Deterministically Testing Sparse Polynomial Identities of Unbounded Degree [⋆]

Markus Bläser [a] Moritz Hardt [b,∗,1] Richard J. Lipton [c]
Nisheeth K. Vishnoi [d]

[a] *Saarland University, Saarbrücken, Germany.*

[b] *Princeton University, Princeton, NJ, USA.*

[c] *Georgia Institute of Technology, Atlanta, GA, USA.*

[d] *IBM India Research Lab, New Delhi, India.*

Abstract

We present two deterministic algorithms for the arithmetic circuit identity testing problem. The running time of our algorithms is polynomially bounded in s and m, where s is the size of the circuit and m is an upper bound on the number monomials with non-zero coefficients in its standard representation. The running time of our algorithms also has a logarithmic dependence on the degree of the polynomial but, since a circuit of size s can only compute polynomials of degree at most 2^s, the running time does not depend on its degree. Before this work, all such deterministic algorithms had a polynomial dependence on the degree and therefore an exponential dependence on s.

Our first algorithm works over the integers and it requires only black-box access to the given circuit. Though this algorithm is quite simple, the analysis of why it works relies on Linnik's Theorem, a deep result from number theory about the size of the smallest prime in an arithmetic progression. Our second algorithm, unlike the first, uses elementary arguments and works over any integral domains, but it uses the circuit in a less restricted manner. In both cases the running time has a logarithmic dependence on the largest coefficient of the polynomial.

Key words: theory of computation, polynomial identity testing, arithmetic circuits, derandomization

[⋆] This paper builds upon the conference paper [1] by two of the authors.
[∗] Corresponding author
Email addresses: `mblaeser@cs.uni-sb.de` (Markus Bläser), `mhardt@cs.princeton.edu` (Moritz Hardt), `rjl@cc.gatech.edu` (Richard J. Lipton), `nisheeth.vishnoi@gmail.com` (Nisheeth K. Vishnoi).
[1] Part of this work was done while the author visited Carnegie Mellon University

1 Introduction

Testing polynomial identities is a fundamental problem in the theory of computation. Informally, the problem is equivalent to testing whether a multivariate polynomial $f(x_1, \ldots, x_n)$ given in some representation is identical to the zero polynomial. Many algorithmic problems such as finding perfect matchings in graphs [2–4], checking the equivalence of read-once branching programs [5], multi-set equality testing [6] and primality testing [7] reduce to this problem. Equally many applications have been discovered in complexity theory; to only mention two celebrated results, both PSPACE=IP [8] and NP=PCP [9] involved polynomial identity testing as a subroutine.

The earliest work in testing polynomial identities is due to DeMillo and Lipton [10], Zippel [11] and Schwartz [12]. These results give efficient *randomized* algorithms for checking polynomial identities. Indeed, no n-variate polynomial of total degree d can vanish on more than half the points in a combinatorial cube of size $(2d)^n$. Moreover, we can efficiently sample from this set of points. This observation led to a very important question which is still open today: Are there efficient *deterministic* algorithms for checking polynomial identities? This problem has become a prominent representative for the more general question about the power of randomization in efficient computation; namely, does P=BPP?

To make the complexity of the polynomial identity testing problem precise, one needs to fix a representation of the polynomial which has to be tested for being identically zero or not. If the polynomial is given as a list of its coefficients, the problem is trivial. In the interesting case, one is given an implicit representation of the polynomial such as the determinant of a matrix or an arithmetic circuit computing it. We say an algorithm is given black-box access, if it is restricted to using the input representation as an oracle only, i.e., it may only query the value of the represented polynomial at specific points. In the case of black-box access, one can show that randomness is necessary for polynomial identity testing (a simple dimension lower bound is given in [13]). When it comes to arithmetic circuits, no such lower bounds are known. However, we only know of deterministic arithmetic circuit identity tests when we have an additional "promise" about the circuit or the polynomial that it represents.

A natural and well-studied promise is that the polynomial is sparse: the number of monomials in it with non-zero coefficients is upper bounded by a parameter m. This parameter, which could be exponential in the number of variables in general, would be a measure of sparsity of the polynomial and we allow the runtime of our identity tests to depend on it. Also, in this paper

hosted by Steven Rudich.

we will consider polynomials represented by arithmetic circuits. Note that an arithmetic circuit of size s can compute a polynomial of degree at most 2^s, even if it computes a sparse polynomial.

We present two deterministic algorithms for the identity testing problem whose running time is a polynomial in s and m. Crucially, the running time of our algorithms is logarithmic in the degree of the polynomial and, hence by the discussion above, is independent of the degree. Before this work, all such deterministic algorithms had a polynomial dependence on the degree.

1.1 Related Work

Interpolating sparse polynomials. Originally, Zippel [11] was interested in Sparse Polynomial Interpolation, a more difficult question that asks to recover the entire list of coefficients of the given polynomial. Clearly, the sparsity of the polynomial is crucial in order to obtain efficient algorithms for this problem. Zippel's work was followed by many results [14–18] in interpolation. These results essentially imply polynomial identity tests with a runtime polynomial in n, m and d where n denotes the number of variables of the input polynomial, m denotes its number of nonzero terms and d its degree. There is also some dependence on the characteristic of the underlying field and the bit size of the points queried varies. A survey of these results can be found in [19].

Randomized complexity. A sequence of results [20,13,7,19,21,22] has studied (and determined in several cases) how much randomness is required in testing polynomial identities when only black-box access to the polynomial is granted. Most of these results are not concerned with sparse polynomials in our sense, but Klivans and Spielman [19] give an upper bound of $O(\log(mnd))$ random bits where m is the number of nonzero terms of the given polynomial, d is the degree and n is the number of variables. Trivial deterministic simulation of this algorithm yields a runtime polynomial in m, n and d. However, as before, the polynomial dependence on d implies an exponential worst-case running time for arithmetic circuits.

In the case of finite fields, Karpinski and Shparlinski [23] give a randomized and a deterministic algorithm with runtimes polynomial in m, n and $\log d$. The main issue here is that their deterministic algorithm has a linear dependence on the characteristic of the field. This makes their algorithm inefficient in the case of finite fields of large prime order. Their randomized algorithm has a better dependence on the characteristic. However, it requires a large number of random bits, namely polynomial in m, n, $\log d$. For comparison, notice this is exponentially more than [19].

Arithmetic circuits. In recent years, there has been progress on deterministic identity testing for depth-3 circuits of constant top fan-in [24–26]. We can represent a polynomial with m terms by an arithmetic circuit of size $m + 1$ and depth 2. Hence, depth-3 circuits can be seen as a generalization of sparse polynomials. Previously, Agrawal [27] gave a "hitting set generator" against the class of depth-2 arithmetic circuits. This result implies a deterministic polynomial identity test for sparse polynomials. However, all of these results require the input degree to be polynomially bounded.

When it comes to general circuits, no deterministic sub-exponential time arithmetic circuit identity test is known. In fact, Kabanets and Impagliazzo [28] show that such an algorithm would imply non-trivial circuit lower bounds. Further connections along these lines were shown by Agrawal [27].

1.2 Our Results

We state our two results here informally and give a brief overview of the proof techniques used. For a multivariate polynomial $f(x_1, \ldots, x_n)$, let m be an upper bound on the number of monomials in it with non-zero coefficients, d be its maximum degree, and H be the largest coefficient in its standard representation (or the size of the field in the case of finite fields).

Theorem 1 *Given an arithmetic circuit of size s representing a multivariate polynomial f with integer coefficients, there is a deterministic algorithm which decides whether f is identically zero in time $O((ms \log H)^{O(1)})$. Furthermore, the only operation the algorithm performs on the circuit is to test whether it vanishes on a set of points.*

Theorem 2 *Given an arithmetic circuit of size s representing a multivariate polynomial f over any integral domain, there is a deterministic algorithm which decides whether f is identically zero in time $O((nm \log d)^2 s \log H)$. Furthermore, the only operation the algorithm performs on the circuit is to test whether f is identically zero modulo certain polynomials.*

Both the algorithms and their proofs are similar in spirit and we now briefly illustrate the key ingredients and the places where they differ. The formal description and proofs appear in Sections 3 and 4.

The first step in both the algorithms is to "convert" the multivariate polynomial $f(x_1, \ldots, x_n)$ to a univariate polynomial $g(X)$. This involves just a computation of a transformation from a query point $(a_1, \ldots, a_n)$ for f to a query point a for g. This is standard and can be done efficiently even when one is given black-box access to the polynomial. The key point about this transformation is that $f(a_1, \dots, a_n)$ is non-zero if and only if $g(a)$ is non-zero. For such a property to hold the degree of g must be at least $(d+1)^n - 1$ and, indeed, there is a transformation, namely $x_i \mapsto X^{(d+1)^{i-1}}$, which ensures that the degree of g is at most $d \sum_{i=1}^{n-1} (d+1)^{i-1} = (d+1)^n - 1$. Henceforth, we may assume that we are given a univariate polynomial. In this setting, our first algorithm can be summarized as follows.

Given a circuit computing $g(X)$ with integer coefficients, declare 'g is identically zero' if and only if $g(X)$ vanishes modulo p for sufficiently many prime numbers p.

To analyze this algorithm we need to show that there is a way to pick these primes so that a non-zero polynomial $g(X)$ does not vanish modulo many of them. There are two ways in which a non-zero $g(X) = \sum_{i=0}^{D} c_i X^i$ can vanish modulo a prime p. It may happen that some non-zero c_i is divisible by p. Since the number of distinct prime divisors of any integer can be at most logarithmic in its magnitude, the number of distinct prime divisors of all the coefficients is bounded $m \log H$. More interestingly, it may happen that some $X^i \equiv X^j$ where X takes value in $\{0, 1, \dots, p-1\}$. This, by Fermat's Little Theorem, implies that $(p-1)|(i-j)$. In general it is difficult to bound the number of primes p such that $p-1$ divides an integer, but here is where we appeal to Linnik's Theorem about the smallest prime in an arithmetic progression.

We pick p carefully from an arithmetic progression $\{kq + 1 | k \geq 1\}$, where q is also a prime. Now, whenever $(p-1)|(i-j)$, q serves as a prime factor for $(i-j)$ and this essentially allows us to upper bound the number of primes which will be "bad" for the algorithm mentioned above. The key questions that remain unanswered are why should there exist primes in arbitrary arithmetic progressions and why should we be able to find them efficiently? Dirichlet generalized the Prime Number Theorem to prove that every arithmetic progression in which the terms have gcd 1 must have the "right density" of primes. But his theorem does not say anything about how to efficiently find such a prime. This is implied by Linnik's Theorem which shows that the smallest prime in an arithmetic progression with difference q is bounded by $q^{O(1)}$. Hence, modulo some details, to complete the proof of Theorem 1 it suffices to pick sufficiently many primes from arithmetic progressions whose difference itself is a prime number. From the discussion above $(mn \log(d+1) + m \log H)^{O(1)}$ such primes suffice.

The second algorithm observes that if one is allowed a weaker kind of black-box access to the polynomial $g(X)$, it is no longer necessary to pick the primes in a clever manner as above. It also has the advantages that it works over any integral domain and it does not require hard number-theoretic results.

Given a circuit computing $g(X)$, declare 'g computes the zero polynomial' if and only if $g(X)$ is identically zero modulo $X^p - 1$ for sufficiently many prime numbers p.

It is easy to see that in this case, the only way a non-zero polynomial will become zero after reducing it modulo $X^p - 1$ is when $p|(i - j)$. Since, as above, the degree of g is upper bounded by $(d + 1)^n$, picking $O(mn \log d)$ prime numbers suffices.

The key difference between the two algorithms is in how we access the given circuit. The first algorithm is black-box in what we may call the *point query model* where we only evaluate the input representation at specific points. The second algorithm is black-box in what we may call the *polynomial query model*. Here we are allowed to specify polynomials $\sigma_1(X), \ldots, \sigma_n(X)$ and $h(X)$ and ask if $f(\sigma_1(X), \ldots, \sigma_n(X))$ is identically zero modulo $h(X)$. The cost of this operation is polynomial in the degree of h and the logarithm of the largest coefficient of f in the case of of infinite fields (or the logarithm of the size of the field in the finite field case). The polynomial query model is natural, since it can be implemented straightforwardly when the polynomial is represented by an arithmetic circuit.

A virtue of both algorithms is their conceptual simplicity. Furthermore, both algorithms have a logarithmic runtime dependence on the degree of the input polynomial. In the arithmetic circuit model, this implies that we do not need to place any restriction on the degree of the input. Even though sparse polynomials have been extensively studied, the natural case of unbounded degrees in the arithmetic circuit model has not been treated previously. Now we proceed to formal proofs. We start with some preliminaries in Section 2.

2 Preliminaries

In this paper, K denotes an integral domain. We will refer to the *degree* of a polynomial $f \in K[x_1, \ldots, x_n]$ as is its maximum degree, that is, the smallest integer d such that the exponent of every variable in f is bounded by d.

We consider division-free arithmetic circuits over K with fan-in 2 at each multiplication and addition gate. Multiplication and addition by constants is allowed. The size of a circuit is the number of multiplication and addition gates. Over fields of characteristic zero, we add the logarithm of the magnitude of each constant to the size of the circuit. We denote the class of such arithmetic circuits of size s by $\mathcal{C}(s)$. We let $\mathcal{C}(s, m, d)$ denote the restriction of $\mathcal{C}(s)$ to circuits that compute polynomials with at most m nonzero terms and degree bounded by d.

We use the notation $\tilde{O}(t)$ to suppress poly-logarithmic factors of t, i.e., $\tilde{O}(t) = O(t \cdot \log^{O(1)} t)$.

Multivariate to univariate reductions. As remarked earlier, our algorithms will be designed for univariate polynomials and extended to the multivariate case using the following (folk) substitution attributed to Kronecker.

Lemma 3 *Let $f \in K[x_1, \ldots, x_n]$ be a polynomial of degree at most d. Then the substitution $x_i \mapsto X^{(d+1)^{i-1}}$ maps f to a univariate polynomial $g \in K[X]$ of degree at most $(d+1)^n$ such that any two distinct monomials w and w' in f map to distinct monomials in g. In particular, if f is not identically zero in $K[x_1, \ldots, x_n]$, then g is not identically zero in $K[X]$.*

2.1 Number-Theoretic Preliminaries

Fact 4 *An integer $k \geq 1$ has at most $\log k$ distinct prime divisors.*

We will need the following variant of the Prime Number Theorem.

Theorem 5 *The k-th prime number is of order $\Theta(k \log k)$.*

Let $P(k)$ denote the smallest prime number in the arithmetic progression $\{jk + 1 \mid j \geq 1\}$. Linnik's Theorem gives an *unconditional* upper bound on $P(k)$.

Theorem 6 (Linnik's Theorem [29]) *There is a constant $L > 1$ (called Linnik's constant) such that $P(k) < k^L$ for every sufficiently large $k \geq k_0$.*

The best known value for L is 5.5 due to Heath and Brown. Schinzel, Sierpinski, and Kanold have conjectured the value to be 2. For a detailed discussion of these facts the reader is referred to the book by Ribenboim [30].

We are interested in the value of $P(q)$ where q itself is a prime number. Notice, it is not immediately clear whether or not two distinct primes $q_1 \neq q_2$, could have $P(q_1) = P(q_2)$. However, we can show that no more than L primes can map to the same smallest prime. In particular, if we take a set of $5t$ distinct primes $\{q_i \mid 1 \leq i \leq 5t\}$, then the set $\{P(q_i) \mid 1 \leq i \leq 5t\}$ has cardinality at least t.

Lemma 7 *Let $q_1 < \cdots < q_v$ and p be primes such that $P(q_i) = p$ for all $i \in \{1, \ldots, v\}$ and suppose $q_1 > k_0$ where k_0 is the constant from Linnik's Theorem above. Then, $v < L$.*

PROOF. By hypothesis, there is a positive integer r such that $p-1 = rq_1q_2\cdots q_v$. Hence, $q_1^v \leq \prod_{i=1}^{v} q_i < p$. As p is the smallest prime in the arithmetic progression $\{1+kq_1 \mid k \geq 1\}$, Linnik's Theorem implies $p < q_1^L$. Hence, $q_1^v < q_1^L$ and thus $v < L$. $\square$

3 Identity Testing via Primes on Arithmetic Progressions

In this section we prove Theorem 1. The key is the following lemma.

Lemma 8 *Let $f = \sum_{i=0}^{d} c_i X^i$ be a univariate polynomial such that each coefficient $c_i \in \mathbb{Z}$ with $|c_i| \leq H$. Suppose that at least one, but no more than m coefficients of f are nonzero. Then, there are less than $5\log H + m\log d$ distinct prime numbers q such that f vanishes on every point modulo $P(q)$.*

PROOF. Let i be the smallest index such that the coefficient $c_i \neq 0$.

Claim 9 *The set $\{q \text{ prime} \mid c_i \equiv 0 \mod P(q)\}$ has cardinality at most $5\log H$.*

This claim follows directly from Lemma 7 once we observe there are at most $\log c_i \leq \log H$ prime numbers q such that $c_i \equiv 0 \mod q$.

Claim 10 *The set $\{q \text{ prime} \mid X^j \equiv X^i \mod P(q) \text{ for some } j > i\}$ has cardinality less than $m\log d$.*

In order to prove Claim 10, suppose there is an index $j > i$ such that $X^i \equiv X^j \mod P(q)$. Equivalently, $(P(q)-1)|(j-i)$ (by Fermat's Little Theorem). Since $P(q)$ lies on the arithmetic progression $\{1+kq \mid k > 0\}$, we conclude $q|(j-i)$. However, $j-i \leq d$ and hence $j-i$ has at most $\log d$ prime divisors. On the other hand, there at most $m-1$ choices for $j > i$. Our second claim is proven.

It remains to note, whenever f vanishes modulo a prime number q, then q is contained at least one of the two sets above. $\square$

Algorithm 1
Input: *Arithmetic circuit C on n inputs, parameters $R, d \in \mathbb{N}$.*
Output: *Whether or not C computes the identically zero polynomial over $\mathbb{Z}$.*

```
(1) Determine the first $R$ distinct prime numbers $p_1, \ldots, p_R$.
(2) For all $p \in \{p_1, \ldots, p_R\}$ and all $k \in \{0, \ldots, p-1\}$, evaluate $C$ at the
    point $(r_1, \ldots, r_n)$ modulo $p$ where $r_j = k^{(d+1)^{j-1}}$.
(3) Output "C is zero" if all queries returned are zero, otherwise output "C
    is nonzero".
```

Theorem 11 *Given a circuit $C \in \mathcal{C}(s, m, d)$ over $\mathbb{Z}$, Algorithm 1 decides whether C computes the identically zero polynomial in time $(ms\log(d+1) + \log H)^{O(1)}$ where H is an upper bound on the absolute value of each coefficient computed by C. Moreover, the algorithm only evaluates C on inputs of bit length logarithmic in $m, s, \log d$ and $\log H$.*

PROOF. The claim is trivial if the circuit C computes the identically zero polynomial over $\mathbb{Z}$. So, suppose otherwise. We claim that Algorithm 1 will reject C for a suitable setting of the parameter R. To see this, let $C'(X)$ denote the circuit obtained from C by applying the Kronecker substitution (Lemma 3). Notice, C' computes a nonzero univariate polynomial of degree at most $(d+1)^n$. Therefore, it follows from Lemma 8 that there exists a prime number q among the first $r = 5\log H + m\log((d+1)^n) = O(\log H + mn\log(d+1))$ prime numbers such that $C'(X)$ does not vanish modulo $P(q)$. In this case, the query computed by Algorithm 1 for $p = P(q)$ and some $k \in \{0, \dots, p-1\}$ will return a nonzero value. By Theorem 6, we have $P(q) \le q^{5.5}$ and hence it suffices to set $R = r^{O(1)}$.

The claimed runtime follows easily. The largest point queried has bit size $\log p_R$ which is logarithmic in $m, n, \log d$ and $\log H$. We conclude by observing that $n \le s$. $\square$

Remark 12 *The analysis of the previous algorithm works even if instead of Linnik's Theorem we consider a bound on the average size of the smallest prime in an arithmetic progression. Baker and Harman [31] showed that Linnik's constant is less than 1.93 on average. This leads to improvements by polynomial factors in the runtime of our algorithm. We omit the details.*

4 A Simple Modular Arithmetic Approach

In this section we prove Theorem 2. Unlike in the previous section, the following easy lemma turns out to be sufficient.

Lemma 13 *Let $f \in K[X]$ be a nonzero univariate polynomial with at most m nonzero terms and degree bounded by d. Then there are less than $m\log d$ prime numbers p for which $f(X)$ is identically zero modulo $X^p - 1$.*

PROOF. Let $f(X) = \sum_{i=0}^{d} c_i X^i$. Fix the smallest index i for which $c_i \neq 0$. If $f(X)$ vanishes modulo $X^p - 1$, then there must exist an index $j > i$ with $c_j \neq 0$ such that $X^j \equiv X^i \mod X^p - 1$. By basic modular arithmetic, this condition is equivalent to $p | (j - i)$. But, $0 < j - i \le d$ and therefore $j - i$ has at most $\log d$ prime divisors (Fact 4). On the other hand, we only have $m-1$ different choices for the index j. $\square$

Testing if a polynomial $f(X)$ is identically zero modulo $X^k - 1$ is a standard operation that can be carried out when f is represented by an arithmetic circuit. Given the circuit, we can recursively compute f in its reduced form. That is, at each gate we carry out the given arithmetic operation and reduce the resulting polynomial modulo $X^k - 1$. In this fashion, we maintain a polynomial of degree at most $k-1$. Reducing the polynomial gate-wise is justified, since we are computing a ring homomorphism. Multiplying two degree $k-1$ polynomials can be done at the cost of $\tilde{O}(k)$ ring operations over any integral domain using Schönhage-Strassen multiplication. Hence, the entire operation requires $\tilde{O}(sk)$ ring operations where s is the size of the given circuit. For details the reader is referred to the book by v. z. Gathen and Gerhard [32].

Our algorithm is stated and analyzed next. It will perform the Kronecker substitution so as to apply to multivariate polynomials as well.

Algorithm 2
Input: *Arithmetic circuit C on n inputs, parameters $R, d \in \mathbb{N}$.*
Output: *Whether or not C computes the identically zero polynomial.*

```
(1) Determine R distinct prime numbers $p_1, \ldots, p_R$.
(2) For each $i \in \{1, \ldots, R\}$:
    (a) Construct the circuit $C_i(X) = C(X^{r_1}, X^{r_2}, \ldots, X^{r_n})$ where $r_j$ denotes
        the remainder of $(d+1)^{j-1}$ modulo $p_i$.
    (b) Test if $C_i(X) \equiv 0 \mod (X^{p_i} - 1)$.
(3) Output "C is zero" if (2b) holds for all $i \in \{1, \ldots, R\}$, otherwise output
    "C is nonzero".
```

Runtime Analysis Step 1 of our algorithm requires time $\tilde{O}(R)$ using a deterministic polynomial time primality test [33]. Computing the circuit C_i requires us to compute the remainder of $d+1$ modulo p. This can be done in time $O(\log d \log p)$ [32]. The remaining values $(d+1)^{j-1}$ modulo p for $1 \leq j \leq n$ can be computed with less than n multiplications modulo p. Since $p = \tilde{O}(R)$ by Theorem 5, Step (2a) costs time $\tilde{O}(Rn \log d)$. If s denotes the size of C, then the size of C_i need not be larger than $s' = s + O(n \log R)$. Hence, the test $C_i(X) \equiv 0 \mod (X^{p_i} - 1)$ in (2b) can be performed with $\tilde{O}(Rs')$ arithmetic operations as discussed earlier. This results in the overall count of $\tilde{O}(R^2(s + n \log R)) = \tilde{O}(R^2 s)$ arithmetic operations.

Theorem 14 *Given an arithmetic circuit $C \in \mathcal{C}(s, m, d)$ on n inputs, Algorithm 2 deterministically decides if C computes the identically zero polynomial at the cost of $\tilde{O}((nm \log d)^2 s)$ ring operations over any integral domain.*

PROOF. We claim that given a circuit $C \in \mathcal{C}(s,m,d)$ and the parameter setting $R = O(mn\log d)$, Algorithm 2 accepts C if and only if C computes the identically zero polynomial. Clearly, the algorithm accepts C, if it computes the identically zero polynomial. On the other hand, suppose C does not compute the identically zero polynomial. Let $C'(X) = C(X, X^{d+1}, \ldots, X^{(d+1)^{n-1}})$. By Lemma 3, C' computes a nonzero univariate polynomial with at most m nonzero terms and degree $(d+1)^n$. Hence, by Lemma 13, there are less than $R = O(mn\log d)$ prime numbers p for which $C'(X) \equiv 0 \mod X^p - 1$. It remains to observe, for all $i \in \{1,\ldots,R\}$, the circuits C' and C_i compute identical polynomials modulo $X^{p_i} - 1$. The runtime of our algorithm is dominated by the number of arithmetic operations for the given setting of R. $\square$

Notice, over fields of characteristic zero, the cost of an arithmetic operation depends on the size of the coefficients of the polynomial. For instance, we obtain the runtime $\tilde{O}\left((nm\log d)^2 s \log H\right)$ over the field of rational numbers where H is an upper bound on the largest coefficient. Over a finite field of order q, the runtime will be $\tilde{O}\left((nm\log d)^2 s \log q\right)$.

Remark 15 *It would be interesting to know whether our analysis can be improved. The authors are not aware of a polynomial f computed by a constant depth circuit for which the univariate polynomial $f(X, X^{d+1}, \ldots, X^{(d+1)^{n-1}})$ vanishes modulo $X^p - 1$ for more than a polynomial (in the size of the circuit) number of primes p. In fact, Agrawal [27] conjectured that a similar test does work for any constant depth circuit.*

References

[1] R. Lipton, N. Vishnoi, Deterministic identity testing for multivariate polynomials, in: Proc. SODA, ACM-SIAM, 2003, pp. 756–760.

[2] S. Chari, P. Rohatgi, A. Srinivasan, Randomness-optimal unique element isolation with applications to perfect matching and related problems, SIAM J. Comput. 24 (5) (1995) 1036–1050.

[3] L. Lovász, On determinants, matchings, and random algorithms, in: FCT, 1979, pp. 565–574.

[4] K. Mulmuley, U. V. Vazirani, V. V. Vazirani, Matching is as easy as matrix inversion, Combinatorica 7 (1) (1987) 105–113.

[5] M. Blum, A. K. Chandra, M. N. Wegman, Equivalence of free boolean graphs can be decided probabilistically in polynomial time, IPL 10 (1980) 80–82.

[6] M. Blum, S. Kannan, Designing programs that check their work, J. ACM 42 (1) (1995) 269–291.

[7] M. Agrawal, S. Biswas, Primality and identity testing via chinese remaindering, J. ACM 50 (4) (2003) 429–443.

[8] A. Shamir, IP = PSPACE, J. ACM 39 (4) (1992) 869–877.

[9] S. Arora, S. Safra, Probabilistic checking of proofs: a new characterization of NP, J. ACM 45 (1) (1998) 70–122.

[10] R. A. DeMillo, R. J. Lipton, A probabilistic remark on algebraic program testing, Inf. Process. Lett. 7 (4) (1978) 193–195.

[11] R. Zippel, Probabilistic algorithms for sparse polynomials, in: Proc. ISSAC, Springer-Verlag, Berlin, 1979, pp. 216–226.

[12] J. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, Journal of the ACM 27 (1980) 701–717.

[13] D. Lewin, S. Vadhan, Checking polynomial identities over any field: Towards a derandomization?, in: Proc. 30th STOC, ACM, 1998, pp. 438–437.

[14] M. Ben-Or, A deterministic algorithm for sparse multivariate polynomial interpolation, in: Proc. 29th STOC, ACM, 1988, pp. 301–309.

[15] D. Y. Grigoriev, M. Karpinski, M. F. Singer, Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields, SIAM J. Comput. 19 (6) (1990) 1059–1063.

[16] M. Clausen, A. Dress, J. Grabmeier, M. Karpinski, On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields, Theor. Comput. Sci. 84 (2) (1991) 151–164.

[17] D. Grigoriev, M. Karpinski, M. F. Singer, Computational complexity of sparse rational interpolation, SIAM J. Comput. 23 (1) (1994) 1–12.

[18] K. Werther, The complexity of sparse polynomial interpolation over finite fields, Applicable Algebra in Engineering, Communication and Computing 5 (1994) 91–103.

[19] A. Klivans, D. A. Spielman, Randomness efficient identity testing of multivariate polynomials, in: Proc. 33th STOC, ACM, 2001, pp. 216–223.

[20] Z.-Z. Chen, M.-Y. Kao, Reducing randomness via irrational numbers, in: Proc. 29th STOC, ACM, 1997, pp. 200–209.

[21] A. Bogdanov, Pseudorandom generators for low degree polynomials., in: Proc. 37th STOC, ACM, 2005, pp. 21–30.

[22] M. Bläser, M. Hardt, D. Steurer, Asymptotically optimal hitting sets against polynomials, in: Proc. 35th ICALP, Springer, 2008, pp. 345–356.

[23] M. Karpinski, I. Shparlinski, On some approximation problems concerning sparse polynomials over finite fields, Theor. Comput. Sci. 157 (2) (1996) 259–266.

[24] N. Kayal, N. Saxena, Polynomial identity testing for depth 3 circuits, in: Proc. 21st CCC, IEEE, 2006, pp. 9–17.

[25] Z. Dvir, A. Shpilka, Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits, SIAM J. Comput. 36 (5) (2007) 1404–1434.

[26] A. Shpilka, Interpolation of depth-3 arithmetic circuits with two multiplication gates, in: Proc. 39th STOC, ACM, 2007, pp. 284–293.

[27] M. Agrawal, Proving lower bounds via pseudo-random generators., in: Proc. 25th FSTTCS, Springer, 2005, pp. 92–105.

[28] V. Kabanets, R. Impagliazzo, Derandomizing polynomial identity tests means proving circuit lower bounds, Comput. Complex. 13 (1/2) (2004) 1–46.

[29] U. V. Linnik, On the least prime in an arithmetic progression. I. The basic theorem, Mat. Sbornik N. S. 15 (57) (1944) 139–178.

[30] P. Ribenboim, The New Book of Prime Number Records, Springer, New York, 1996.

[31] R. C. Baker, G. Harman, The Brun-Titchmarsh theorem on average, in: Analytic number theory, Vol. 1 (Allerton Park, IL, 1995), Vol. 138 of Progr. Math., Birkhäuser Boston, Boston, MA, 1996, pp. 39–103.

[32] J. V. Z. Gathen, J. Gerhard, Modern Computer Algebra, Cambridge University Press, New York, NY, USA, 2003.

[33] M. Agrawal, N. Kayal, N. Saxena, PRIMES is in P, Ann. of Math. (2) 160 (2) (2004) 781–793.