Primality and Identity Testing via Chinese Remaindering*

Manindra Agrawal†
Department of Computer Science and Engineering
Indian Institute of Technology
Kanpur 208016, India
manindra@iitk.ac.in

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

February 21, 2003

Abstract

We give a simple and new randomized primality testing algorithm by reducing primality testing for number n to testing if a specific univariate identity over Z_n holds.

We also give new randomized algorithms for testing if a multivariate polynomial, over a finite field or over rationals, is identically zero. The first of these algorithms also works over Z_n for any n. The running time of the algorithms is polynomial in the size of arithmetic circuit representing the input polynomial and the error parameter. These algorithms use fewer random bits and work for a larger class of polynomials than all the previously known methods, e.g., the Schwartz-Zippel test [Sch80, Zip79], Chen-Kao and Lewin-Vadhan tests [CK97, LV98].

1 Introduction

Primality testing is one of the fundamental problems in algorithmic number theory with important applications in cryptography and elsewhere. A number of "efficient" algorithms for the problem have been known for a long time [Mil76, Rab80, SS77, APR83]. However, the problem was not known to be in P until recently:[1] while the algorithms of [Rab80, SS77] are randomized, the algorithm of [APR83] requires (slightly) super-polynomial time, and the algorithm of [Mil76] is in P only under an unproved number-theoretic hypothesis. All of these algorithms are based on various properties of prime numbers, e.g., Fermat's Little Theorem (in [Rab80, Mil76]), quadratic residues in prime fields (in [SS77]) etc.

In this paper, we propose a new test for primality based on the following polynomial identity that prime numbers n (and only prime numbers) satisfy:

$$(1+z)^n = 1 + z^n (\text{mod } n).$$

*A preliminary version of this paper was presented in FOCS'99.

†This research was partially supported by grant no. AICTE/CSE/96263 from AICTE.

[1] Very recently, the problem was finally shown to be in P by modifying the primality test we propose in this paper [AKS02].

Our test is also a randomized polynomial-time algorithm and is arguably the simplest among all known primality tests.

After transforming primality testing to a polynomial identity testing problem, we turn our attention to identity testing. This is also a fundamental problem that has a number of applications in different branches of algorithms and complexity: matching [Lov79, MVV87, CRS95], read-once branching programs [BCW80], multi-set equality [BK95] etc. It also arises 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 most general form of the identity testing problem is: given a polynomial $P(x_1, \ldots, x_n)$ and a ring R, check if the polynomial is identically zero over R (the primality testing identity above is univariate). The problem is trivial if the input polynomial P is given explicitly as a sum of monomials—P is zero iff the coefficient of each monomial is zero. (Notice that we are treating the polynomial as formal polynomial. So, for example, the polynomial $x^2 - x$ is a *non-zero* polynomial over F_2 for us). However, in general, P is given in some implicit form, e.g., a symbolic determinant, product of polynomials, straight-line program, arithmetic circuit 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 ring R, these can be used to simulate subtraction). We will also assume that the two ring operations can be done efficiently. The degree of a variable in such a polynomial can be exponential in the depth of the circuit. We let d_i denote the degree of the variable x_i.

The identity testing problem is also not known to belong to P.[2] However, a number of randomized polynomial-time algorithms are known for it when the ring R is a field, say F. The first such algorithm was given simultaneously by Schwartz and Zippel [Sch80, Zip79] who 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. The time taken by this test is polynomial in the size of the circuit and $\log \frac{1}{\epsilon}$.

A few years ago, 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 by the test are the same ($= \sum_{i=1}^{n} \lceil \log(d_i + 1) \rceil$) *irrespective* of the error probability. This gives a novel time-error tradeoff instead of the usual randomness-error tradeoff. The time taken by the algorithm is a polynomial in n, $d = \sum_{i=1}^{n} d_i$, and $\frac{1}{\epsilon}$ as the bit-length of the approximation is required to be at least $d + \frac{1}{\epsilon}$. This makes the algorithm exponential time on arithmetic circuits since the degree of the polynomial can be exponential in the size of the circuit as noted above. The interesting feature of the algorithm, apart from the novelty of approach, is the time-error tradeoff that it provides.

Lewin and Vadhan [LV98] generalized the Chen-Kao paradigm to work for identity testing

[2] Very recently, it has been shown that proving the problem in P implies lower bounds on arithmetic circuits [KI02]: if this problem is in P then the class NEXP does not have polynomial-sized arithmetic circuits.

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 this paper, we propose another paradigm for identity testing: via Chinese remaindering over polynomials. The idea here is simple: first (deterministically) transform the given multivariate polynomial to a (exponentially) higher degree univariate polynomial such that the resulting polynomial is zero iff the original one is. Then test if the univariate polynomial is zero using division by a small degree polynomial randomly chosen from a suitable set. By selecting the sample space of small degree polynomials carefully, we are able to obtain algorithms both for finite fields and for $\mathcal{Q}$ that work in time *polynomial* in the size of the arithmetic circuit and $\frac{1}{\epsilon}$, yielding at the same time the time-error tradeoff as in Chen-Kao and Lewin-Vadhan. This is considerably better than these two algorithms as their running time is polynomial in $\frac{1}{\epsilon}$ and *exponential* in the size of the arithmetic circuits.

We summarize the running times and random bits required by various algorithms in the table below. In the table, s denotes the size of the input arithmetic circuit, n the number of variables, d_i the degree of i^{th} variable, and $d = \sum_{i=1}^{n} d_i$.

Algorithm	Over Rationals		Over Finite Field F_q	
	Time	Random Bits	Time	Random Bits
Schwartz-Zippel	$(s + \log \frac{1}{\epsilon})^{O(1)}$	$\Omega(n(\log d + \log \frac{1}{\epsilon}))$	$(s + \log \frac{1}{\epsilon})^{O(1)}$	$\Omega(n(\log d + \log \frac{1}{\epsilon}))$
Chen-Kao	$2^{O(s+\log \frac{1}{\epsilon})}$	$\sum_{i=1}^{n} \lceil \log d_i \rceil$	-	-
Lewin-Vadhan	-	-	$2^{O(s+\log \frac{1}{\epsilon})}$	$\sum_{i=1}^{n} \lceil \log d_i \rceil$
Our Algorithms	$(s + \frac{1}{\epsilon})^{O(1)}$	$\lceil \sum_{i=1}^{n} \log d_i \rceil$	$(s + \frac{1}{\epsilon} + \log q)^{O(1)}$	$\lceil \sum_{i=1}^{n} \log d_i \rceil$

Our algorithms for finite fields also works for Z_n where n is any number not necessarily prime. This makes our algorithms work for a larger class of polynomials than any previous algorithm.

Remark. If the time complexity of an algorithm providing a time-error tradeoff grows as $(\frac{1}{\epsilon})^{o(1)}$ then the algorithm can be derandomized in subexponential time by choosing $\frac{1}{\epsilon}$ to be larger than 2^r where r is the number of random bits used. This means that the time complexity of our algorithms is optimal up to a polynomial.

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 an algebraic element α. F_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. In case the underlying ring is integers or rationals, the size of coefficients too can be exponential in the size of the circuit.

We use the notation $O^{\sim}(t(n))$ to denote the time complexity $O(t(n) \cdot (\log t(n))^{O(1)})$. We will use the following facts about the operations over polynomial rings [vzGG99]: given three polynomials $P(x)$, $Q(x)$, and $R(x)$ over the field F_q with their degrees bounded by d, the time complexity of adding and multiplying $P(x)$ and $Q(x)$ modulo $R(x)$ over F_q is $O^{\sim}(d \cdot \log q)$.

3 Primality testing

In this section, we exhibit a new primality test which is based on a polynomial identity satisfied by prime numbers. This identity is, in fact, a well known property of finite fields that is used in many places.

Let

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

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

Proof. We can rewrite $\mathcal{P}_n(z)$ as:

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

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

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 $\mathcal{P}_n(z) \neq 0 (\mathrm{mod}\ n)$. ∎

Remark. The above identity is a generalization of Fermat's Little Theorem which states that for every prime n, and for every $a \in Z_n$, $a^n = a (\mathrm{mod}\ n)$. This can be obtained from the above identity by restricting z to elements of Z_n: $(0+1)^n = 0^n + 1 \pmod{n}$ (base case), and inductively assuming that $i^n = i \pmod{n}$ we get $(i+1)^n = i^n + 1 \pmod{n}$ (by the identity) $= i + 1 \pmod{n}$ (by the induction hypothesis).

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. However, the following simple algorithm works:

```
Prime(n).
1.  If n equals 2, 3, 5, 7, 11, or 13, output PRIME.
2.  Else if n is divisible by any of the above numbers, output
    COMPOSITE.
3.  If n is of the form a^b for b > 1, output COMPOSITE.
4.  Else, randomly choose a monic polynomial Q(z) of degree ⌈log n⌉
    with coefficients in Z_n.
5.  Output PRIME if Q(z) divides P_n(z) over Z_n, else output
    COMPOSITE.
```

Algorithm for Primality Testing

Theorem 3.2 *If n is prime, Prime(n) outputs PRIME with probability one, and if n is composite, Prime(n) outputs COMPOSITE with probability at least $\frac{2}{3}$. Further, Prime(n) requires $\lceil \log n \rceil^2$ random bits and works in time $O^{\sim}((\log n)^4)$.*

Proof. If n is prime, the polynomial $\mathcal{P}_n(z)$ is identically zero over Z_n, and therefore the algorithm always outputs PRIME.

If n is composite, the polynomial $\mathcal{P}_n(z)$ is non-zero over Z_n. Since the algorithm is correct when n is a prime power or divisible by primes up to 13, we need to analyze the case when n not a prime power, odd, and its every prime factor is at least 17.

We first observe that $\mathcal{P}_n(z)$ is also non-zero over Z_p for any prime factor p of n: if p^a is the largest power of p that divides n then $\binom{n}{p^a}$ is not divisible by p and hence the term for z^{p^a} will survive.[3]

We will now argue over Z_p since polynomials factorize uniquely over it. Let $\ell = \lceil \log n \rceil$. In the sample space for $Q(z)$, any monic polynomial of degree ℓ over Z_p occurs *exactly* $(\frac{n}{p})^\ell$ times. Also, since the degree of $\mathcal{P}_n(z)$ is n, it can have at most $\frac{n}{\ell}$ different irreducible factors of degree ℓ over Z_p. Using this, we can almost immediately obtain a weak bound on probability of error as follows: From the distribution theorem for irreducible polynomials (see, e.g., [LN86]) it follows that $I(d)$, the number of monic irreducible polynomials of degree d over F_p, is between $\frac{p^d}{d} - p^{d/2}$ and $\frac{p^d}{d} + p^{d/2}$. Since $n > p > 16$, $d = \ell > 4$. Therefore, $I(d) \geq \frac{1}{2\ell} * p^\ell$. Therefore, $Q(z)$ will be an irreducible polynomial over Z_p with probability at least $\frac{1}{2\ell}$ and so $Q(z)$ will not divide $\mathcal{P}_n(z)$ with probability at least $\frac{1}{2\ell} - \frac{n}{p^\ell \ell} \geq \frac{1}{4 \log n}$ (since $n > p > 16$). Since if $Q(z)$ divides $\mathcal{P}_n(z)$ over Z_n, it also divides it over Z_p, it follows that with probability at least $\frac{1}{4 \log n}$, $Q(z)$ will not divide $\mathcal{P}_n(z)$ over Z_n.

A more careful counting gives us a better bound on error: Let $\mathcal{I}$ be the set of monic irreducible polynomials of degree between $1 + \frac{\ell}{2}$ and ℓ over F_p. For $P \in \mathcal{I}$, let C_P be the set of degree ℓ polynomials that have polynomial P as a factor. Clearly, $|C_P| = p^{\ell - \deg(P)}$. Also, $C_P \cap C_{P'} = \emptyset$ for $P \neq P'$, since their degree is more than $\frac{\ell}{2}$. So the number of monic polynomials of degree ℓ in above sets is

$$\begin{aligned}
\sum_{P \in \mathcal{I}} |C_P| &= \sum_{k=1+\frac{\ell}{2}}^{\ell} I(k) \cdot p^{\ell - k} \\
&\geq \sum_{k=1+\frac{\ell}{2}}^{\ell} p^\ell \cdot (\frac{1}{k} - \frac{1}{p^{k/2}}) \\
&\geq (\log_e 2 - \frac{1}{48}) \cdot p^\ell \text{ (since } n > p > 16).
\end{aligned}$$

Also, $|C_P| \leq p^{\frac{\ell}{2}-1}$. So at most $\frac{2n}{\ell} \cdot p^{\frac{\ell}{2}-1}$ of these polynomials can divide $\mathcal{P}_n(z)$. Therefore, the probability that a random monic Q of degree ℓ does not divide $\mathcal{P}_n(z)$ is at least

$$\log_e 2 - \frac{1}{48} - \frac{1}{8n\ell} > \frac{2}{3}$$

since $n > 16$.

The algorithm clearly requires $\lceil \log n \rceil^2$ random bits. It does $O(\log n)$ multiplications of two degree $O(\log n)$ polynomials over Z_n and computes same number of remainders modulo a third degree $O(\log n)$ polynomial. Each of these requires $O^{\sim}((\log n)^3)$ steps. Thus the time complexity of the algorithm is $O^{\sim}((\log n)^4)$. ∎

[3] This is not true when n is a prime power!

Remark. In our analysis of the test, we are not considering degree $\lceil \log n \rceil$ polynomials with small irreducible factors. It can be shown that even if one considers these polynomials, the error probability does not reduce substantially. In particular, a degree n polynomial that consists entirely of "small" (of degree less than $\frac{\log n}{\log p}$) irreducible factors will pass the test with at least a constant probability. So unless one makes use of the special structure of the polynomial $\mathcal{P}_n(z)$, our analysis is within a constant factor of the optimal.

Remark. In comparison to our test, the standard Miller-Rabin test requires $\lceil \log n \rceil$ random bits, takes time $O^{\sim}((\log n)^2)$ and errs with probability at most $\frac{1}{4}$. Clearly, this test scores over our test in terms of practical utility (the same is true for many other tests too). However, our test is conceptually simpler and has an easier proof of correctness.

In the next section, we give another algorithm for testing the above identity that uses fewer random bits (this will fall out of the general algorithm for testing identities).

New Development: Deterministic Algorithm for Primality Testing

Very recently [AKS02], a deterministic polynomial-time algorithm was given for primality testing. The new approach to primality testing introduced by our algorithm was instrumental in the design of this deterministic algorithm. In fact, the new algorithm is simply a *derandomization* of our randomized algorithm. This can be seen in the following way.

Our randomized algorithm is based on testing the identity $(1+z)^n = 1+z^n \pmod{n}$ modulo a randomly chosen polynomial $Q(z)$ of degree $\lceil \log n \rceil$ and succeeds with probability at least $\frac{2}{3}$. The sample space for $Q(z)$ is clearly exponential sized $(= n^{\lceil \log n \rceil})$. It is shown in [AKS02] that the sample space for $Q(z)$ can be shrunk to $O(\log^4 n)$ without reducing the probability of success! They show that it is enough to consider $Q(z)$ of the form $(z+a)^r - 1$ for a fixed $r = O(\log^6 n)$ and a varying from 1 to $O(\log^4 n)$. Although their algorithm appears to be testing a number of identities of the form $(z-a)^n = z^n - a \pmod{n}$ modulo $z^r - 1$, these are equivalent to our identity modulo different $Q(z)$'s of the above form, as shown by the lemma below.

Lemma 3.3 *Fix any $r > 0$ and any $t > 0$. Then,*

$$(z+1)^n = z^n + 1 \pmod{n, (z+a)^r - 1} \text{ for } 1 \le a \le t \tag{1}$$

if and only if

$$(z-a)^n = z^n - a \pmod{n, z^r - 1} \text{ for } 1 \le a \le t. \tag{2}$$

(Here notation $\pmod{n, Q(z)}$ denotes that the coefficients are reduced modulo n and the polynomial modulo $Q(z)$.)

Proof. The proof is by induction on t. When $t = 1$, equation (1) gives:

$$(z+1)^n = z^n + 1 \pmod{n, (z+1)^r - 1}.$$

Making the substitution $z-1$ for z everywhere and rearranging terms we get the equation (2) for $a = 1$. Similarly, starting with equation (2), making the substitution $z+1$ for z and rearranging terms we get equation (1).

Now suppose the equivalence holds for every a up to $t-1$. Then, for $a = t$, equation (1) is:

$$(z+1)^n = z^n + 1 \pmod{n, (z+t)^r - 1}.$$

Substituting $z - t$ for z, we get:

$$(z - (t-1))^n = (z-t)^n + 1 \pmod{n, z^r - 1}.$$

Using the induction hypothesis for $a = t - 1$, we can replace the LHS of the last equation by $z^n - (t-1)$. Rearranging, we get equation (2) for $a = t$.

To prove the other direction, start with equation (2) for $a = t$:

$$(z-t)^n = z^n - t \pmod{n, z^r - 1}.$$

Substituting $z + 1$ for z, we get:

$$(z - (t-1))^n = (z+1)^n - t \pmod{n, (z+1)^r - 1}.$$

Using the induction hypothesis for $a = 1$, we can replace the RHS of the last equation by $z^n + 1 - t$. Now substituting $z + 1$ for z again, we get:

$$(z - (t-2))^n = (z+1)^n - (t-1) \pmod{n, (z+2)^r - 1}.$$

Using the induction hypothesis for $a = 2$, we again replace RHS of this equation by $z^n - (t-2)$. Continuing this way, we eventually get equation (1) for $a = t$. ■

4 Identity testing

In this section, we give our identity testing algorithms. The section is divided into 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 for 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 a finite field F_q of characteristic p. Let d be the degree of P. In this subsection operations will always be over the field F_q unless explicitly mentioned otherwise. We can use essentially the same algorithm presented in the previous section to test polynomial P. However, the success probability in case the polynomial is non-zero can only be amplified by repeating the test a number of times—this does not achieve a time-error tradeoff. We can increase this probability to $1 - \epsilon$ by choosing the divisor polynomial Q from a set of polynomials such that the lcm of any ϵ fraction of these polynomials has degree at least d. Obviously, it should be possible to sample from such a set of polynomials in polynomial time.

One possible way of doing this is to consider a 'large' set of mutually co-prime polynomials. This can be achieved by choosing the set of polynomials to be

$$\{(z - i) \mid i \in F_q\}.$$

However, for the required error bound, this set should contain at least $\frac{d}{\epsilon}$ polynomials and this is possible only when $d \leq \epsilon \cdot q$. Otherwise, we need to first extend the field F_q to a larger field F_{q^u} such that $q^u \geq \frac{d}{\epsilon}$ and then for the above set of polynomials over F_{q^u}.[4] But this requires construction of a degree u irreducible polynomial over F_q which in turn, requires additional

[4]This scheme, in effect, is precisely the Schwartz-Zippel test for univariate identities!

random bits. This scheme, therefore, cannot achieve the time-error tradeoff and also is restricted to work for fields.

We, therefore, take a different approach. Instead of extending the field and using the above family of polynomials, we construct a different family of polynomials of higher degree that satisfy the required property. Our family may not have all mutually co-prime polynomials, but, the polynomials in the family share very few common factors.

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 $\ell \geq 0$. For a binary vector $b \in [0,1]^\ell$ and number $t \geq \ell$, let

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

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

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

Our set of polynomials will consist of $T_{r,b,t}$ for all values of b 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 $T_{r,b,t}(x)$ for any value of b and t has no factors of degree less than ℓ.*

Proof. Let irreducible polynomial $U(x)$ of degree v divide $T_{r,b,t}(x)$. Let α be a root of $U(x)$. By adjoining α to F_q we get the extension field F_{q^v}. In this extension field we have $Q_r(A_{b,t}(\alpha)) = T_{r,b,t}(\alpha) = 0$ since $U(x)$ divides $T_{r,b,t}(x)$. Let $\beta = A_{b,t}(\alpha)$. Then, β is a root of $Q_r(z)$ in F_{q^v}. Since $r \neq p$, $\beta \neq 1$. Also, $\beta^r = 1$ in F_{q^v}. Therefore, r must divide the order of the multiplicative group $F^*_{q^v}$. 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 $T_{r,b,t}(x)$ for at most $r-1$ different values of b.*

Proof. Let irreducible polynomial $U(x)$ of degree v divide $T_{r,b_1,t}(x)$, ..., $T_{r,b_a,t}(x)$. As before, consider the extension field F_{q^v} obtained by adjoining a root α of $U(x)$ to F_q. It follows that $A_{b_1,t}(\alpha)$, ..., $A_{b_a,t}(\alpha)$ are a roots of $Q_r(z)$ over F_{q^v}. Consider $A_{b_i,t}(\alpha) - A_{b_j,t}(\alpha)$ for $i \neq j$. This is a polynomial of degree at most $\ell - 1$ in α. Since the minimal polynomial of α over F_q has degree v and $v \geq \ell$ (from Lemma 4.1), it follows that the above difference cannot be zero and so $A_{b_i,t}(\alpha) \neq A_{b_j,t}(\alpha)$. 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 $T_{r,b_1,t}(x)$, $T_{r,b_2,t}(x)$, ..., $T_{r,b_K,t}(x)$ for some K distinct values of b. 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 theorem proves its correctness.

Input: An arithmetic circuit of size s representing polynomial $P(x)$ of degree d over field F_q, number q, and error parameter ϵ.

1. Let $\ell = \lceil \log d \rceil$, and $t = \max\{\ell, \frac{1}{\epsilon}\}$.
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 a $b \in [0,1]^\ell$ and compute the polynomial $T_{r,b,t}(x) = Q_r(A_{b,t}(x))$.
4. Accept iff $P(x)$ is divisible by $T_{r,b,t}(x)$.

Figure 1: Algorithm A

Theorem 4.4 *Let $P(x)$ be a univariate polynomial of degree d over finite field F_q presented as an arithmetic circuit of size s. Then, Algorithm A tests if $P(x)$ is identically zero and 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 $O^\sim(s^3 \cdot (s + \frac{1}{\epsilon}) \cdot (\log q)^2)$.*

Proof. It is clear that the algorithm uses ℓ random bits. Its time complexity follows from the following fact: testing if polynomial P is divisible by a degree v polynomial can be done by computing the value of each gate of the circuit modulo the degree v polynomial. Since the degree of polynomials $T_{r,b,t}(x)$ is bounded by $r \cdot t$, the time complexity of the algorithm is $O^\sim(s \cdot r \cdot t \cdot \log q)$.

Number r is the smallest prime that does not divide any of $q-1$, ..., $q^{\ell-1}-1$. By the Prime Number Theorem, $r = O(\ell^2 \cdot \log q)$ (as the product of all primes smaller than x is about e^x and the product of $q-1$, ..., $q^{\ell-1}-1$ is bounded by q^{ℓ^2}). Since d is bounded by 2^s, and t is $O(\ell + \frac{1}{\epsilon})$, we get the required bound on the time.

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 Lemma 4.3. Therefore, the probability of acceptance is at most $\frac{d}{2^\ell \cdot t} \leq \epsilon$. ∎

Remark. Our construction of a family of nearly coprime polynomials can be used in other places too. For example, in [AGHP90], one of the constructions of k-wise ϵ-biased source requires to uniformly and efficiently sample irreducible polynomials of degree d. This is used to show that a certain degree 2^d polynomial is divisible by very few of these randomly generated polynomials. One can replace the irreducible polynomials by our construction of nearly coprime polynomials and the same property will continue to hold. This saves on random bits needed for the source.

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 F_q. Let d_i be the degree of x_i in P.

Input: An arithmetic circuit of size s representing polynomial $P(x_1, \ldots, x_n)$ of degree d_i in x_i over field F_q, number 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

Let $D_i = \prod_{j=1}^{i-1}(d_j + 1)$ for $1 \leq i \leq n$. We define a sequence of polynomials P_i for $0 \leq i \leq n$ as follows:

$$\begin{aligned} P_0(y, x_1, x_2, \ldots, x_n) &= P(x_1, \ldots, x_n), \\ P_i(y, x_{i+1}, \ldots, x_n) &= P_{i-1}(y, y^{D_i}, x_{i+1}, \ldots, x_n) \\ &= P(y^{D_1}, y^{D_2}, \ldots, y^{D_i}, x_{i+1}, \ldots, x_n) \text{ for } i > 0. \end{aligned}$$

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.

Theorem 4.6 *Let $P(x_1, \ldots, x_n)$ be a multivariate polynomial of degree d_i in x_i over finite field F_q presented as an arithmetic circuit of size s. Then, Algorithm B tests if P is identically zero and 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 $O^{\sim}(s^3 n^3 \cdot (sn + \frac{1}{\epsilon}) \cdot (\log q)^2)$.*

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 $O(s + n \cdot (\sum_{i=1}^{n} \log d_i)) = O(sn)$. The theorem follows from Theorem 4.4. ■

Input: An arithmetic circuit of size s representing polynomial $P(x_1, \ldots, x_n)$ of degree d_i in x_i over Z_m, 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 $\ell = \lceil \log D_n \rceil$ and $t = \max\{\lceil \frac{1}{\epsilon} \rceil, \ell\}$.
4. Randomly choose a $b \in [0,1]^\ell$.
5. For every prime $r \leq \ell^2 \cdot \log m$ do the following:
   (a) Compute the polynomial $T_{r,b,t}(x) = Q_r(A_{b,t}(x))$.
   (b) Check if $P'(x)$ is divisible by $T_{r,b,t}(x)$ over Z_m.
6. Accept iff $P'(x)$ is divisible by $T_{r,b,t}(x)$ for every value of r.

Figure 3: Algorithm C

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 $\ell^2 \cdot \log m$. It was shown earlier that the smallest "right" value of r is bounded by $\ell^2 \cdot \log p \leq \ell^2 \cdot \log m$. 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 theorem:

Theorem 4.8 *Let $P(x_1, \ldots, x_n)$ be a multivariate polynomial of degree d_i in x_i over Z_m presented as an arithmetic circuit of size s. Then, Algorithm C tests if P is identically zero and 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 $O^{\sim}(s^5 n^5 \cdot (sn + \frac{1}{\epsilon}) \cdot (\log m)^3)$.*

4.4 Polynomials over rationals

In this subsection, we give a test for identities over rationals. Here, since the field in infinite, we can choose degree-one polynomial family

$$\{(z - i) \mid i \in \mathcal{Q}\}$$

irrespective of the degree of P (of course, first we convert the polynomial to a univariate one). The only 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 using a standard trick also based on Chinese remaindering: choose a collection of small numbers such that a non-zero polynomial remains non-zero modulo most of these numbers. These numbers can simply be the set of all numbers up to the size of the circuit. The probability of success may be low here and needs to be increased by repetition.

The above method, however, does not yield a time-error tradeoff as was the case with earlier algorithms. To achieve this tradeoff, we again adopt a different route. First we define a collection of numbers: let $S = \{N, N + 1, \ldots, N + N_0 - 1\}$ for suitable values of N and N_0 to be fixed later. 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 is at least $2^{K \cdot \log N - 2 \cdot N_0 \cdot \log N_0}$.*

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. Any prime power p^i with $p^i < N_0$ divides at most $\lceil \frac{N_0}{p^i} \rceil$ of the numbers in the set S. On the other hand, p^i divides at least $\lfloor \frac{N_0}{p^i} \rfloor \geq \frac{1}{2} \cdot \lceil \frac{N_0}{p^i} \rceil$ numbers among the first N_0 numbers. Therefore, the contribution of all numbers less than N_0 to the product Π_K is at most $(N_0!)^2 \leq 2^{2 \cdot N_0 \cdot \log N_0}$. The lower bound on the lcm follows. ∎

We now reduce the identity over integers to an identity over Z_m by going modulo one of the numbers in the family, and then invoke the test for Z_m. Figure 4 contains the algorithm for integers. The following theorem proves its correctness.

Input: An arithmetic circuit of size s representing polynomial $P(x_1, \ldots, x_n)$ of degree d_i in x_i over rationals with its largest coefficient bounded by C, and error parameter ϵ.

1. Let $N_0 = \lceil \log C \rceil$, $t = \lceil \frac{2}{\epsilon} \rceil$, and $N = 2^t \cdot N_0^{2t}$.
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

Theorem 4.10 *Let $P(x_1, \ldots, x_n)$ be a multivariate polynomial of degree d_i in x_i over rationals with its largest coefficient bounded by C and presented as an arithmetic circuit of size s. Then, Algorithm D tests if P is identically zero and 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 $O^{\sim}(s^5 n^5 \cdot (sn + \frac{1}{\epsilon}) \cdot (\frac{\log \log C}{\epsilon})^3)$.*

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 $2^{K \cdot \log N - 2 \cdot N_0 \cdot \log N_0}$ according to the previous lemma, we get:

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

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 straightforward to see that the algorithm has the claimed bounds on the number of random bits and time complexity. ∎

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. This problem assumes added importance in light of the recent result [KI02] that a complete derandomization of identity testing implies that NEXP does not have polynomial-sized arithmetic circuits. 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 easily, and one such derandomization has indeed been shown for primality testing [AKS02].

One can try a similar approach for derandomizing other specific polynomials too, e.g., the matching polynomial. The standard polynomial for the matching problem is a multivariate one. One can first transform it to a univariate one using the trick in Algorithm B and then try to find a small sample space for divisor polynomials in a similar fashion.

Acknowledgments

We thank Devdatt Dubashi, Ramesh Hariharan, Jaikumar Radhakrishnan, Sundar, and Vinay for useful discussions. We thank Madhu Sudan, Salil Vadhan, and two anonymous referees for useful suggestions.

References

[AGHP90] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 544–553, 1990.

[AKS02] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. Preprint (http://www.cse.iitk.ac.in/users/manindra/primality.ps), August 2002.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegendy. Proof verification and hardness of approximation problems. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[APR83] L. M. Adleman, C. Pomerance, and R. S. Rumely. On distinguishing prime numbers from composite numbers. *Ann. Math.*, 117:173–206, 1983.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, 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 Annual IEEE Symposium on Foundations of Computer Science*, 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 Annual ACM Symposium on the Theory of Computing*, 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.

[KI02] Valentine Kabanets and Russell Impagliazzo. Derandomizing polyonmial identity test means proving circuit lower bounds. TR02-055 (http://www.eccc.uni-trier.de/eccc-local/Lists/TR-2002.html), 2002.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nissan. Algebraic methods for interactive proof systems. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990.

[LN86] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications.* Cambridge University Press, 1986.

[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 Annual ACM Symposium on the Theory of Computing*, pages 428–437, 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 Annual IEEE Symposium on Foundations of Computer Science*, 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.

[vzGG99] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, 1999.

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