol{z}]$, we associate the polynomial $\widehat{f} \in \mathbb{F}[x, y, \boldsymbol{z}]$ defined by

$$\widehat{f}(x, y, \boldsymbol{z}) = f(x, yz_1, \dots, yz_n) \,.$$

The point now is to consider $\widehat{f}$ as a polynomial in $\mathbb{F}[\boldsymbol{z}][x, y]$, that is, as a bivariate polynomial in x and y with coefficients in $\mathbb{F}[\boldsymbol{z}]$. We list some properties.

1. $f(x, \boldsymbol{z}) = \widehat{f}(x, 1, \boldsymbol{z})$,
2. $\deg(\widehat{f}) \leq 2 \deg(f)$,
3. f monic in x $\implies$ $\widehat{f}$ monic in x,
4. $f = gh \implies \widehat{f} = \widehat{g}\,\widehat{h}$,
5. $\widehat{f} = g'\,h' \implies f = g'(x, 1, \boldsymbol{z})\, h'(x, 1, \boldsymbol{z})$.

By property 4, factors of f yield factors of $\widehat{f}$. The following lemma shows that also the irreducibility of the factors is maintained.

▶ **Lemma 8.** *Let f be monic in x and g be a non-trivial irreducible factor of f. Then $\widehat{g}$ is a non-trivial irreducible factor of $\widehat{f}$.*

Proof. By property 4 above, $\widehat{g}$ is a factor of $\widehat{f}$. We argue that $\widehat{g}$ is irreducible.

Let $\widehat{g} = uv$ be a factorization of $\widehat{g}$. By item 5 above, this yields a factorization of g as $g = u(x, 1, \boldsymbol{z})\, v(x, 1, \boldsymbol{z})$. Since g is monic in x, the same holds for $\widehat{g}$, and therefore also for factors u, v. Hence, either u or v must be constant, because otherwise they would provide a non-trivial factorization of g. ◀

Thus, to get an ABP for an irreducible factor g of f, first we show that there is an ABP for the irreducible factor $\widehat{g}$. This yields an ABP for g by substituting $g = \widehat{g}(x, 1, \boldsymbol{z})$.

Given an ABP B_f of size s for f, we get an ABP $B_{\widehat{f}}$ for $\widehat{f}$ by putting an edge labeled y in series with every edge labeled z_i in B_f, so that $B_{\widehat{f}}$ computes yz_i at every place where B_f uses z_i. Hence, the size of $B_{\widehat{f}}$ is at most $2s$.

3.7 Solving a linear system with polynomials as matrix entries

We show how to solve a linear system $M\boldsymbol{v} = 0$ for a polynomial matrix M with entries from $\mathbb{F}[\boldsymbol{z}]$ given as ABPs. We are seeking for a nonzero vector $\boldsymbol{v}$. Note that such a $\boldsymbol{v}$ exists over the ring $\mathbb{F}[\boldsymbol{z}]$ iff it exists over the field $\mathbb{F}(\boldsymbol{z})$.

Except for minor modifications, this follows from classical linear algebra. Kopparty, Saraf, and Shpilka [12, Lemma 2.6] have shown the same result for circuits. The proof works as well for ABPs.

▶ **Lemma 9** (Solving linear systems [12]). *Let $M = (m_{i,j}(\boldsymbol{z}))_{i,j}$ be a polynomial matrix of dimension $k \times m$ and variables $\boldsymbol{z} = (z_1, \ldots, z_n)$, where the entries are polynomials $m_{i,j} \in \mathbb{F}[\boldsymbol{z}]$ that can be computed by ABPs of size s.*

Then there is an ABP of size $\mathrm{poly}(k, m, s)$ computing a nonzero vector $\boldsymbol{v} \in \mathbb{F}[\boldsymbol{z}]^m$ such that $M\boldsymbol{v} = 0$ (if it exists).

Proof. After swapping rows of M, we ensure that the $j \times j$ submatrix M_j that consists of the first j rows and the first j columns has full rank, iteratively for $j = 1, 2, \ldots$.

For $j = 1$ this means to find a nonzero entry in the first column and swap that row with the first row. If the first column is a zero-column, then $\boldsymbol{v} = \begin{pmatrix}1 & 0 & \cdots & 0\end{pmatrix}^T$ is a solution and we are done. To extend from j to $j+1$, suppose we have ensured that M_j has full rank. Now we search for a row from row $j+1$ on, such that after a swap with row $j+1$, the submatrix M_{j+1} has full rank. This can be tested by Lemma 3. If no such row exists, then the process stops at j. If $j = m$ then M has full rank and the zero vector is the only solution. Otherwise, assume the above process stops with $j < m$.

Now Cramer's rule can be used to find the unique solution $\boldsymbol{u} = \begin{pmatrix}u_1 & u_2 & \cdots & u_j\end{pmatrix}^T$ of the system

$$M_j \boldsymbol{u} = \begin{pmatrix}m_{1,j+1} & m_{2,j+1} & \cdots & m_{j,j+1}\end{pmatrix}^T .$$

We have $u_i = \frac{\det M_j^i}{\det M_j}$, where M_j^i is the matrix obtained by replacing the i-th column of M_j by the vector $\begin{pmatrix}m_{1,j+1} & \cdots & m_{j,j+1}\end{pmatrix}^T$. Now, define

$$\boldsymbol{v} = \begin{pmatrix}\det M_j^1 & \det M_j^2 & \cdots & \det M_j^j & -\det M_j & 0 & \cdots & 0\end{pmatrix}^T .$$

Then $\boldsymbol{v}$ is a solution to the original system. Its entries are determinants of matrices with entries computed by ABPs of size s. Hence, all the entries of $\boldsymbol{v}$ have ABPs of size $\mathrm{poly}(k, m, s)$. ◀

4 Factors of Arithmetic Branching Programs

In this section, we prove that ABPs are closed under factoring.

▶ **Theorem 10.** *Let p be a polynomial over a field $\mathbb{F}$ with characteristic 0. For all factors q of p, we have*

$$\mathrm{size}_{\mathrm{ABP}}(q) \leq \mathrm{poly}(\mathrm{size}_{\mathrm{ABP}}(p)) .$$

We prove Theorem 10 in the rest of this section. First observe that it suffices to prove the poly(s) size upper bound for the irreducible factors of p. This yields the same bound for all the factors. The case when $p = q^e$ is proved in Section 3.2. So it remains to consider the general case when $p = p_1^{e_1} \cdots p_m^{e_m}$, for $m \geq 2$, where $p_1, \ldots, p_m$ are the different irreducible factors of p. We want to prove an ABP size upper bound for an irreducible factor, say p_1.

We start by several transformations on the input polynomial $p(\boldsymbol{z})$, where $\boldsymbol{z} = (z_1, \ldots, z_n)$.

1. As described in Section 3.3, taking $k = e_1 - 1$ times the derivative w.r.t. some variable, say z_1, we get the polynomial $p'(\boldsymbol{z}) = \frac{\partial^k p(\boldsymbol{z})}{\partial z_1^k}$, where the factor p_1 has multiplicity 1.
2. Next, by Lemma 7, we transform $p'(\boldsymbol{z})$ to a polynomial $p''(x, \boldsymbol{z})$ that is monic in x, for a new variable x. Thereby also the factors of $p'(\boldsymbol{z})$ are transformed, maintaining their irreducibilty and multiplicity. The degree of p'' is twice the degree of p'.
3. At this point, we may have to shift the variables $\boldsymbol{z}$ as described in Section 3.5 to ensure the properties needed for starting the Hensel lifting. This shift preserves the monicness and the irreducibility of the factors.
4. Finally, the transformation to a bivariate polynomial is explained in Section 3.6. This yields polynomial $p'''(x, y, \boldsymbol{z})$, with new variable y and monic in x. We rewrite p''' as a polynomial in x and y with coefficients in the ring $\mathbb{K} = \mathbb{F}[\boldsymbol{z}]$ and call the representation f. That is, $f(x, y) \in \mathbb{K}[x, y]$. By Lemma 8, the tranformation maintains irreducible factors. Note also that by the definition of p''' we have $f(x, 0) = p'''(x, 0, 0, \ldots, 0) = f(x, y) \bmod y$, so that $f(x, y) \bmod y$ is univariate.

The main part now is to factor $f(x, y) \in \mathbb{K}[x, y]$, say $f = gh$, where $g \in \mathbb{K}[x, y]$ is irreducible and coprime to $h \in \mathbb{K}[x, y]$, and f, g, h are monic in x and have x-degree ≥ 1. Let d be the total degree of f in x, y.

From the factor g of f, we will recover the factor f_1 of p by reversing the above transformations. We show that g can be computed by an ABP of size poly(s). It follows that the irreducible factor f_1 has an ABP of size poly(s).

The basic strategy is to first factor the univariate polynomial $f \bmod y$, and then apply Hensel lifting to get a factorization of $f \bmod y^t$, for large enough t. Finally, from the lifted factors modulo y^t, we compute the absolute factors of f.

4.1 Hensel lifting

There are various versions of Hensel lifting in the literature (see for example [18]). In our case, an ABP should be able to perform several iterations of the lifting. Therefore we use the lifting in a way suitable for ABPs. In particular, in contrast to other presentations, we will *not* maintain the monicness of the lifted factors.

Hensel lifting works over rings $\mathcal{R}$ modulo an ideal $\mathcal{I} \subseteq \mathcal{R}$. In our case, $\mathcal{R} = \mathbb{K}[x, y]$, where $\mathbb{K} = \mathbb{F}[\boldsymbol{z}]$, and $\mathcal{I} = \langle y \rangle^t$, for some $t \geq 1$.

▶ **Definition 11** (Lifting). *Let $\mathcal{R}$ be a ring and $\mathcal{I} \subseteq \mathcal{R}$ be an ideal. Let $f, g, h, a, b \in R$ such that $f \equiv gh \pmod{\mathcal{I}}$ and $ag + bh \equiv 1 \pmod{\mathcal{I}}$. Then we call $g', h' \in \mathcal{R}$* a lift of g, h with respect to f*, if*

(i) $f \equiv g'h' \pmod{\mathcal{I}^2}$,

(ii) $g' \equiv g \pmod{\mathcal{I}}$ *and* $h' \equiv h \pmod{\mathcal{I}}$*, and*

(iii) $\exists a', b' \in \mathcal{R} \quad a'g' + b'h' \equiv 1 \pmod{\mathcal{I}^2}$.

▶ Remark. The three conditions in Definition 11 are the *invariants* when iterating the lifting.

Note that the last condition is actually redundant. It follows from the assumptions together with the second condition. This can be seen in the proof of Lemma 12 below, where a lift g', h' from g, h is constructed, together with a', b'. When we show that condition (iii) holds, we do *not* use the specific form of g', h' constructed there, it suffices to have condition (ii).

▶ **Lemma 12** (Hensel Lifting). *Let $\mathcal{R}$ be a ring and $\mathcal{I} \subseteq \mathcal{R}$ be an ideal. Let $f, g, h, a, b \in R$ such that $f \equiv gh \pmod{\mathcal{I}}$ and $ag + bh \equiv 1 \pmod{\mathcal{I}}$. Then we have:*

1. (Existence). *There exists a lift g', h' of g, h w.r.t. f.*
2. (Uniqueness). *For any other lift g^*, h^* of g, h w.r.t. f, there exists a $u \in \mathcal{I}$ such that*

$$g^* \equiv g'(1+u) \pmod{\mathcal{I}^2} \quad \textit{and} \quad h^* \equiv h'(1-u) \pmod{\mathcal{I}^2}.$$

Proof. We first show the existence part. Let

1. $e = f - gh$,
2. $g' = g + be$ and $h' = h + ae$,
3. $c = ag' + bh' - 1$,
4. $a' = a(1-c)$ and $b' = b(1-c)$.

We verify that g', h' is a lift of g, h. Because $f \equiv gh \pmod{\mathcal{I}}$, we have $e = f - gh \equiv 0 \pmod{\mathcal{I}}$. In other words, $e \in \mathcal{I}$. It follows that $g' \equiv g \pmod{\mathcal{I}}$ and $h' \equiv h \pmod{\mathcal{I}}$.

Next we show that $f \equiv g'h' \pmod{\mathcal{I}^2}$.

$$\begin{aligned} f - g'h' &= f - (g+be)(h+ae) \\ &= f - gh - e(ag+bh) - abe^2 \\ &\equiv e - e(ag+bh) \pmod{\mathcal{I}^2} \\ &\equiv e(1-(ag+bh)) \pmod{\mathcal{I}^2} \\ &\equiv 0 \pmod{\mathcal{I}^2} \end{aligned}$$

In the second line, note that $e^2 \in \mathcal{I}^2$. The last equality holds because $e \in \mathcal{I}$ and $1-(ag+bh) \in \mathcal{I}$.

Now, we verify that $a'g' + b'h' \equiv 1 \pmod{\mathcal{I}^2}$. First, observe that

$$\begin{aligned} c &= ag' + bh' - 1 \\ &\equiv ag + bh - 1 \pmod{\mathcal{I}} \\ &\equiv 0 \pmod{\mathcal{I}} \end{aligned}$$

Hence, $c \in \mathcal{I}$ and we conclude that $a' \equiv a \pmod{\mathcal{I}}$ and $b' \equiv b \pmod{\mathcal{I}}$. Now,

$$\begin{aligned} a'g' + b'h' - 1 &= a(1-c)g' + b(1-c)h' - 1 \\ &= ag' + bh' - 1 - c(ag' + bh') \\ &= c - c(ag' + bh') \\ &= c(1-(ag' + bh')) \\ &\equiv 0 \pmod{\mathcal{I}^2} \end{aligned}$$

The last equality holds because $c \in \mathcal{I}$ and $1 - (ag' + bh') \equiv -c \equiv 0 \pmod{\mathcal{I}}$.

For the uniqueness part, let g^*, h^* be another lift of g, h. Let $\alpha = g^* - g'$ and $\beta = h^* - h'$. By Definition 11 (*ii*), we have $g' \equiv g \equiv g^* \pmod{\mathcal{I}}$ and $h' \equiv h \equiv h^* \pmod{\mathcal{I}}$, and therefore $\alpha, \beta \in \mathcal{I}$.

We first show

$$\beta g' + \alpha h' \equiv 0 \pmod{\mathcal{I}^2}. \tag{4}$$

$$\begin{aligned} \beta g' + \alpha h' &= \beta g' + (g^* - g')h' \\ &= \beta g' + g^*h' - g'h' \\ &\equiv \beta g' + g^*h' - g^*h^* \pmod{\mathcal{I}^2} \\ &\equiv \beta g' - \beta g^* \pmod{\mathcal{I}^2} \\ &\equiv -\alpha\beta \pmod{\mathcal{I}^2} \\ &\equiv 0 \pmod{\mathcal{I}^2} \end{aligned}$$

Define $u = a'\alpha - b'\beta$. Because $\alpha, \beta \in \mathcal{I}$, also $u \in \mathcal{I}$. Then, by (4) and because $a'g' + b'h' \equiv 1 \pmod{\mathcal{I}^2}$, we have

$$\begin{aligned}
g'(1+u) &= g'(1 + (a'\alpha - b'\beta)) \\
&= g' + a'g'\alpha - b'g'\beta \\
&\equiv g' + a'g'\alpha + b'h'\alpha \pmod{\mathcal{I}^2} \\
&\equiv g' + \alpha \pmod{\mathcal{I}^2} \\
&\equiv g^* \pmod{\mathcal{I}^2}.
\end{aligned}$$

Similarly, we get $h^* \equiv h'(1-u) \pmod{\mathcal{I}^2}$. ◀

For the ABP-size, recall that the size just adds up when doing additions or multiplications. Hence, when f, g, h, a, b have ABPs of size $\leq s$ and we construct ABPs for g', h', a', b' according to steps 1 - 4 in the above proof, then we get ABPs of size $O(s)$.

▶ Remark. In the *monic version* of Hensel Lifting there is a division in addition to the 4 steps from above. When we assume that g is monic, we can compute polynomials q and r such that $g' - g = qg + r$, where $\deg_x(r) < \deg_x(g)$. Then one can show that $\hat{g} = g + r$ and $\hat{h} = h'(1+q)$ are a lift of g, h w.r.t. f. Moreover, when g, h are monic, so are $\hat{g}, \hat{h}$. Also the Bézout-coefficients $\hat{a}, \hat{b}$ can be computed. For $\hat{c} = a\hat{g} + b\hat{h} - 1$, let $\hat{a} = a(1-\hat{c})$ and $\hat{b} = b(1-\hat{c})$.

The advantage of the monic version is that the result is really unique. There is no $1+u$ factor between monic lifts. The disadvantage is the extra division which would blow up the ABP-size too much.

4.2 Iterating Hensel lifting

Let $f = gh$, where g is irreducible and coprime to h, and f, g, h are monic in x with x-degree ≥ 1.

To start the Hensel lifting procedure, we factor the univariate polynomial $f(x,0) = f \bmod y$ as $f(x,0) = g_0(x)\, h_0(x)$, where g_0 is a divisor of $g \bmod y$, and coprime to h_0, and $\deg_x(g_0) \geq 1$. Recall that by the pre-processing in Section 3.5, we may assume that there is such a decomposition of $f(x,0)$.

By the Euclidian algorithm, there are polynomials $a_0(x), b_0(x)$ such that $a_0 g_0 + b_0 h_0 = 1$. Hence, for $\mathcal{I}_0 = \langle y \rangle$, we have $a_0 g_0 + b_0 h_0 \equiv 1 \pmod{I_0}$ and initiate Hensel lifting with

$$f \equiv g_0 h_0 \pmod{\mathcal{I}_0}.$$

We iteratively apply Hensel lifting to g_0, h_0 as described in the proof of Lemma 12. Each time, the ideal gets squared. Let $\mathcal{I}_k = \mathcal{I}_0^{2^k}$. That is, we get polynomials g_k, h_k such that

$$f \equiv g_k h_k \pmod{\mathcal{I}_k},$$

and g_k, h_k is a lift of g_{k-1}, h_{k-1} w.r.t. f. The following lemma states that g_k divides g modulo $\mathcal{I}_k$, for all $k \geq 0$. In a sense, the g_k's approximate g modulo increasing powers of y.

▶ **Lemma 13.** *With the notation from above, for all $k \geq 0$ and some polynomial h'_k,*

$$g \equiv g_k h'_k \pmod{\mathcal{I}_k} \quad \text{and} \quad h_k \equiv h\, h'_k \pmod{\mathcal{I}_k}.$$

Moreover, g_k, h'_k is a lift of g_{k-1}, h'_{k-1} w.r.t. g and $\deg_x(h'_k) \leq \deg_x(h_k) = 2^{O(k)}$, for $k \geq 1$.

Proof. The proof is by induction on $k \geq 0$. For the base case, we have that g_0 divides g modulo I_0, as explained above. Thus, for some polynomial h_0' that is coprime to g_0, we have

$$g \equiv g_0 h_0' \pmod{\mathcal{I}_0},$$

Hence, we have $h_0 \equiv h_0' h \pmod{\mathcal{I}_0}$. Note that h_0' can be just 1.

For the inductive step, assume that

$$g \equiv g_{k-1} h_{k-1}' \pmod{\mathcal{I}_{k-1}} \quad \text{and} \quad h_{k-1} \equiv h\, h_{k-1}' \pmod{\mathcal{I}_{k-1}}. \tag{5}$$

Let g_k', h_k'' be a lift of g_{k-1}, h_{k-1}' w.r.t. g, so that in particular

$$g_k' h_k'' \equiv g \pmod{\mathcal{I}_k}. \tag{6}$$

We claim that then $g_k',\ h\, h_k''$ is a lift of $g_{k-1},\ h\, h_{k-1}'$, i.e., of g_{k-1}, h_{k-1} by (5), w.r.t. f.

▷ Claim 14. $g_k',\ h\, h_k''$ is a lift of g_{k-1}, h_{k-1} w.r.t. f.

Proof. We check the three conditions for a lift in Definition 11. For the product condition (i), we have by (6)

$$g_k'\, h\, h_k'' = (g_k'\, h_k'')\, h \equiv gh \pmod{\mathcal{I}_k}.$$

For condition (ii), we have $g_k' \equiv g_{k-1} \pmod{\mathcal{I}_{k-1}}$ by assumption and similarly

$$h\, h_k'' \equiv h\, h_{k-1}' \equiv h_{k-1} \pmod{\mathcal{I}_{k-1}}.$$

By the remark after Definition 11, the condition (iii) already follows now. This proves Claim 14. ◁

Recall that also g_k, h_k is a lift of g_{k-1}, h_{k-1}. Hence, by the uniqueness property of Hensel lifting, there is a $u \in \mathcal{I}_{k-1}$ such that

$$g_k' \equiv g_k\,(1+u) \pmod{\mathcal{I}_k} \quad \text{and} \quad h\, h_k'' \equiv h_k\,(1-u) \pmod{\mathcal{I}_k} \tag{7}$$

Now observe that we can move the factor $1+u$: we have that $g_k\,(1+u),\ h\, h_k''$ is a lift of g_{k-1}, h_{k-1}, then also $g_k,\ h\, h_k''\,(1+u)$ is a lift of g_{k-1}, h_{k-1}.

▷ Claim 15. $g_k,\ h\, h_k''\,(1+u)$ is a lift of g_{k-1}, h_{k-1} w.r.t. f.

Proof. We check the conditions for a lift in Definition 11. The first two of them are trivial: moving the factor $1+u$ clearly does not change the product. Because $u \in \mathcal{I}_{k-1}$ we still have the equality with the factors g_{k-1} and h_{k-1} modulo $\mathcal{I}_{k-1}$, respectively.

By the remark after Definition 11, the third condition already follows, but it also easy to check now:

Let $a, b \in \mathcal{R}$ such that $ag_k + bh_k \equiv 1 \pmod{\mathcal{I}_k}$. It follows by (7) that

$$ag_k + bh\, h_k''(1+u) \equiv ag_k + bh_k(1-u)(1+u) \equiv ag_k + bh_k(1-u^2) \equiv 1 \pmod{\mathcal{I}_k}.$$

This proves Claim 15. ◁

Now, define $h'_k = h''_k(1+u)$. Note that

$$h'_k \equiv h''_k \equiv h'_{k-1} \pmod{\mathcal{I}_{k-1}}. \tag{8}$$

By (7), we have

$$h\,h'_k \equiv h\,h''_k\,(1+u) \equiv h_k(1-u)(1+u) \equiv h_k \pmod{\mathcal{I}_k}. \tag{9}$$

By (9) we have

$$f = gh \equiv g_k h_k \equiv g_k h\,h'_k \pmod{\mathcal{I}_k}. \tag{10}$$

It follows from (10) that $gh \equiv g_k h'_k h \pmod{\mathcal{I}_k}$. Now we want to cancel h in the last equation and conclude that $g \equiv g_k h'_k \pmod{\mathcal{I}_k}$. This we can do because h is monic in x, it does not contain a factor y, i.e. $h \notin \mathcal{I}_0$. Hence, together with (8), we conclude that g_k, h'_k is a lift of g_{k-1}, h'_{k-1} w.r.t. g.

For the x-degree of h'_k, consider the equation $h_k \equiv h\,h'_k \pmod{\mathcal{I}_k}$. Since h is monic in x the highest x-degree term in the product $h\,h'_k$ will survive the modulo operation. Therefore $\deg_x(h'_k) \le \deg_x(h) + \deg_x(h'_k) = \deg_x(h_k)$.

To bound the degree of h_k observe that in each round of the Hensel lifting, the maximum possible degree is bounded by a constant multiple (≤ 5) of the maximum degree in the previous round. After k iterations, the degree of the lifted factors is therefore bounded by $2^{O(k)}$. ◀

For the ABP-size, we observed at the end of Section 4.1 that the size increases by a constant factor in one iteration. Hence, after k iterations, the size increases by a factor $2^{O(k)}$.

4.3 Factor reconstruction for ABP

We show how to get the absolute factor g of f from the lifted factor. This is called the *jump step* in Sudan's lecture notes [18]. The difference to the earlier presentations is that our lifted factor might not be monic.

Let $f = gh$, where f has degree d, factor g is irreducible and coprime to h, and f, g, h are monic in x. In the previous section, we started with a factorization $f \equiv g_0 h_0 \pmod{\mathcal{I}_0}$, where g_0 is irreducible and coprime to h_0. Moreover, $g \equiv g_0 h'_0 \pmod{\mathcal{I}_0}$, for some h'_0 such that $h_0 = h\,h'_0 \pmod{\mathcal{I}_0}$.

Then we apply Hensel lifting, say t-times, for some t to be determined below. By Lemma 13, we get a factorization $f \equiv g_t h_t \pmod{\mathcal{I}_t}$ such that

$$g \equiv g_t h'_t \pmod{\mathcal{I}_t}, \tag{11}$$

for some h'_t such that $h_t \equiv h\,h'_t \pmod{\mathcal{I}_t}$.

Equation (11) gives us a relation between the known g_t and the unknown g, via the unknown h'_t. We set up a linear system of equations to find a polynomial $\tilde{g} \in \mathbb{K}[x,y]$ that is monic in x and has minimal degree in x such that

$$\tilde{g} \equiv g_t \tilde{h} \pmod{\mathcal{I}_t}, \tag{12}$$

for some polynomial $\tilde{h}$. We give some more details to the linear system below, after the next lemma. Before, we show that $\tilde{g}$ is indeed the factor we were looking for, for large enough t.

▶ **Lemma 16.** *$\tilde{g} = g$, for $t \ge \log(4d^2+1)$.*

Proof. Consider the resultant $r(y) = \mathrm{Res}_x(g, \tilde{g})$. We show that $r(y) = 0$. Then it follows from Lemma 2 that g and $\tilde{g}$ share a common factor with positive x-degree. Since g is irreducible it must be a divisor of $\tilde{g}$. Since both of them are monic and have the same x-degree, we get equality $\tilde{g} = g$, up to constant multiples.

To argue that $r(y) = 0$, recall from Lemma 2 that the resultant can be written as $r(y) = ug + v\tilde{g}$, for some polynomials u and v. By (11) and (12), we have

$$ug + v\tilde{g} \equiv g_t(uh'_t + v\tilde{h}) \pmod{\mathcal{I}_t}$$

Consider g_t and $w = uh'_t + v\tilde{h}$ as polynomials in y with coefficients in x. Suppose $g_t = c_0(x) + c_1(x)y + \ldots + c_{d'}(x)y^{d'}$. By the properties of Hensel lifting, we have $g_t \equiv g_0 \pmod{\mathcal{I}_0}$, and therefore $c_0(x) = g_0(x)$. Recall that g_0 is non-constant, $\deg(g_0) \geq 1$.

Let $j \geq 0$ be the least power of y that appears in w and let its coefficient be $w_j(x)$. Suppose for the sake of contradiction that $j < 2^t$. Then the least power of y in $g_t w$ is also j, and its coefficient is $g_0(x)w_j(x)$, which is a nonzero polynomial in x.

The monomials present in $g_0(x)w_j(x)y^j$ cannot be canceled by other monomials in $g_t w$ because they have larger y-degree. It follows that $g_t w \bmod \mathcal{I}_t$ is not free of x. On the other hand, $r(y) \equiv g_t w \pmod{\mathcal{I}_t}$ and $r(y) \in \mathbb{K}[y]$ is a polynomial with no variable x. This is a contradiction.

We conclude that $j \geq 2^t$, which means that $w \equiv 0 \pmod{\mathcal{I}_t}$. Hence, we get $r(y) \equiv 0 \pmod{\mathcal{I}_t}$. Recall that $\deg(r) \leq 4d^2$. Now we choose $t \geq \log(4d^2 + 1)$. Then we can conclude that indeed $r(y) = 0$. ◀

Details for setting up the linear system. Equation (12) can be used to set up a homogeneous system of linear equations. For the degree bounds of the polynomials, let $d_x = \deg_x(g)$ and $d_y = \deg_y(g)$. We may assume that we know d_x and d_y. Let $D_x = \deg_x(g_t)$ and $D_y = 4d^2$, where $d = \deg(f)$. Let $D'_x = \deg_x(h_t)$. Recall from Lemma 13 that $\deg_x(\tilde{h}) \leq D'_x$. Let

$$g_t = \sum_{i \leq D_x, j \leq D_y} c_{i,j}\, x^i y^j,$$
$$\tilde{g} = x^{d_x} + \sum_{i < d_x, j \leq d_y} r_{i,j}\, x^i y^j,$$
$$\tilde{h} = \sum_{i \leq D'_x, j \leq D_y} s_{i,j}\, x^i y^j,$$

where the coefficients $c_{i,j}, r_{i,j}, s_{i,j}$ are polynomials in the variables $z_1, \ldots, z_n$. To ensure that $\tilde{g}$ is monic, we set the coefficient of x^{d_x} in $\tilde{g}$ to be 1.

Note that we have an ABP that computes g_t. Hence, there are ABPs for computing the coefficients $c_{i,j}$ of g_t by Lemma 4. The coefficients $r_{i,j}, s_{i,j}$ of $\tilde{g}$ and $\tilde{h}$ we treat as unknowns. Equation (12) now becomes

$$\sum_{i \leq d_x, j \leq d_y} r_{i,j} x^i y^j \equiv \sum_{i \leq D_x, j \leq D_y} c_{i,j} x^i y^j \sum_{i \leq D'_x, j \leq D_y} s_{i,j} x^i y^j \pmod{y^{2d^2+1}} \tag{13}$$

Now we equate the coefficients of the monomials $x^k y^l$ on both sides in (13). Then we get $(D_x + D'_x + 1)(D_y + 1)$ homogeneous linear equations in $d_x(d_y + 1) + (D'_x + 1)(D_y + 1)$ many unknowns $r_{i,j}$ and $s_{i,j}$. This system can be expressed in the form $M\boldsymbol{v} = 0$, for a matrix M and unknown vector $\boldsymbol{v}$.

By Lemma 9, an ABP can efficiently compute a solution vector $\boldsymbol{v}$ of polynomials from $\mathbb{F}[\boldsymbol{z}]$. Note that by (11), a nontrivial solution is guaranteed to exist. From $\boldsymbol{v}$, we get the coefficients $r_{i,j}$ of $\tilde{g}$, and hence of the factor g.

4.4 Size Analysis

We summarize the bound on the ABP-size of the factor computed. Given polynomial p of degree d_p and $\text{size}_{\text{ABP}}(p) = s$. We have seen that the pre-processing transformations yield a polynomial f of degree $d_f \leq 2d_p$ and $\text{size}_{\text{ABP}}(f) = \text{poly}(s)$. Then we do $t = \log(2d_f^2 + 1)$ iterations of Hensel lifting. The initial polynomials f_0, g_0, h_0 have ABP-size bound by $2d_f$. Hence, the polynomials after the last iteration have ABP-size bounded by $2^t \,\text{poly}(s) = \text{poly}(s, d_p) = \text{poly}(s)$.

From the lifted factor we construct the actual factor of f. This step involves solving a linear system. We argued that the resulting polynomial g has ABP-size $\text{poly}(s)$.

Finally, we reverse the transformations from the beginning and get a factor of p that has an ABP of size $\text{poly}(s)$. This finishes the proof of Theorem 10.

5 Applications

5.1 Root Finding

Given a polynomial $p \in \mathbb{F}[x, \boldsymbol{y}]$, the *root finding* problem asks for a polynomial $r \in \mathbb{F}[\boldsymbol{y}]$ such that $p(r(\boldsymbol{y}), \boldsymbol{y}) = 0$. By a lemma of Gauß, r is a root of p iff $x - r(\boldsymbol{y})$ is an irreducible factor of p. By Theorem 10, when p is given by an ABP, we get an ABP for $x - r(\boldsymbol{y})$. Setting $x = 0$ and inverting the sign gives an ABP for $r(\boldsymbol{y})$.

▶ **Corollary 17.** *The solutions of the root finding problem for a polynomial p given by an ABP can be computed by ABPs of size* $\text{poly}(\text{size}_{\text{ABP}}(p))$.

5.2 Hardness vs. Randomness

As an application of Theorem 10, we get that lower bounds for ABPs imply a black-box derandomization of polynomial identity tests (PIT) for ABPs, similar to the result of Kabanets and Impagliazzo [8, Theorem 7.7] for arithmetic circuits.

▶ **Theorem 18** (Hitting-set from hard polynomial)**.** *Let $\{q_m\}_{m\geq 1}$ be a multilinear polynomial family such that q_m is computable in time $2^{O(m)}$, but has no ABP of size $2^{o(m)}$. Then one can compute a hitting set for ABPs of size s in time $s^{O(\log s)}$.*

The proof is similar to the proof given by Kabanets and Impagliazzo [8, Theorem 7.7] for circuits. At one point, they invoke Kaltofen's factor result for circuits. This can be replaced now by Theorem 10 for ABPs. Finally, from a given ABP of size s with n variables, we can get another ABP of size $\text{poly}(s)$ and $\log n$ variables by replacing the original variables by a hitting set generator, n polynomials computed by small size ABPs. This final composition step also goes through for ABPs. We will give more details in the final version of the paper.

6 Conclusion and Open Problems

We prove that the class of polynomials computed by ABPs is closed under factors. As a direct corollary, we get that the gcd of two polynomials computed by small-sized ABPs has small ABP size.

Our proof seems not to extend to the model of arithmetic formulas. The bottleneck is the last step, as the determinant of a symbolic matrix $(x_{i,j})_{n\times n}$ may not have $\text{poly}(n)$ size formulas. One way to avoid computing the determinant is by making the lifted factor monic in each round of Hensel lifting. But the direct implementation of monic Hensel lifting leads to a quasi-poly blow-up of formula size because it involves polynomial division in each step. So the closure of formulas under factors remains an open problem.

If one could show that arithmetic formulas are *not* closed under factors, i.e. if some polynomial $f(x_1, \ldots, x_n)$ exists that requires formula of size $\geq n^{\log n}$, but has a nonzero multiple of formula-size poly(n), then, by our result, VF would be separated from VBP and by Kaltofen's result, VF would be separated from VP.

Besides arithmetic formulas, there are other models for which poly(s, d) upper bound on the size of factors are not known. For example, read-once oblivious arithmetic branching programs (ROABP) and constant depth arithmetic circuits.

References

1 Peter Bürgisser. The complexity of factors of multivariate polynomials. *Foundations of Computational Mathematics*, 4(4):369–396, 2004.

2 Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7 of *Algorithms and Computation in Mathematic*. Springer, 2013.

3 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure of VP under taking factors: a short and simple proof. Technical Report arXiv:1903.02366, arXiv, 2019. `arXiv:1903.02366`.

4 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure results for polynomial factorization. *Theory of Computing*, 15(13):1–34, 2019. `doi:10.4086/toc.2019.v015a013`.

5 Pranjal Dutta. Discovering the roots: Unifying and extending results on multivariate polynomial factoring in algebraic complexity. Master's thesis, Chennai Mathematical Institute, 2018.

6 Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. Discovering the roots: Uniform closure results for algebraic classes under factoring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1152–1165. ACM, 2018.

7 Michael A. Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. Proof complexity lower bounds from algebraic circuit complexity. In *Proceedings of the 31st Conference on Computational Complexity (CCC)*, page 32. LIPIcs, 2016.

8 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, pages 355–364. ACM, 2003.

9 Erich Kaltofen. Uniform closure properties of p-computable functions. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 330–337, 1986.

10 Erich Kaltofen. Single-factor Hensel lifting and its application to the straight-line complexity of certain polynomials. In *Proceedings of the 19th annual ACM Symposium on Theory of Computing (STOC)*, pages 443–452. ACM, 1987.

11 Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Randomness and Computation*, 5:375–412, 1989.

12 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *computational complexity*, 24(2):295–331, 2015.

13 Mrinal Kumar and Ramprasad Saptharishi. Hardness-randomness tradeoffs for algebraic computation. *Bulletin of EATCS*, 3(129), 2019.

14 Meena Mahajan. Algebraic complexity classes. In *Perspectives in Computational Complexity*, pages 51–75. Springer, 2014.

15 Meena Mahajan and V Vinay. Determinant: Old algorithms, new insights. *SIAM Journal on Discrete Mathematics*, 12(4):474–490, 1999.

16 Rafael Oliveira. Factors of low individual degree polynomials. *computational complexity*, 2(25):507–561, 2016.

17 Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. `https://github.com/dasarpmar/lowerbounds-survey/releases`, 2016.

18 Madhu Sudan. Algebra and computation. http://people.csail.mit.edu/madhu/FT98/course.html, 1998. Lecture Notes.
19 Joachim von zur Gathen. Hensel and Newton methods in valuation rings. *Mathematics of Computation*, 42(166):637–661, 1984.
20 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2013.
21 Hans Zassenhaus. On Hensel factorization, I. *Journal of Number Theory*, 1(3):291–311, 1969.