Electronic Colloquium on Computational Complexity, Report No. 18 (1999)

Reducing Randomness via Chinese Remaindering

Manindra Agrawal
Dept. of Computer Science
Indian Institute of Technology
Kanpur 208016, India
manindra@iitk.ac.in

Somenath Biswas
Dept. of Computer Science
Indian Institute of Technology
Kanpur 208016, India
sb@iitk.ac.in

Abstract

We give new randomized algorithms for testing multivariate polynomial identities over finite fields and rationals. The algorithms use $\lceil \sum_{i=1}^{n} \log(d_i + 1) \rceil$ (plus $\lceil \log \log C \rceil$ in case of rationals where C is the largest coefficient) random bits to test if a polynomial $P(x_1, \ldots, x_n)$ is zero where d_i is the degree of x_i in P and has an error probability of ϵ for any $\epsilon > 0$. The running time of the algorithms is polynomial in the size of arithmetic circuit representing the input polynomial and $1/\epsilon$. These algorithms use fewer random bits than all the known methods and also take an order of magnitude less time compared to some of the recently proposed methods [CK97, LV98]. Our algorithms first transform the input polynomial to a univariate polynomial and then uses Chinese remaindering over univariate polynomials to efficiently test if it is zero.

We also give a simple test for primality based on identity checking.

1 Introduction

Given a polynomial $P(x_1, \ldots, x_n)$ and a field F, to test if the polynomial is identically zero over F is an important problem with applications to matching [Lov79, MVV87, CRS95], read-once branching programs [BCW80], multi-set equality [BK95] etc. It is also used in the context of interactive and probabilistically-checkable proofs [LFKN90, Sha90, BFL90, AS92, ALM+92] and learning sparse multivariate polynomials [Zip79, GKS90, CDGK91, RB91].

The problem is trivial if the input polynomial P is given as a sum of terms—P is zero iff all the terms are zero. However, in general, P is given in some implicit form, e.g., a symbolic determinant, product of polynomials, straight-line programs, arithmetic circuits etc. Of these, arithmetic circuits is the most general model since all the other forms can be efficiently transformed to such circuits. Henceforth, we will assume that the polynomial P is given as an arithmetic circuits containing addition and multiplication gates with input being the n variables $x_1, \ldots, x_n$ (the circuit may also have constants from the given field F). We will also assume that the two field operations can be done efficiently. The degree of a variable in such a polynomial can be exponential in the size of the circuit, and value of coefficients (in case F is infinite) doubly exponential. We let d_i denote the degree of the variable x_i, $d = \max_i\{d_i\}$, and C the value of the largest coefficient.

Schwartz and Zippel [Sch80, Zip79] observed that if a random assignment for the variables of P is chosen from the set S^n, $S \subseteq F'$ ($F' = F$ if F is infinite, otherwise it may be a small extension of F) and P is evaluated on this assignment, then the probability that a non-zero P evaluates to zero is bounded by $\sum_{i=1}^{n} d_i/|S|$. By choosing $|S| \geq \sum_{i=1}^{n} d_i/\epsilon$ we get an efficient (since the given arithmetic circuit can be evaluated efficiently on the assignment) randomized test for the problem that errs with probability at most ϵ. Clearly, the number of random bits required by the algorithm is at least $n \cdot \log |S| = n \cdot (\log \sum_{i=1}^{n} d_i + \log \frac{1}{\epsilon})$ and the time taken is

ISSN 1433-8092

polynomial in the size of the circuit and $\log \frac{1}{\epsilon}$. (There is an additional parameter here: the size of the largest coefficient in case F is infinite, and size of the field F in case F is finite but we shall ignore it for the time being).

Recently, Chen and Kao [CK97] proposed a new paradigm for identity testing over the field $\mathcal{Q}$: they constructed for each x_i a set of $2^{\lceil \log(d_i+1) \rceil}$ *irrational* assignments such that P is zero iff it evaluates to zero on *any* of these assignments. As one cannot evaluate the polynomial on irrational values, they replaced the irrational values by suitable rational approximations and then showed that if a random assignment is picked for each x_i from its corresponding set, a non-zero P evaluates to zero on this assignment with probability inversely proportional to the bit-length of the approximations. Thus, the number of random bits needed in this test is exactly $\sum_{i=1}^{n} \lceil \log(d_i + 1) \rceil$ *irrespective* of the error probability. The time taken by the algorithm is a polynomial in n, d, and $\frac{1}{\epsilon}$ as the bit-length of the approximation is required to be at least $d + \frac{1}{\epsilon}$.

Lewin and Vadhan [LV98] generalized the above paradigm to work for identity testing over finite fields. Instead of choosing irrational values for variables in the polynomial, they chose square roots of irreducible polynomials over $F[x]$. These square roots are infinite formal power series and so they approximated these by truncating the power series at certain degree. They showed that similar properties hold for these assignments as well and obtained exactly the same bounds for the number of random bits used and the running time.

In comparison to the Schwartz-Zippel test, the Chen-Kao/Lewin-Vadhan tests require far fewer random bits, and their number is independent of the error parameter. However, these tests take more time than Schwartz-Zippel: the former take time polynomial in d and $\frac{1}{\epsilon}$, which is exponential in both the size of the arithmetic circuit representing the polynomial and $\log \frac{1}{\epsilon}$ as opposed to polynomial in the two parameters in the latter. While the second gap cannot be bridged without eliminating randomness altogether (if the time taken is polynomial in $\log \frac{1}{\epsilon}$ in Chen-Kao/Lewin-Vadhan then by choosing $\frac{1}{\epsilon}$ to be a large enough exponential one can completely de-randomize the algorithm as the number of random bits is independent of ϵ), the first gap is a little unsatisfactory. It means that the Chen-Kao/Lewin-Vadhan tests are inefficient even for a univariate identity with high degree.

In this paper, we make two contributions to the identity testing problem. First, we provide a new, and important, application of this problem: primality testing. We exhibit a univariate polynomial of degree n which is zero (over a suitable finite field) iff the number n is prime. This test is conceptually simpler than the existing tests for primality, e.g., Miller-Rabin [Mil76, Rab80], Solovay-Strassen [SS77], although not as efficient.

Our second contribution is yet another paradigm for identity testing: via Chinese remaindering over polynomials. The idea here is simple: first transform the given multivariate polynomial to a higher degree univariate polynomial such that the identity property is preserved. Then test if the univariate polynomial is zero using division by a small degree polynomial randomly chosen from a suitable set. Using this, we are able to obtain two algorithms—one for finite fields and the other for $\mathcal{Q}$—that require $\lceil \sum_{i=1}^{n} \log(d_i + 1) \rceil$ (plus $\lceil \log \log C \rceil$ if the field is $\mathcal{Q}$) random bits and work in time polynomial in the size of the arithmetic circuit and $\frac{1}{\epsilon}$. These algorithms bridge the aforementioned gap and thus are essentially optimal in both the parameters. Even the number of random bits required by the algorithms is smaller than Chen-Kao/Lewin-Vadhan (consider the case when $d_i = 2$ for each i: our algorithms require $O(n)$ fewer bits in this case). This fact is not immediately apparent when the field is $\mathcal{Q}$ since Chen-Kao test requires $\sum_{i=1}^{n} \lceil \log(d_i + 1) \rceil$ random bits as opposed to an additional term of $\lceil \log \log C \rceil$ in our algorithm. However, it is implicitly assumed in the Chen-Kao test that the coefficients of the input polynomial are polynomial sized whereas we allow the coefficients to be exponentially sized. If the coefficients are polynomially sized then our algorithm requires only $\lceil \sum_{i=1}^{n} \log(d_i + 1) \rceil$ random bits since the additional requirement of $\lceil \log \log C \rceil$ random bits can be eliminated by cycling through all possible values of these bits (there are only polynomially many such values).

The organization of the paper is as follows: section 2 defines the formal machinery used in the paper, section 3 gives the primality testing algorithm, section 4 gives the new identity testing algorithms, and section 5 discusses some de-randomization issues.

2 Formal Setting

Let F be a field. By $F[\alpha]$ we denote the field extension of F obtained by adjoining to F the algebraic element α. GF$[q]$ denotes a finite field of size q. $\mathcal{Q}$ denotes the field of rationals.

We represent polynomials as arithmetic circuits. An arithmetic circuit over a ring consists of input variables, constants, and addition and multiplication gates where the constants and arithmetic operations are from the underlying ring. The size of the circuit is the number of gates in it. It is easy to see that the degree of a polynomial represented by an arithmetic circuit can be exponential in the size of the circuit, and so can be the size of coefficients in case the underlying ring is integers/ rationals.

3 Primality testing

In this section, we exhibit a new primality test which is based on polynomial identity testing.

Let $\mathcal{P}(z) = (1+z)^n - 1 - z^n$.

Lemma 3.1 *$\mathcal{P}(z) = 0 (\mathrm{mod}\ n)$ iff n is prime.*

Proof. We can rewrite $\mathcal{P}$ as:

$$\mathcal{P}(z) = \sum_{j=1}^{n-1} \binom{n}{j} z^j.$$

Suppose n is prime. Then, $\binom{n}{j} = 0 (\mathrm{mod}\ n)$ for every j, $1 \leq j < n$ since n occurs in the numerator but not in the denominator of $\binom{n}{j}$. Therefore, $\mathcal{P}(z) = 0 (\mathrm{mod}\ n)$.

Suppose n is composite. Let p be a prime divisor of n. Consider $\binom{n}{p}$. Suppose p^a is the largest power of p that divides n. Then $\binom{n}{p}$ is divisible by p^{a-1} but *not* by p^a. Therefore, $\binom{n}{p} \neq 0 (\mathrm{mod}\ p^a)$ which implies $\binom{n}{p} \neq 0 (\mathrm{mod}\ n)$. Clearly, $1 < p < n$ and so $\mathcal{P}(z) \neq 0 (\mathrm{mod}\ n)$. ■

We cannot immediately invoke any of the identity testing algorithms to obtain a randomized polynomial-time algorithm for primality. This is because all the tests work over a field and the number n here may be composite. Also, the Lewin-Vadhan test would require time polynomial in n which is exponential in the input size. In the next section, we show how to make our test work for composite moduli.

Our test requires $\lceil \log n \rceil$ random bits, takes time polynomial in $\log n$ and $\frac{1}{\epsilon}$ (the exponent of $\log n$ is large, at least six), and errs with probability ϵ. In comparison, the Miller-Rabin test requires $\lceil \log n \rceil$ random bits, takes time $O((\log n)^3)$ and errs with probability $\frac{1}{4}$. This probability can be reduced at the cost of increasing the number of random bits slightly. Clearly, this test scores over our test in terms of practical utility (same is true for Solovay-Strassen test too). However, our test is conceptually simpler and has an easier proof of correctness.

4 Identity testing

In this section, we give our identity testing algorithms. The section is divided in four subsections. In the first one, we give the algorithm that works for univariate polynomials over any finite field.

In the next subsection we generalize the algorithm to work over multivariate polynomials. In the third subsection, we modify the algorithm to work for polynomials modulo composite numbers and in the final subsection we give the algorithm that works over rationals.

4.1 Univariate polynomials over finite fields

Let $P(x)$ be a univariate polynomial over finite field GF[q] of characteristic p. Let d be the degree of P. In this subsection operations will always be over the field GF[q] unless explicitly mentioned otherwise. The idea of testing the polynomial is simple: since we cannot multiply out the polynomial to check if all its coefficients are zero (this would require $d^{O(1)}$ steps), we use Chinese remaindering to test if the polynomial is zero. In other words, randomly choose a small degree polynomial $Q(x)$ and test if $P(x)$ is divisible by $Q(x)$. If P is non-zero, this test will fail with high probability. To make the test suceed with probability $1-\epsilon$, we need to choose the polynomial Q from a set of polynomials such that the lcm of any ϵ fraction of these polynomials has degree at least d. Also, to keep the number of random bits required to a minimum, we need to sample from this set using only $\log d$ random bits.

One possible way of doing this is to consider a set of d mutually co-prime polynomials of degree $\frac{1}{\epsilon}$ each (or more strongly, a set of d irreducible polynomials of the same degree). However, sampling from such a set using only $\log d$ random bits appears difficult as there is no known efficient way of constructing a large number of mutually co-prime polynomials[1]. Therefore, we will simply construct a set of d small degree polynomials satisfying the required property though the polynomials in the set may not be mutually co-prime.

For a prime number r, let

$$Q_r(z) = \sum_{i=0}^{r-1} z^i,$$

(Q_r is the r^{th} cyclotomic polynomial). Let $\ell = \lceil \log d \rceil$. For a binary vector $s \in [0,1]^\ell$ and number $t \geq \ell$, let

$$A_{s,t}(x) = x^t + \sum_{i=0}^{\ell-1} s[i] \cdot x^i,$$

where $s[i]$ is the i^{th} component of s. Let

$$T_{r,s,t}(x) = Q_r(A_{s,t}(x)).$$

Our set of polynomials will consist of $T_{r,s,t}$ for all values of s and suitably fixed values of r and t. We first prove that this set has the required property in the following three lemmas.

Lemma 4.1 *Let r be any prime number, $r \neq p$, such that r does not divide any of $q-1$, q^2-1, ..., $q^{\ell-1}-1$. Then the polynomial $T_{r,s,t}(x)$ for any value of s and t has no factors of degree less than ℓ.*

Proof. Let irreducible polynomial $U(x)$ of degree v divide $T_{r,s,t}(x)$. Let α be a root of $U(x)$. By adjoining α to GF[q] we get the extension field GF[q^v]. In this extension field we have $Q_r(A_{s,t}(\alpha)) = T_{r,s,t}(\alpha) = 0$ since $U(x)$ divides $T_{r,s,t}(x)$. Let $\beta = A_{s,t}(\alpha)$. Then, β is a root of $Q_r(z)$ in GF[q^v]. Since $r \neq p$, $\beta \neq 1$. Also, $\beta^r = 1$ in GF[q^v]. Therefore, r must divide the order

[1] If the size of the field is larger than d then such a set of polynomials can be constructed easily: take polynomials of the form $z^{\frac{1}{\epsilon}} - i$ where i varies over d different field elements. However, this does not work for small field sizes as we need to first extend the field so that the size of extension becomes more than d. But for extending the field we need an irreducible polynomial of necessary degree which requires additional random bits. Our method works for all field sizes.

Input: Polynomial $P(x)$ of degree d, field GF$[q]$, and error parameter ϵ.

1. Let $\ell = \lceil \log d \rceil$ and $t = \max\{\lceil \frac{1}{\epsilon} \rceil, \ell\}$.
2. Find the smallest prime r such that $r \neq p$ and r does not divide any of $q-1$, q^2-1, ..., $q^{\ell-1}-1$.
3. Randomly choose an $s \in [0,1]^\ell$ and compute the polynomial $T_{r,s,t}(x) = Q_r(A_{s,t}(x))$.
4. Accept iff $P(x)$ is divisible by $T_{r,s,t}(x)$.

Figure 1: Algorithm A

of the multiplicative group GF$^*[q^v]$. In other words, $r \mid q^v - 1$. By the choice of r, we have $v \geq \ell$. ■

Lemma 4.2 *Let r be chosen as above. Then for any t and any polynomial $U(x)$, $U(x)$ can divide $T_{r,s,t}(x)$ for at most $r-1$ different values of s.*

Proof. Let irreducible polynomial $U(x)$ of degree v divide $T_{r,s_1,t}(x)$, ..., $T_{r,s_a,t}(x)$. As before, consider the extension field GF$[q^v]$ obtained by adjoining a root α of $U(x)$ to GF$[q]$. It follows that $A_{s_1,t}(\alpha)$, ..., $A_{s_a,t}(\alpha)$ are a roots of $Q_r(z)$ over GF$[q^v]$. Consider $A_{s_i,t}(\alpha) - A_{s_j,t}(\alpha)$ for $i \neq j$. This is a polynomial of degree at most $\ell - 1$ in α. Since the minimal polynomial of α over GF$[q]$ has degree v and $v \geq \ell$ (from Lemma 4.1), it follows that the above difference cannot be zero and so $A_{s_i,t}(\alpha) \neq A_{s_j,t}(\alpha)$. Therefore, all the above a roots of $Q_r(z)$ are distinct. As the degree of $Q_r(z)$ is $r-1$, we get $a \leq r-1$. ■

Lemma 4.3 *Let r be chosen as before. Then for any t the lcm of any K polynomials from the set has degree at least $K \cdot t$.*

Proof. Consider polynomials $T_{r,s_1,t}(x)$, $T_{r,s_2,t}(x)$, ..., $T_{r,s_K,t}(x)$ for some K distinct values of s. The product of these polynomials has degree $K \cdot t \cdot (r-1)$. Any polynomial $U(x)$ can divide at most $r-1$ of these according to the Lemma 4.2. Therefore, the lcm of these polynomials has degree at least $K \cdot t$. ■

The algorithm is now obvious—it is given in Figure 1. The following lemma proves its correctness.

Lemma 4.4 *Algorithm A solves identity testing problem for univariate polynomials over finite fields. It errs with probability at most ϵ if $P(x)$ is not identically zero. Further, it uses $\lceil \log d \rceil$ random bits and works in time polynomial in M, $\log q$, and $\frac{1}{\epsilon}$ where M is the size of the arithmetic circuit representing $P(x)$.*

Proof. It is clear that the algorithm uses ℓ random bits. That it works in time polynomial in M, t, r, and $\log q$ follows from the easily observed fact: testing if polynomial P is divisible by a degree v polynomial can be done in time polynomial in v and the size of the circuit by multiplying out the circuit modulo the degree v polynomial.

Number r is the smallest prime that does not divide any of $q-1$, ..., $q^{\ell-1}-1$. By the Prime Number Theorem, $r \leq \ell^3 \cdot (\log q)^2$ (a very crude estimate). Since ℓ is bounded by M, we get the required bound on the time taken.

Input: Polynomial $P(x_1, \ldots, x_n)$ of degree d_i in x_i, field GF$[q]$, and error parameter ϵ.

1. Let $D_i = \prod_{j=1}^{i-1}(d_j + 1)$ for $1 \leq i \leq n$.
2. Let $P'(y) = P(y^{D_1}, y^{D_2}, \ldots, y^{D_n})$.
3. Run Algorithm A on P' and ϵ.

Figure 2: Algorithm B

It is also clear that if $P(x)$ is zero then the algorithm always accepts. If $P(x)$ is non-zero then at most $\frac{d}{t}$ polynomials $T_{r,s,t}(x)$ will divide $P(x)$ by the above lemma. Therefore, the probability of acceptance is at most $\frac{1}{t} \leq \epsilon$. ■

4.2 Multivariate polynomials over finite fields

In this subsection, we generalize the algorithm A to work over multivariate polynomials. The idea is to simply transform the given multivariate polynomial to a univariate one which is zero iff the former is and then apply algorithm A.

Let $P(x_1, x_2, \ldots, x_n)$ be the input polynomial over field GF$[q]$. Let d_i be the degree of x_i in P and $d = \max_i\{d_i\}$.

Let $D_i = \prod_{j=1}^{i-1}(d_j + 1)$ for $1 \leq i \leq n$ and $P_0 = P$. We define a sequence of polynomials P_i for $1 \leq i \leq n$ as follows:

$$\begin{aligned} P_1(y, x_2, \ldots, x_n) &= P(x_1, \ldots, x_n), \\ P_i(y, x_{i+1}, \ldots, x_n) &= P_{i-1}(y, y^{D_i}, x_{i+1}, \ldots, x_n) \text{ for } i > 1. \end{aligned}$$

The following lemma lists crucial properties of these polynomials:

Lemma 4.5 *For $1 \leq i \leq n$, polynomial P_i is zero iff P_{i-1} is. Further, variable y has degree at most $D_{i+1} - 1$ in P_i.*

Proof. We prove this by induction on i. For $i = 1$ the above properties are trivially true. Suppose they are true for $i = j - 1$. P_j is obtained by substituting y^{D_j} for the variable x_j in P_{j-1}. Clearly, if P_{j-1} is zero then so is P_j. Suppose P_{j-1} is not zero. Polynomial P_{j-1} can be written as a degree d_j polynomial in variable x_j whose coefficients are themselves polynomials over the remaining variables. Let $P_{j-1}(y, x_j, \ldots, x_n) = \sum_{l=0}^{d_j} C_l(y, x_{j+1}, \ldots, x_n) \cdot x_j^l$ and let C_m be the highest non-zero coefficient of P_{j-1}. By the induction hypothesis, the degree of y in each of the coefficient polynomials C_l is at most $D_j - 1$. Consider P_j. The term $C_m \cdot y^{m \cdot D_j}$ is a non-zero polynomial that has degree at least $m \cdot D_j$ in y. Any other non-zero term of P_j is a polynomial that has degree at most $(m-1) \cdot D_j + D_j - 1 = m \cdot D_j - 1$ in y. Therefore, P_j cannot be zero. The degree of y in P_j is at most $d_j \cdot D_j + D_j - 1 = D_{j+1} - 1$. ■

The algorithm is now obvious: it is given in Figure 2.

Lemma 4.6 *Algorithm B solves identity testing problem for multivariate polynomials over finite fields. It errs with probability at most ϵ if P is not identically zero. Further, it uses $\lceil \sum_{i=1}^{n} \log d_i \rceil$ random bits and works in time polynomial in M, $\log q$, and $\frac{1}{\epsilon}$ where M is the size of the arithmetic circuit representing the input polynomial.*

Input: Polynomial $P(x_1, \ldots, x_n)$ of degree d_i in x_i, number m, and error parameter ϵ.

1. Let $D_i = \prod_{j=1}^{i-1}(d_j + 1)$ for $1 \leq i \leq n$.
2. Let $P'(y) = P(y^{D_1}, y^{D_2}, \ldots, y^{D_n})$.
3. Let $\ell = \lceil \log d \rceil$ and $t = \max\{\lceil \frac{1}{\epsilon} \rceil, \ell\}$.
4. Randomly choose an $s \in [0,1]^\ell$.
5. For every prime $r \leq \ell^3 \cdot (\log m)^2$ do the following:
   (a) Compute the polynomial $T_{r,s,t}(x) = Q_r(A_{s,t}(x))$.
   (b) Check if $P'(x)$ is divisible by $T_{r,s,t}(x)$ over Z_m.
6. Accept iff $P'(x)$ is divisible by $T_{r,s,t}(x)$ for every value of r.

Figure 3: Algorithm C

Proof. Polynomial P' is a univariate polynomial of degree $\prod_{j=1}^{n}(d_j + 1) - 1$ and is zero iff P is. Moreover, the arithmetic circuit representing P' has size $O(M + n^2 \cdot \log d) = O(M^3)$. The lemma follows from the properties of Algorithm A. ■

4.3 Polynomials over Z_m

In this subsection, we modify our test to work for polynomials over Z_m where m is not necessarily a prime. This is useful in applications such as primality testing as shown in the previous section, and also in testing identities over rationals as we will show in the next section.

We will make use of the following lemma in our test:

Lemma 4.7 *For any univariate polynomial $P(x)$ the following holds:*

- *For any number m, if $P(x) \neq 0 (\mathrm{mod}\ m)$ then there is a prime p and exponent a such that p^{a+1} divides m, and $\frac{1}{p^a} P(x) \neq 0 (\mathrm{mod}\ p)$.*
- *For any prime p and univariate monic polynomial $Q(x)$, if $P(x)$ is not divisible by $Q(x)$ over Z_p then for every number m and a such that p^{a+1} divides m, $p^a \cdot P(x)$ is not divisible by $Q(x)$ over Z_m.*

Proof. If $P(x) \neq 0 (\mathrm{mod}\ m)$ then there is a prime p and exponent b such that p^b divides m but not every coefficient of $P(x)$. Let a be the largest power of p that divides every coefficient of $P(x)$. Then p^{a+1} divides m and $\frac{1}{p^a} P(x) \neq 0 (\mathrm{mod}\ p)$.

If $p^a \cdot P(x) = Q(x) \cdot R(x) (\mathrm{mod}\ m)$ where p^{a+1} divides m then $p^a \cdot P(x) = Q(x) \cdot R(x) + m \cdot R'(x)$ over integers. Since p^a divides m, $Q(x) \cdot R(x)$ is divisible by p^a and since the coefficient of largest power of $Q(x)$ is not divisible by p, p^a must divide $R(x)$. Therefore, $P(x) = Q(x) \cdot \frac{1}{p^a} R(x) (\mathrm{mod}\ p)$. ■

Given a univariate polynomial P and number m, if we run the Algorithm A by doing all the calculations modulo m (instead of over a field), the algorithm would still work correctly: if $P(x) = 0 (\mathrm{mod}\ m)$ it always accepts, and if $P(x) \neq 0 (\mathrm{mod}\ m)$ then by the above lemma $\frac{1}{p^a} P(x) \neq 0 (\mathrm{mod}\ p)$ for some prime p and number a. In this case, any polynomial $Q(x)$ that

does not divide $\frac{1}{p^a}P(x)$ over Z_p will also not divide $P(x)$ over Z_m (by the above lemma). Thus the error probability does not increase.

The only problem is in computing r: the number r should be chosen such that it does not divide $p^j - 1$ for $1 \leq j < \ell$. However, p is not known. This is taken care of by trying out all possible values of r up to $\ell^3 \cdot (\log m)^2$. It was shown earlier that the smallest "right" value of r is bounded by $\ell^3 \cdot (\log p)^2 \leq \ell^3 \cdot (\log m)^2$. Therefore for at least one of the values of r the algorithm would decide correctly and that is sufficient. For the multivariate case, the polynomial is converted to a univariate one in the same way as before—the argument there goes through for rings also. The full algorithm is given in Figure 3.

The above discussion implies the following lemma:

Lemma 4.8 *Algorithm C solves identity testing problem for multivariate polynomials over Z_m for any m. It errs with probability at most ϵ if P is not identically zero. Further, it uses $\lceil \sum_{i=1}^{n} \log d_i \rceil$ random bits and works in time polynomial in M, $\log m$, and $\frac{1}{\epsilon}$ where M is the size of the arithmetic circuit representing P.*

One can modify the Schwartz-Zippel test to work over Z_m in a similar fashion as above.

4.4 Polynomials over rationals

In this subsection, we give a test for identities over rationals. The additional problem we need to take care of is the size of coefficients—in the arithmetic circuit model the size of coefficients can be exponential in the size of the input. We solve this problem, again, by Chinese remaindering: choose a collection of small numbers such that a non-zero polynomial remains non-zero modulo most of these numbers. To keep the number of random bits to a minimum, we do not try to choose prime numbers, or even relatively prime numbers. Instead, we define the collection as follows.

Our collection of numbers will consist of $N + j$ for $0 \leq j < N_0$ and suitable values of N and N_0. The following lemma captures the property we require of this collection:

Lemma 4.9 *For any value of N and N_0, the lcm of any K numbers from the set $S = \{N, N + 1, \ldots, N + N_0 - 1\}$ is at least $2^{K \cdot \log N - 4 \cdot N_0 \cdot (\log N_0)^2}$.*

Proof. The product Π_K of any K numbers from the set is at least N^K. To compute lcm we need to eliminate factors occurring in more than one number. Observe that if number a divides two numbers from the set S then $a < N_0$ since a would also divide their difference. We now count the maximum possible contribution of all the prime numbers (and their powers) less than N_0 to the product Π_K.

A prime $p < N_0$ divides at most N_0/p of the numbers in the set. Thus it contributes at most $p^{N_0/p}$ to the product. But this is not all as there may be multiples of higher powers of p in S. Multiples of p^2 would contribute a further factor of p^{N_0/p^2} to the product etc. In all, prime p and its powers contribute at most $p^{N_0 \cdot (\sum_{i=1}^{\log N_0} \frac{1}{p^i})} < p^{N_0/(p-1)} \leq 2^{N_0 \cdot \log N_0/(p-1)}$ to Π_K. Therefore, the contribution of all primes and their powers less than N_0 is bounded by $2^{N_0 \cdot \log N_0 \cdot (\sum_{i=1}^{N_0} \frac{1}{i})} < 2^{4 \cdot N_0 \cdot (\log N_0)^2}$. The lower bound on the lcm follows. ∎

Figure 4 contains the algorithm for integers. The following lemma proves its correctness.

Lemma 4.10 *Algorithm D solves identity testing problem for multivariate polynomials over integers. It errs with probability at most ϵ if P is not identically zero. Further, it uses $\lceil \sum_{i=1}^{n} \log d_i \rceil + \lceil \log \log C \rceil$ random bits and works in time polynomial in M and $\frac{1}{\epsilon}$.*

Input: Polynomial $P(x_1, \ldots, x_n)$ of degree d_i in x_i, largest coefficient C, and error parameter ϵ.

1. Let $N_0 = \lceil \log C \rceil$, $t = \lceil \frac{2}{\epsilon} \rceil$, and $N = 2^{t \cdot (1+4 \cdot (\log N_0)^2)}$.
2. Randomly choose a number j, $0 \leq j \leq N_0 - 1$.
3. Call Algorithm C with parameters P, $N + j$, and $\epsilon/2$.

Figure 4: Algorithm D

Proof. If P is zero over integers, it remains zero modulo any number and so the algorithm always accepts. If P is non-zero with its largest coefficient being C, it can be zero modulo at most K numbers from the set S where K is such that the lcm of some K numbers from S is less than $C \leq 2^{N_0}$. Since this lcm is at least $2^{K \cdot \log N - 4 \cdot N_0 \cdot (\log N_0)^2}$ according to the previous lemma, we get:

$$K < N_0 \cdot (1 + 4 \cdot (\log N_0)^2) / \log N = N_0 / t.$$

Therefore, with probability at most $\epsilon/2$, P is zero modulo $N + j$. And if P is non-zero modulo $N + j$ then Algorithm C accepts with probability at most $\epsilon/2$. Therefore, the overall error probability is at most ϵ.

It is clear that the algorithm uses the claimed number of random bits, and works in time polynomial in M, $\log \log C$, and $\frac{1}{\epsilon}$. And since $\log \log C$ is bounded by M, we get the desired time bound. ∎

5 De-randomizing identity testing?

Lewin and Vadhan [LV98] suggested that stronger algebraic tools may eventually lead to a complete de-randomization of the identity testing problem. However, at the moment it is not clear how to achieve this. In case of certain specific polynomials we may be able to do this more quickly. Such specific de-randomizations will definitely be of interest, e.g., de-randomization of primality testing polynomial, matching polynomial etc. We now discuss briefly how this may be achieved.

Recall that the primality testing polynomial was $\mathcal{P}(z) = (1 + z)^n - 1 - z^n (\text{mod } n)$. We conjecture that if n is composite, this polynomial is not divisible by at least one polynomial of the form $z^r - 1$ for $r \leq \log n$. The justification for this is that if $\mathcal{P}(z)$ is divisible by $z^r - 1$ then the r sums $\sum_{i=0}^{k:k \cdot r + j < n} \binom{n}{i \cdot r + j}$ for $1 \leq j \leq r$ are all zero modulo n. It seems very unlikely that this happens for all $\log n$ different values of r.

One can come up with suitable hypothesis with other specific polynomials too, e.g., the matching polynomial. The standard polynomial for the matching problem is a multivariate one. First transform it to a univariate one using the trick in Algorithm B and then one can try to find a small sample space for divisor polynomials.

Acknowledgements

We would like to thank Devdatt Dubashi, Ramesh Hariharan, Jaikumar Radhakrishnan, Madhu Sudan, Sundar, and Vinay for useful discussions and suggestions.

References

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegendy. Proof verification and hardness of approximation problems. In *Proceedings of FOCS*, pages 14–23, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs. In *Proceedings of FOCS*, pages 2–13, 1992.

[BCW80] M. Blum, A. K. Chandra, and M. N. Wegman. Equivalence of free boolean graphs can be tested in polynomial time. *Information Processing Letters*, 10:80–82, 1980.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proceedings of FOCS*, pages 16–25, 1990.

[BK95] M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42:269–291, 1995.

[CDGK91] M. Clausen, A. Dress, J. Grabmeier, and M. Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science*, 84(2):151–164, 1991.

[CK97] Zhi-Zhong Chen and Ming-Yang Kao. Reducing randomness via irrational numbers. In *Proceedings of STOC*, pages 200–209, 1997.

[CRS95] Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM Journal on Computing*, 24(5):1036–1050, 1995.

[GKS90] D. Y. Grigoriev, M. Karpinski, and M. F. Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM Journal on Computing*, 19(6):1059–1063, 1990.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nissan. Algebraic methods for interactive proof systems. In *Proceedings of FOCS*, pages 2–10, 1990.

[Lov79] L. Lovasz. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computing Theory*. Akademia-Verlag, 1979.

[LV98] Daniel Lewin and Salil Vadhan. Checking polynomial identities over any field: Towards a derandomization? In *Proceedings of STOC*, pages 000–000, 1998.

[Mil76] G. L. Miller. Riemann's hypothesis and tests for primality. *J. Comput. Sys. Sci.*, 13:300–317, 1976.

[MVV87] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[Rab80] M. O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12:128–138, 1980.

[RB91] R. M. Roth and G. M. Benedek. Interpolation and approximation of sparse multivariate polynomials over GF[2]. *SIAM Journal on Computing*, 20(2):291–314, 1991.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Sha90] A. Shamir. IP = PSPACE. In *Proceedings of FOCS*, pages 11–15, 1990.

[SS77] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6:84–86, 1977.

[Zip79] R. E. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSCAM'79*, pages 216–226. Springer LNCS 72, 1979.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'