Fast modular composition in any characteristic

Kiran S. Kedlaya[*]
MIT

Christopher Umans[†]
Caltech

August 3, 2008

Abstract

We give an algorithm for modular composition of degree n univariate polynomials over a finite field $\mathbb{F}_q$ requiring $n^{1+o(1)} \log^{1+o(1)} q$ bit operations; this had earlier been achieved in characteristic $n^{o(1)}$ by Umans (2008). As an application, we obtain a randomized algorithm for factoring degree n polynomials over $\mathbb{F}_q$ requiring

$$(n^{1.5+o(1)} + n^{1+o(1)} \log q) \log^{1+o(1)} q$$

bit operations, improving upon the methods of von zur Gathen & Shoup (1992) and Kaltofen & Shoup (1998). Our results also imply algorithms for irreducibility testing and computing minimal polynomials whose running times are best-possible, up to lower order terms.

As in Umans (2008), we reduce modular composition to certain instances of multipoint evaluation of multivariate polynomials. We then give an algorithm that solves this problem optimally (up to lower order terms), in arbitrary characteristic. The main idea is to lift to characteristic 0, apply a small number of rounds of *multimodular reduction*, and finish with a small number of multidimensional FFTs. The final evaluations are then reconstructed using the Chinese Remainder Theorem. As a bonus, we obtain a very efficient data structure supporting polynomial evaluation queries, which is of independent interest.

Our algorithm uses techniques which are commonly employed in practice, so it may be competitive for real problem sizes. This contrasts with previous asymptotically fast methods relying on fast matrix multiplication.

[*] Supported by NSF DMS-0545904 (CAREER) and a Sloan Research Fellowship.

[†] Supported by NSF CCF-0346991, BSF 2004329, a Sloan Research Fellowship, and an Okawa Foundation research grant.

1 Introduction

The problem of MODULAR COMPOSITION is, given three univariate polynomials $f(x), g(x), h(x)$ over a ring with h having invertible leading coefficient, to compute $f(g(x)) \pmod{h(x)}$. Modular composition serves as the backbone of numerous algorithms for computing with polynomials over finite fields, most notably the asymptotically fastest methods for polynomial factorization.

In contrast to other basic modular operations on polynomials (e.g modular multiplication), it is *not* possible to obtain an asymptotically fast algorithm for modular composition with fast algorithms for each step in the natural two step procedure (i.e., first compute $f(g(x))$, then reduce modulo $h(x)$). This is because $f(g(x))$ has n^2 terms, while we hope for a modular composition algorithm that uses only about $O(n)$ operations. Not surprisingly, it is by considering the overall operation (and beating n^2) that asymptotic gains are made in algorithms that employ modular composition.

Perhaps because nontrivial algorithms for modular composition must handle the modulus in an integrated way (rather than computing a remainder after an easier, nonmodular computation) there have been few algorithmic inroads on this seemingly basic problem. Brent & Kung [BK78] gave the first nontrivial algorithm in 1978, achieving an operation count of $O(n^{(\omega+1)/2})$, where ω is the exponent of matrix multiplication. Huang & Pan [HP98] achieved a slight improvement, by noting that the bound is actually $O(n^{\omega_2/2})$ where ω_2 is the exponent of $n \times n$ by $n \times n^2$ matrix multiplication, and giving an upper bound on ω_2 that is slightly better than the best known bound on ω, plus one. These algorithms cannot beat $O(n^{1.5})$, and it is not feasible in practice to achieve their theoretical guarantees, because those rely on the asymptotically fastest algorithms for matrix multiplication, which are currently impractical. Finding new algorithms for MODULAR COMPOSITION with running times closer to $O(n)$ was mentioned several times as an important and longstanding open problem (cf. [Sho94, KS98], [BCS97, Problem 2.4], [vzGG99, Research Problem 12.19]).

Very recently, Umans [Uma08] gave an algorithm that achieves the optimal operation count up to lower order terms, but only in fields with small characteristic (specifically, the characteristic p was required to be $n^{o(1)}$).

In this paper, we essentially solve the MODULAR COMPOSITION problem completely, presenting an algorithm for modular composition over any finite field, whose running time is optimal up to lower order terms. Our algorithm uses the reduction from MODULAR COMPOSITION to MULTIVARIATE MULTIPOINT EVALUATION from [Uma08], and then solves the latter problem in a completely different way, by lifting to characteristic 0 followed by multimodular reduction and a small number of multidimensional FFTs.

In contrast to [Uma08], our algorithm is nonalgebraic, which carries some minor disadvantages. One is that a general method (the "transposition principle") for transforming an algebraic algorithm for MODULAR COMPOSITION into one for the *transpose* problem (called MODULAR POWER PROJECTION, itself useful in algorithms for computing with polynomials), does not directly apply. However, in Section 5.2 we show that this disadvantage can be overcome – the nonalgebraic parts of our algorithm interact well with the transposition principle – and consequently we obtain an algorithm for MODULAR POWER PROJECTION whose running time is optimal up to lower order terms.

A major advantage of our algorithm (apart from working in any characteristic) is that it is simple, practical and implementable. Multimodular reduction is used in practice in a variety of settings, and while we use it recursively to state our most general results, only two rounds are required to achieve an algorithm for MODULAR COMPOSITION whose running time is optimal up to lower order terms.

1.1 From modular composition to multipoint evaluation

While the algorithms of [BK78] and [HP98] reduce MODULAR COMPOSITION to matrix multiplication, the method of [Uma08] reduces MODULAR COMPOSITION to the problem of MULTIVARIATE MULTIPOINT EVALUATION of polynomials over $\mathbb{F}_q$: given an m-variate polynomial $f(x_0, \ldots, x_{m-1})$ over $\mathbb{F}_q$ of degree at most $d-1$ in each variable, and given $\alpha_i \in \mathbb{F}_q^m$ for $i = 0, \ldots, N-1$, compute $f(\alpha_i)$ for $i = 0, \ldots, N-1$. Using this reduction, an algorithm for MULTIVARIATE MULTIPOINT EVALUATION that is optimal up to lower order terms yields an algorithm for MODULAR COMPOSITION that is optimal up to lower order terms.

Unfortunately, MULTIVARIATE MULTIPOINT EVALUATION does not seem susceptible to the techniques successfully used to obtain near-optimal (up to polylogarithmic factor) algorithms for the *univariate* case, and in general seems to be a more challenging problem. In fact, prior to this paper, there were only two nontrivial algorithms for MULTIVARIATE MULTIPOINT EVALUATION. First, Nüsken & Ziegler [NZ04] gave an algorithm for the bivariate case that can be generalized to yield an algorithm with operation count $O(d^{(\omega_2/2)(m-1)+1})$ times lower order terms, but this is not sufficient to make any gains over Huang & Pan's algorithm for MODULAR COMPOSITION via the reduction. Second, Umans [Uma08] gave an algorithm that uses a somewhat intricate lifting method using the p-power Frobenius, for p the characteristic of $\mathbb{F}_q$. This operation count for this algorithm is optimal up to lower order terms for $m \leq d^{o(1)}$, but it only works in small characteristic $p \leq d^{o(1)}$.

This paper gives a new algorithm for MULTIVARIATE MULTIPOINT EVALUATION over any field $\mathbb{F}_q$ (when $m \leq d^{o(1)}$) with running time $(d^m + N)^{1+\delta} \log^{1+o(1)} q$ (for any constant $\delta > 0$ and sufficiently large d) that is optimal up to lower order terms. Via the reduction, this yields an algorithm for MODULAR COMPOSITION whose running time is optimal up to lower order terms. We describe the main idea next, for the case when $q = p$ is prime; the reduction from the general case to this case uses similar ideas.

1.2 Our techniques

A basic observation when considering algorithms for MULTIVARIATE MULTIPOINT EVALUATION is that *if* the evaluation points happen to be all of $\mathbb{F}_p^m$, then they can be computed all at once via the multidimensional FFT, with an operation count that is best-possible up to logarithmic factors. More generally, if the evaluation points happen to be *well-structured* in the sense of being all of S^m for some subset $S \subseteq \mathbb{F}_p$, then by viewing $\mathbb{F}_p[X_1, X_2, \ldots, X_m]$ as $\mathbb{F}_p[X_1, X_2, \ldots, X_{m-1}][X_m]$ and applying an algorithm for univariate multipoint evaluation, and repeating m times, one can achieve an essentially optimal algorithm. But these are both very special cases, and the general difficulty with MULTIVARIATE MULTIPOINT EVALUATION is contending with highly unstructured sets of evaluation points in $\mathbb{F}_p^m$.

Our main idea is to use *multimodular reduction* to transform an arbitrary set of evaluation points into a "structured" one to which the FFT solution can be applied directly. We lift f and each evaluation point α_i to the integers by identifying the field $\mathbb{F}_p$ with the set $\{0, \ldots, p-1\}$. We can then compute the multipoint evaluation by doing so over $\mathbb{Z}$ and reducing modulo p. To actually compute the evaluation over $\mathbb{Z}$, we reduce modulo several smaller primes $p_1, \ldots, p_k$, producing separate instances of MULTIVARIATE MULTIPOINT EVALUATION over $\mathbb{F}_{p_i}$ for $i = 1, \ldots, k$. After solving these instances, we reconstruct the original evaluations using the Chinese Remainder Theorem.

This multimodular reduction can be applied recursively, with the primes in each round shrinking until they reach $p^* \approx (md)$ in the limit. By this last round, the evaluation points have been "packed" so tightly into the domain $\mathbb{F}_{p^*}^m$ that we can apply the FFT to obtain *all* evaluations in $\mathbb{F}_{p^*}^m$ with little loss: d^m operations are required just to read the input polynomial, and the FFT part of our algorithm requires only about $(dm)^m$ operations (and recall our requirement that $m < d^{o(1)}$).

To obtain our most general result, we may need to apply three rounds of multimodular reduction; for the application to MODULAR COMPOSITION, only two rounds are needed, making the algorithm quite practical.

We remark that our algorithm can be used in the univariate ($m = 1$) case (via a simple transformation to the $m \gg 1$ case; see the proof of Corollary 3.5). The overall algorithm requires only elementary modular arithmetic in $\mathbb{Z}$, and the FFT. Thus, our algorithm may be competitive, in simplicity and speed, with the "classical" algorithm for univariate multipoint evaluation (see any standard textbook, e.g., [vzGG99]). One striking contrast with the classical algorithm is that after a preprocessing step we can achieve $\mathrm{poly}(\log n, \log q)$ *actual* time for each evaluation (as opposed to amortized time); this can be interpreted as giving a powerful data structure supporting polynomial evaluation queries (see Section 4).

1.3 Why wasn't this algorithm discovered earlier?

In retrospect, our approach is quite simple, and, we believe, natural. Certainly this is not the first algorithm to employ multimodular reduction, or even recursive multimodular reduction. We point out three conceptual barriers that (possibly) explain why the overall algorithm and approach may have been harder to find than it appears with the benefit of hindsight.

First, there is a tendency to try to find algebraic algorithms for algebraic problems; our gains come from allowing nonalgebraic operations.

Second, the original MODULAR COMPOSITION problem is *not* amenable to multimodular reduction, because in the integers, the output of a lifted modular composition problem is longer than the input by a factor of n, rather than a negligible factor of dm that appears after applying the reduction to MULTIVARIATE MULTIPOINT EVALUATION. Thus the reduction to MULTIVARIATE MULTIPOINT EVALUATION (which only appeared in the last year) is more than just a convenience; it is *critical* for the multimodular approach to succeed.

Finally, we benefit from multimodular reduction for a quite different reason than other algorithms that employ this technique. Typically, multimodular reduction is used to reduce the "word size", when computing with large word sizes would be prohibitive or spoil the target complexity. In our case we are perfectly happy computing with word size $\log q$, so the multimodular reduction provides no benefit there. What it does do, however, is "pack" the evaluation points into a smaller and smaller space, and it does so extremely efficiently (requiring only local computations on each point). Thus, we are benefitting from the aggregate effect of applying multimodular reduction to an entire *set*, rather than directly from the reduced word size.

1.4 Application to polynomial factorization

As noted above, MODULAR COMPOSITION is used as a black box in a number of important algorithms for polynomials over finite fields. The same is true for a related problem, MODULAR POWER PROJECTION, for which we also obtain a near-optimal algorithm in Section 5.2. As merely one example, we recall the case of factorization of degree n univariate polynomials[1].

Kaltofen & Shoup [KS98] show that an algorithm for modular composition requiring $f(n, q)$ bit operations gives rise to an algorithm for polynomial factorization requiring

$$n^{0.5+o(1)} f(n, q) + n^{1+o(1)} \log^{2+o(1)} q$$

[1] Because our algorithms are nonalgebraic, the running times in this paper count bit operations. Therefore, the reader familiar with the accounting in previous work, which counts arithmetic operations in the field, should expect to see an "extra" $\log q$ factor.

bit operations (this dependence on $f(n, q)$ is worked out explicitly in [Uma08]). Using our algorithm for modular composition, we thus obtain an algorithm for polynomial factorization requiring

$$(n^{1.5+o(1)} + n^{1+o(1)} \log q) \log^{1+o(1)} q$$

bit operations. By contrast, the best previous algorithms that work over arbitrary finite fields (von zur Gathen & Shoup [vzGS92] and Kaltofen & Shoup [KS98]) require $(n^{2+o(1)} + n^{1+o(1)} \log q) \log^{1+o(1)} q$ and $n^{1.815+o(1)} \log^{2+o(1)} q$ bit operations, respectively; we thus obtain an asymptotic improvement in the range $\log q < n$. (Again, this improvement had been obtained in [Uma08] under the additional restriction $p \le n^{o(1)}$, for p the characteristic of $\mathbb{F}_q$.)

In Section 6.1 we discuss two additional fundamental algorithms for which our results lead to faster algorithms: irreducibility testing, and computing minimal polynomials.

1.5 Structure of the paper

In Section 2, we give formal statements of the MODULAR COMPOSITION and MULTIVARIATE MULTIPOINT EVALUATION problems, and recall from [Uma08] the reduction of the former to the latter. In Section 3, we describe and analyze an algorithm for MULTIVARIATE MULTIPOINT EVALUATION, and in Section 4 we describe the data structure for polynomial evaluation arising from our algorithm. In Section 5, we analyze the resulting algorithm for MODULAR COMPOSITION, as well as an algorithm for MODULAR POWER PROJECTION obtained by a careful application of the transposition principle to the algebraic parts of our algorithm. In Section 6, we discuss applications (including polynomial factorization) and mention some further open problems.

2 Preliminaries

In this paper, R is an arbitrary commutative ring, unless otherwise specified. In our complexity estimates, we will use standard facts about fast polynomial arithmetic (cf. [vzGG99]). For cleaner statements, we sometimes omit floors and ceilings when dealing with them would be routine. We use $o(1)$ frequently in exponents. We will always write things so that the exponentiated quantity is an expression in a single variable x, and it is then understood that the $o(1)$ term is a quantity that goes to zero as x goes to infinity.

2.1 Problem statements

For ease of exposition, we restrict to the univariate version of MODULAR COMPOSITION, defined next, which is the one used in all applications we are aware of. One can also define a version in which f is a multivariate polynomial (as in [Uma08]), and our results extend easily to that problem.

Problem 2.1 (MODULAR COMPOSITION). *Given $f(X), g(X), h(X)$ in $R[X]$, each with degree at most $n-1$, and with the leading coefficient of h a unit in R, output $f(g(X)) \bmod h(X)$.*

The main insight in [Uma08] is that MODULAR COMPOSITION is reducible to MULTIVARIATE MULTIPOINT EVALUATION, defined next:

Problem 2.2 (MULTIVARIATE MULTIPOINT EVALUATION). *Given $f(X_0, \ldots, X_{m-1})$ in $R[X_0, \ldots, X_{m-1}]$ with individual degrees at most $d-1$, and evaluation points $\alpha_0, \ldots, \alpha_{N-1}$ in R^m, output $f(\alpha_i)$ for $i = 0, 1, 2, \ldots, N-1$.*

Most of our effort in this paper is focused on obtaining a nearly-optimal algorithm for MULTIVARIATE MULTIPOINT EVALUATION; namely, one that runs in time $(d^m + N)^{1+\delta} \log^{1+o(1)} |R|$ (for any constant $\delta > 0$ and sufficiently large d).

2.2 Useful facts

We will need the following number theory fact:

Lemma 2.3. *For all integers $N \geq 2$, the product of the primes less than or equal to $16 \log N$ is greater than N.*

The constant 16 is not optimal; the Prime Number Theorem implies that any constant $c > 1$ can be used for N above some bound depending on c.

Proof. The exponent of the prime p in the factorization of $n!$ equals $\sum_{i=1}^{\infty} \lfloor \frac{n}{p^i} \rfloor$ since this counts multiples of p, multiples of p^2, etc., in $\{1, \ldots, n\}$. This implies Kummer's formula

$$\binom{n}{m} = \prod_{p \leq n} p^{e_p}, \qquad e_p = \sum_{i=1}^{\infty} \left(\left\lfloor \frac{n}{p^i} \right\rfloor - \left\lfloor \frac{m}{p^i} \right\rfloor - \left\lfloor \frac{n-m}{p^i} \right\rfloor \right).$$

Note that $e_p \leq 1$ for $\sqrt{n} < p \leq n$, and $e_p \leq \log_p n$ for all p. From this, and the fact that $\binom{n}{\lfloor n/2 \rfloor} \geq \binom{n}{m}$ for all m, it follows that

$$\frac{2^n}{n+1} \leq \binom{n}{\lfloor n/2 \rfloor} \leq \left(\prod_{\sqrt{n} < p \leq n} p \right) n^{\sqrt{n}} \leq \left(\prod_{p \leq n} p \right) n^{\sqrt{n}}.$$

For $N \geq 50$, we have $2^n n^{-\sqrt{n}}/(n+1) \geq N$ for $n = \lfloor 16 \log N \rfloor$, so the claim follows. For $N < 50$, the claim may be checked by hand. □

We repeat the following definition from [Uma08]:

Definition 2.4. *The map $\psi_{h,\ell}$ from $R[X_0, X_1, \ldots, X_{m-1}]$ to $R[Y_{0,0}, \ldots, Y_{m-1,\ell-1}]$ is defined as follows. Given X^a, write a in base h: $a = \sum_{j \geq 0} a_j h^j$ and define the monomial*

$$M_a(Y_0, \ldots, Y_{\ell-1}) \stackrel{\text{def}}{=} Y_0^{a_0} Y_1^{a_1} \cdots Y_{\ell-1}^{a_{\ell-1}}.$$

The map $\psi_{h,\ell}$ sends X_i^a to $M_a(Y_{i,0}, \ldots, Y_{i,\ell-1})$ and extends multilinearly to $R[X_0, X_1, \ldots, X_{m-1}]$.

For a polynomial $f \in R[X_0, X_1, \ldots, X_{m-1}]$ with individual degrees at most $h^\ell - 1$, we have:

$$f(X_0, \ldots, X_{m-1}) = \psi_{h,\ell}(f)(X_0^{h^0}, X_0^{h^1}, \ldots, X_0^{h^{\ell-1}}, \cdots, X_{m-1}^{h^0}, X_{m-1}^{h^1}, \ldots, X_{m-1}^{h^{\ell-1}})$$

and in this sense the map ψ is the inverse of the Kronecker substitution. We will use this map to transform instances of MULTIVARIATE MULTIPOINT EVALUATION with parameters d, m, N into instances with parameters $d' = d^{1/c}, m' = cm, N$ by applying $\psi_{d',c}$ and mapping each evaluation point $\alpha = (\alpha_0, \ldots, \alpha_{m-1}) \in R^m$ to the evaluation point

$$\alpha' = (\alpha_0^{d'^0}, \alpha_0^{d'^1}, \alpha_0^{d'^2}, \ldots, \alpha_0^{d'^{c-1}}, \cdots, \alpha_m^{d'^0}, \alpha_m^{d'^1}, \alpha_m^{d'^2}, \ldots, \alpha_m^{d'^{c-1}})$$

in $R^{m'}$.

2.3 The reduction

In Section 5.2 we consider the "transpose" of MODULAR COMPOSITION, which is used in particular in [Sho99]. We argue that despite the fact that our algorithm for MULTIVARIATE MULTIPOINT EVALUATION (and hence for MODULAR COMPOSITION via the reduction) is nonalgebraic, we can employ the transposition principle to obtain a similarly fast algorithm for the transpose problem. Doing so requires discussing the individual steps of the reduction, so we repeat it here (specialized to the univariate version of MODULAR COMPOSITION).

Theorem 2.5 ([Uma08])**.** *Given $f(X), g(X), h(X)$ in $R[X]$ each with degree at most $n-1$, and with the leading coefficient of h a unit in R, there is, for every integer $d > 0$, an algorithm that outputs $f(g(X)) \bmod h(X)$ in time*

$$O(nm^2d^2 \log^{1+o(1)} |R|) \cdot \text{poly}\log(n, m, d) + T(d, m, N)$$

(where $m = \lceil \log_d n \rceil$, $N = d^m md \le nmd^2$, and $T(d, m, N)$ is the time to solve MULTIVARIATE MULTIPOINT EVALUATION *with parameters d, m, N), provided that the algorithm is supplied with N distinct elements of R whose differences are units in R.*

Proof. Set $n' = d^m \le nd$. We perform the following steps:

1. Compute $f' = \psi_{d,m}(f)$.
2. Compute $g_i(X) \overset{\text{def}}{=} g(X)^{d^i} \bmod h(X)$ for $i = 0, 1, \ldots, m-1$.
3. Select $N = n'md$ distinct elements of R, $\beta_0, \ldots, \beta_{N-1}$, whose differences are units in R. Compute $\alpha_{i,j} \overset{\text{def}}{=} g_i(\beta_j)$ for $i = 0, 1, \ldots, m-1$ and $j = 0, 1, \ldots, N-1$.
4. Compute $f'(\alpha_{0,j}, \ldots, \alpha_{m-1,j})$ for $j = 0, 1, \ldots, N-1$.
5. Interpolate to recover $f'(g_0(X), \ldots, g_{m-1}(X))$ (which is a univariate polynomial of degree less than N) from these evaluations.
6. Output the result modulo $h(X)$.

Correctness follows from the observation that

$$f'(g_0(X), \ldots, g_{m-1}(X)) \equiv f(g(X)) \pmod{h(X)}.$$

Step 1 takes $O(n' \log(|R|))$ time. Using repeated squaring, Step 2 incurs complexity at most

$$O(n \log n \log^{1+o(1)} |R|) \cdot \log(n')$$

to compute each of the m polynomials g_i. Step 3 incurs complexity $O(mN \log^2 N \log^{1+o(1)} |R|)$ using fast multipoint evaluation for univariate polynomials. Step 4 invokes an algorithm for MULTIVARIATE MULTIPOINT EVALUATION at a cost of $T(d, m, N)$. Step 5 incurs complexity $O(N \log^2 N \log^{1+o(1)} |R|)$ using fast univariate interpolation, and Step 6 incurs complexity $O(N \log N \log^{1+o(1)} |R|)$. □

3 Fast multivariate multipoint evaluation

We describe our algorithm for MULTIVARIATE MULTIPOINT EVALUATION, first for prime fields, then for rings $\mathbb{Z}/r\mathbb{Z}$, and then for extension rings (and in particular, all finite fields).

3.1 Prime fields

For prime fields, we have a straightforward algorithm that uses fast Fourier transforms. The dependence on the field size p is quite poor, but we will remove that in our final algorithm using multimodular reductions.

Theorem 3.1. *Given an m-variate polynomial $f(X_0, \dots, X_{m-1}) \in \mathbb{F}_p[X_0, \dots, X_{m-1}]$ (p prime) with degree at most $d-1$ in each variable, and $\alpha_0, \dots, \alpha_{N-1} \in \mathbb{F}_p^m$, there exists a deterministic algorithm that outputs $f(\alpha_i)$ for $i = 0, \dots, N-1$ in*

$$O(m(d^m + p^m + N)\operatorname{poly}(\log p))$$

bit operations.

Proof. We perform the following steps to compute $f(\alpha_i)$ for $i = 0, \dots, N-1$.

1. Compute the reduction $\overline{f}$ of f modulo $X_j^p - X_j$ for $j = 0, \dots, m-1$.
2. Use a fast Fourier transform[2] to compute $\overline{f}(\alpha) = f(\alpha)$ for all $\alpha \in \mathbb{F}_p^m$.
3. Look up and return $f(\alpha_i)$ for $i = 0, \dots, N-1$.

In Step 1, the reductions modulo $X_j^p - X_j$ may be performed using md^m arithmetic operations in $\mathbb{F}_p$, for a total complexity of $O(md^m \operatorname{poly}(\log p))$.

In Step 2, we may perform the FFTs one variable at a time for a total time of $O(mp^m \operatorname{poly}(\log p))$. The details follow: we will give a recursive procedure for computing evaluations of an m-variate polynomial with individual degrees at most $p-1$ over all of $\mathbb{F}_p^m$, in time $m \cdot O(p^m \operatorname{poly}(\log p))$. When $m = 1$, we apply fast (univariate) multipoint evaluation at a cost of $O(p \operatorname{poly}(\log p))$. For $m > 1$, write $\overline{f}(X_0, X_1, \dots, X_{m-1})$ as $\sum_{i=0}^{p-1} X_0^i f_i(X_1, \dots, X_{m-1})$, and for each f_i, recursively compute its evaluations at all of $\mathbb{F}_p^{m-1}$ in time $(m-1) \cdot O(p^{m-1} \operatorname{poly}(\log p))$. Finally, for each $\beta \in \mathbb{F}_p^{m-1}$ evaluate the univariate polynomial $\sum_{i=0}^{p-1} X_0^i f_i(\beta)$ at all of $\mathbb{F}_p$ at a cost of $O(p \operatorname{poly}(\log p))$, again using fast (univariate) multipoint evaluation. The overall time is

$$(m-1) \cdot O(p^{m-1} \operatorname{poly}(\log p)) \cdot p + O(p \operatorname{poly}(\log p)) \cdot p^{m-1},$$

which equals $m \cdot O(p^m \operatorname{poly}(\log p))$ as claimed.

In Step 3, we look up N entries from a table of length p^m, for a total complexity of $O(mN \operatorname{poly}(\log p))$. This gives the stated complexity. □

3.2 Rings of the form $\mathbb{Z}/r\mathbb{Z}$

We now apply multimodular reduction recursively to remove the suboptimal dependence on p. Our main algorithm for rings $\mathbb{Z}/r\mathbb{Z}$ (r arbitrary) appears below. It accepts an additional parameter t which specifies how many rounds of multimodular reduction should be applied.

[2] We need the *finite field* Fourier transform here, since we care about evaluations over $\mathbb{F}_p$.

Algorithm MULTIMODULAR$(f, \alpha_0, \ldots, \alpha_{N-1}, r, t)$

where f is a m-variate polynomial $f(x_0, \ldots, x_{m-1}) \in (\mathbb{Z}/r\mathbb{Z})[x_0, \ldots, x_{m-1}]$ with degree at most $d-1$ in each variable, $\alpha_0, \ldots, \alpha_{N-1}$ are evaluation points in $(\mathbb{Z}/r\mathbb{Z})^m$, and t is the number of rounds.

1. Construct the polynomial $\tilde{f}(X_0, \ldots, X_{m-1}) \in \mathbb{Z}[X_0, \ldots, X_{m-1}]$ from f by replacing each coefficient with its lift in $\{0, \ldots, r-1\}$. For $i = 0, \ldots, N-1$, construct the m-tuple $\tilde{\alpha}_i \in \mathbb{Z}^m$ from α_i by replacing each coordinate with its lift in $\{0, \ldots, r-1\}$.
2. Compute the primes $p_1, \ldots, p_k$ less than or equal to $\ell = 16 \log(d^m (r-1)^{md})$, and note that $k \le \ell$.
3. For $h = 1, \ldots, k$, compute the reduction $f_h \in \mathbb{F}_{p_h}[X_0, \ldots, X_{m-1}]$ of $\tilde{f}$ modulo p_h. For $h = 1, \ldots, k$ and $i = 0, \ldots, N-1$, compute the reduction $\alpha_{h,i} \in \mathbb{F}_{p_h}^m$ of $\tilde{\alpha}_i$ modulo p_h.
4. If $t = 1$, then for $h = 1, \ldots, k$, apply Theorem 3.1 to compute $f_h(\alpha_{h,i})$ for $i = 0, \ldots, N-1$; otherwise if $t > 1$, then run MULTIMODULAR$(f_h, \alpha_{h,0}, \ldots, \alpha_{h,N-1}, p_h, t-1)$ to compute $f_h(\alpha_{h,i})$ for $i = 0, \ldots, N-1$.
5. For $i = 0, \ldots, N-1$, compute the unique integer in $\{0, \ldots, (p_1 p_2 \cdots p_k) - 1\}$ congruent to $f_h(\alpha_{h,i})$ modulo p_h for $h = 1, \ldots, k$, and return its reduction modulo r.

To bound the running time it will be convenient to define the function

$$\lambda_i(x) = x \log x \log\log x \log\log\log x \cdots \log^{(i-1)}(x).$$

Note that $\lambda_i(x) \le x(\log x)^{\log^* x} = x^{1+o(1)}$ (where $\log^* x$ denotes the least nonnegative integer i such that $\log^{(i)}(x) \le 1$) and that $\lambda_i(x) \le \lambda_j(x)$ for positive x and $i < j \le log^* x$.

Theorem 3.2. *Algorithm* MULTIMODULAR *returns $f(\alpha_i)$ for $i = 0, 1, \ldots, N-1$, and it runs in*

$$O((\lambda_t(d)^m + N)\lambda_t(\log r)\lambda_t(d)^t \lambda_t(m)^{m+t+1}) \cdot O(\log^{(t)} r)^m \cdot \operatorname{poly}\log(md \log r)$$

bit operations.

Proof. Correctness follows from the fact that $0 \le \tilde{f}(\tilde{\alpha}_i) \le d^m (r-1)^{md} < p_1 \cdots p_k$ by Lemma 2.3, and Theorem 3.1.

Observe that in the i-th level of recursion, the primes p_h have magnitude at most $\ell_i = O(\lambda_i(m)\lambda_i(d) \log^{(i)} r)$. For convenience, set $\ell_0 = 1$.

At the i-th level of the recursion tree, the algorithm is invoked at most $\ell_0 \ell_1 \ell_2 \cdots \ell_{i-1}$ times. Each invocation incurs the following costs from the steps before and after the recursive call in Step 4. Step 1 incurs complexity at most $O((d^m + mN)\ell_i)$. Step 2 incurs complexity $O(\ell_i \log \ell_i)$ using the Sieve of Eratosthenes (cf. [Sho08, §5.4]). Step 3 incurs complexity $O((d^m + mN)\ell_i \operatorname{poly}(\log \ell_i))$ by using remainder trees to compute the reductions modulo $p_1, \ldots, p_k$ all at once [Ber, §18], [vzGG99, Theorem 10.24]. Step 5 incurs complexity $O(N\ell_i \operatorname{poly}(\log \ell_i))$ as in [Ber, §23] or [vzGG99, Theorem 10.25]. At the last level (the t-th level) of the recursion tree when the FFT is invoked, Step 4 incurs complexity $O((d^m + \ell_t^m + N)m\ell_t \operatorname{poly}(\log \ell_t))$.

Thus, using the fact that $\operatorname{poly}\log(\ell_i) \leq \operatorname{poly}\log(md\log r)$ for all i, each invocation at level $i < t$ uses

$$O((d^m + N)m\ell_i) \cdot \operatorname{poly}\log(md\log r)$$

operations while each invocation at level t uses

$$O((d^m + \ell_t^m + N)m\ell_t) \cdot \operatorname{poly}\log(md\log r)$$

operations. There are a total of $\ell_0\ell_1\ell_2\cdots\ell_{i-1}$ invocations at level i. The total number of operations is thus

$$\left(\ell_1\ell_2\cdots\ell_t \cdot O((d^m + \ell_t^m + N)m) + \sum_{i=1}^{t-1} \ell_1\ell_2\cdots\ell_i \cdot O((d^m + N)m)\right) \cdot \operatorname{poly}\log(md\log r)$$

which is at most

$$\begin{aligned} &O(\ell_1\ell_2\cdots\ell_t) \cdot O((d^m + \ell_t^m + N)m) \cdot \operatorname{poly}\log(md\log r) \\ &\qquad \leq O(\lambda_t(m)^t\lambda_t(d)^t\lambda_{t-1}(\log r)) \cdot O((d^m + \ell_t^m + N)m) \cdot \operatorname{poly}\log(md\log r) \\ &\qquad \leq O((\lambda_t(d)^m + N)\lambda_{t-1}(\log r)\lambda_t(d)^t\lambda_t(m)^{m+t+1}) \cdot O(\log^{(t)} r)^m \cdot \operatorname{poly}\log(md\log r) \end{aligned}$$

operations over all t levels. The bound in the theorem statement follows. □

Plugging in parameters, we find that this yields an algorithm whose running time is optimal up to lower order terms, when $m \leq d^{o(1)}$.

Corollary 3.3. *For every constant $\delta > 0$ there is an algorithm for* MULTIVARIATE MULTIPOINT EVALUATION *over $\mathbb{Z}/r\mathbb{Z}$ with running time $(d^m + N)^{1+\delta}\log^{1+o(1)} r$, for all d, m, N with d sufficiently large and $m \leq d^{o(1)}$.*

Proof. Let c be a sufficiently large constant (depending on δ). We may assume $m > c$ by applying the map from Definition 2.4, if necessary, to produce an equivalent instance of MULTIVARIATE MULTIPOINT EVALUATION with more variables and smaller individual degrees. Now if $\log^{(3)} r < m$, then we choose $t = 3$, which gives a running time of

$$O((d^{(1+o(1))m} + N)d^3m^{m(1+o(1))(1+4/c)}(\log r)^{1+o(1)}) \cdot O(m)^m \cdot \operatorname{poly}\log(md\log r),$$

which simplifies to the claimed bound using $m \leq d^{o(1)}$. Otherwise, $\log^{(3)} r \geq m$, and we choose $t = 2$, which gives a running time of

$$O((d^{(1+o(1))m} + N)d^2m^{m(1+o(1))(1+3/c)}(\log r)^{1+o(1)}) \cdot O(\log^{(2)} r)^{\log^{(3)} r} \cdot \operatorname{poly}\log(md\log r),$$

which simplifies to the claimed bound, using $m \leq d^{o(1)}$ and $O(\log^{(2)} r)^{\log^{(3)} r} \leq O(\log^{o(1)} r)$.

□

3.3 Extension rings

Using algorithm MULTIMODULAR and some additional ideas, we can handle extension rings, and in particular, all finite fields. The strategy is to lift to $\mathbb{Z}[Z]$, then evaluate at $Z = M$ and reduce modulo r' for suitably large integers M, r'. Our algorithm follows:

Algorithm MULTIMODULAR-FOR-EXTENSION-RING$(f, \alpha_0, \ldots, \alpha_{N-1}, t)$

where R is a finite ring of cardinality q given as $(\mathbb{Z}/r\mathbb{Z})[Z]/(E(Z))$ for some monic polynomial $E(Z)$ of degree e, f is an m-variate polynomial $f(X_0, \ldots, X_{m-1}) \in R[X_0, \ldots, X_{m-1}]$ with degree at most $d-1$ in each variable, $\alpha_0, \ldots, \alpha_{N-1}$ are evaluation points in R^m, and $t > 0$ is the number of rounds.

Put $M = d^m(e(r-1))^{(d-1)m+1} + 1$ and $r' = M^{(e-1)dm+1}$.

1. Construct the polynomial $\tilde{f}(X_0, \ldots, X_{m-1}) \in \mathbb{Z}[Z][X_0, \ldots, X_{m-1}]$ from f by replacing each coefficient with its lift which is a polynomial of degree at most $e-1$ with coefficients in $\{0, \ldots, r-1\}$. For $i = 0, \ldots, N-1$, construct the m-tuple $\tilde{\alpha}_i \in \mathbb{Z}[Z]^m$ from α_i by replacing each coordinate with its lift which is a polynomial of degree at most $e-1$ with coefficients in $\{0, \ldots, r-1\}$.

2. Compute the reduction $\overline{f} \in (\mathbb{Z}/r'\mathbb{Z})[X_0, \ldots, X_{m-1}]$ of $\tilde{f}$ modulo r' and $Z - M$. For $i = 0, \ldots, N-1$, compute the reduction $\overline{\alpha}_i \in (\mathbb{Z}/r'\mathbb{Z})^m$ of $\tilde{\alpha}_i$ modulo r' and $Z - M$. Note that the reductions modulo r' don't do anything computationally, but are formally needed to apply Algorithm MULTIMODULAR, which only works over finite rings $\mathbb{Z}/r\mathbb{Z}$.

3. Run MULTIMODULAR$(\overline{f}, \overline{\alpha}_0, \overline{\alpha}_1, \ldots, \overline{\alpha}_{N-1}, r', t)$ to compute $\beta_i = \overline{f}(\overline{\alpha}_i)$ for $i = 0, \ldots, N-1$.

4. For $i = 0, \ldots, N-1$, compute the unique polynomial $Q_i[Z] \in \mathbb{Z}[Z]$ of degree at most $(e-1)dm$ with coefficients in $\{0, \ldots, M-1\}$ for which $Q_i(M)$ has remainder β_i modulo $r' = M^{(e-1)dm+1}$, and return the reduction of Q_i modulo r and $E(Z)$.

Theorem 3.4. *Algorithm* MULTIMODULAR-FOR-EXTENSION-RING *returns $f(\alpha_i)$ for $i = 0, 1, \ldots, N-1$, and it runs in*

$$O((\lambda_t(d)^m + N)\lambda_t(\log q)\lambda_t(d)^{t+2}\lambda_t(m)^{m+t+3}) \cdot O(\log^{(t-1)}(d^2 m^2 \log q \log\log q))^m \cdot \operatorname{poly}\log(md \log q)$$

bit operations.

Proof. To see that the algorithm outputs $f(\alpha_i)$ for $i = 0, \ldots, N-1$, note that $\tilde{f}(\tilde{\alpha}_i) \in \mathbb{Z}[Z]$ has nonnegative coefficients and its degree is at most $(e-1)dm$. Moreover, the value at $Z = 1$ of each coordinate of $\tilde{\alpha}_i$ and each coefficient of $\tilde{f}$ is at most $e(r-1)$, so $\tilde{f}(\tilde{\alpha}_i)(1) \leq d^m(e(r-1))^{(d-1)m+1} = M - 1$. In particular, each coefficient of $\tilde{f}(\tilde{\alpha}_i)$ belongs to $\{0, \ldots, M-1\}$. We now see that the polynomials $\tilde{f}(\tilde{\alpha}_i), Q_i \in \mathbb{Z}[Z]$ both have degree at most $(e-1)dm$ and coefficients in $\{0, \ldots, M-1\}$, and their evaluations at $Z = M$

are congruent modulo $r' = M^{(e-1)dm+1}$. This implies that the polynomials coincide, so the reduction of Q_i modulo r and $E(Z)$ agrees with the corresponding reduction of $\tilde{f}(\tilde{\alpha}_i)$, which equals $f(\alpha_i)$.

We expect a $\log q = \log(r^e)$ term in the running time, and recall that Algorithm MULTIMODULAR is invoked over a ring of cardinality $r' = M^{(e-1)(d-1)m+1}$. We have:

$$\begin{aligned} \log r' = \log(M^{(e-1)(d-1)m+1}) &\leq (e-1)dm \log(d^m(e(r-1))^{(d-1)m+1} + 1) \\ &\leq O(ed^2m^2(\log e + \log r)) \\ &\leq O(\log q \log\log q)d^2m^2. \end{aligned} \tag{1}$$

The dominant step is step 3, whose complexity is (by Theorem 3.2)

$$O((\lambda_t(d)^m + N)\lambda_t(\log r')\lambda_t(d)^t\lambda_t(m)^{m+t+1}) \cdot O(\log^{(t)} r')^m \cdot \operatorname{poly}\log(md\log r'),$$

which, using (1) above, yields the stated complexity. □

Similar to Corollary 3.3, we obtain:

Corollary 3.5. *For every constant $\delta > 0$ there is an algorithm for* MULTIVARIATE MULTIPOINT EVALUATION *over any ring $(\mathbb{Z}/r\mathbb{Z})[Z]/(E(Z))$ of cardinality q, with running time $(d^m + N)^{1+\delta}\log^{1+o(1)} r$, for all d, m, N with d sufficiently large and $m \leq d^{o(1)}$.*

Proof. The proof is the same as the proof of Corollary 3.3, except the two cases depend on m in relation to the quantity r' appearing in the proof of Theorem 3.4. The argument in the proof of Corollary 3.3 yields the claimed running time with r' in place of q; we then use the inequality $\log r' \leq O(\log q \log\log q)d^2m^2$. □

4 A data structure for polynomial evaluation

In this section we observe that it is possible to interpret our algorithm for MULTIVARIATE MULTIPOINT EVALUATION as a data structure supporting rapid "polynomial evaluation" queries.

Consider a degree n univariate polynomial $f(X) \in \mathbb{F}_q[X]$ (and think of q as being significantly larger than n). If we store f as a list of n coefficients, then to answer a single evaluation query $\alpha \in \mathbb{F}_q$ (i.e. return the evaluation $f(\alpha)$), we need to look at all n coefficients, requiring $O(n \log q)$ bit operations. On the other hand, a *batch* of n evaluation queries $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ can be answered all at once using $O(n \log^2 n)$ $\mathbb{F}_q$-operations, using fast algorithms for univariate multipoint evaluation (cf. [vzGG99]). This is often expressed by saying that the *amortized time* for an evaluation query is $O(\log^2 n)$ $\mathbb{F}_q$-operations. Can such a result be obtained in a non-amortized setting? Certainly, if we store f as a table of its evaluations in $\mathbb{F}_q$, then a single evaluation query $\alpha \in \mathbb{F}_q$ can be trivially answered in $O(\log q)$ bit operations. However, the stored data is highly redundant; it occupies space $q \log q$, when information-theoretically $n \log q$ should suffice.

By properly interpreting our algorithm for MULTIVARIATE MULTIPOINT EVALUATION, we arrive at a data structure that achieves "the best of both worlds": we can preprocess the n coefficients describing f in nearly-linear time, to produce a nearly-linear size data structure T from which we can answer evaluation queries in time that is polynomial in $\log n$ and $\log q$. This is a concrete benefit of our approach to multipoint evaluation even for the univariate case, as it seems impossible to obtain anything similar by a suitable re-interpretation of previously known algorithms for univariate multipoint evaluation.

Theorem 4.1. *Let $R = (\mathbb{Z}/r\mathbb{Z})[Z]/(E(Z))$ be a ring of cardinality q, and let $f(X) \in R[X]$ be a degree n polynomial. Choose any constant $\delta > 0$. For sufficiently large n, one can compute from the coefficients of f in time at most*

$$T = n^{1+\delta} \log^{1+o(1)} q$$

a data structure of size at most T with the following property: there is an algorithm that given $\alpha \in \mathbb{F}_q$, computes $f(\alpha)$, in time

$$\operatorname{poly}\log n \cdot \log^{1+o(1)} q$$

with random access to the data structure.

Proof. We will choose parameters d, m such that $d^m = n$, and apply map $\psi_{d,m}$ from Definition 2.4 to f.

Then, given this m-variate polynomial f, algorithm MULTIMODULAR-FOR-EXTENSION-RING computes $\overline{f}$ with coefficients in $\mathbb{Z}/r'\mathbb{Z}$. This is followed by t rounds of multimodular reduction which produce *reduced polynomials* $f_{p_1,p_2,\ldots,p_t} \in \mathbb{F}_{p_t}[X]$ for certain sequences $p_1, p_2, \ldots, p_t$ of primes (the p_i are the moduli in the t rounds of multimodular reduction). Each $f_{p_1,p_2,\ldots,p_t}$ is evaluated over its entire domain $\mathbb{F}_{p_t}^m$ using the multidimensional FFT. The key observation is that these computations do not depend on the evaluation points, and can thus comprise a preprocessing phase that produces the data structure consisting of tables of evaluations of each $f_{p_1,p_2,\ldots,p_t}$.

Using notation from the proof of Theorem 3.2, there are at most $\ell_1\ell_2\cdots\ell_t$ reduced polynomials, each p_t has magnitude at most ℓ_t, and it holds that $\ell_i = O(\lambda_i(m)\lambda_i(d)\log^{(i)} r')$. Referring to the proof of Theorem 3.1, we see that the cost incurred to produce the required tables of evaluations is at most

$$\begin{aligned} T &= \ell_1\ell_2\cdots\ell_t \cdot O(m\ell_t^m) \cdot \operatorname{poly}\log(\ell_t) \\ &\leq O(\lambda_t(m)^{t+m+1}\lambda_t(d)^{t+m}\lambda_{t-1}(\log r')) \cdot (\log^{(t)} r')^m \cdot \operatorname{poly}\log(md\log r') \end{aligned}$$

At this point, an evaluation query $\alpha \in R$ can be answered from the tables by first computing the point $(\alpha, \alpha^d, \ldots, \alpha^{d^{m-1}}) \in R^m$, then (as in algorithm MULTIMODULAR-FOR-EXTENSION-RING) lifting each coordinate to $\mathbb{Z}/r'\mathbb{Z}$ and finally applying t rounds of multimodular reduction, to produce *reduced evaluation points* $\alpha_{p_1,p_2,\ldots,p_t} \in \mathbb{F}_{p_t}^m$. The desired evaluations $f_{p_1,p_2,\ldots,p_t}(\alpha_{p_1,p_2,\ldots,p_t})$ can be found in the precomputed tables, and then $f(\alpha)$ is reconstructed by t rounds of application of the Chinese Remainder Theorem. Again adopting the notation from the proof of Theorem 3.2, this reconstruction is invoked $\ell_1\ell_2\cdots\ell_{i-1}$ times at level i, each time with cost $O(\ell_i \operatorname{poly}\log(\ell_i))$. The overall cost for an evaluation query is thus

$$\begin{aligned} \sum_{i=1}^{t} \ell_1\ell_2\cdots\ell_{i-1} \cdot O(\ell_i \operatorname{poly}\log(\ell_i)) &\leq \sum_{i=1}^{t} \ell_1\ell_2\cdots\ell_i \cdot \operatorname{poly}\log(md\log r') \\ &\leq O(\ell_1\ell_2\cdots\ell_t) \cdot \operatorname{poly}\log(md\log r') \\ &\leq O(\lambda_t(m)^t\lambda_t(d)^t\lambda_{t-1}(\log r')) \cdot \operatorname{poly}\log(md\log r') \end{aligned}$$

It remains to choose the parameters d, m and t. If $r' > 2^{2^n}$, then we choose $d = n, m = 1, t = 2$; if $r' \leq 2^{2^n}$, then choose $d = \log^c n$ and $m = (\log n)/(c\log\log n)$ for a sufficiently large constant c, and $t = 4$. These choices give the claimed running times for preprocessing and queries, with r' in place of q. As in the proof of Theorem 3.4, we have $\log r' \leq O(\log q \log\log q)d^2m^2$, which completes the proof. □

Theorem 4.1 is surprising in light of a number of lower bounds for this problem under certain restrictions. For example, in the purely algebraic setting, and when the underlying field in $\mathbb{R}$, Belaga [Bel61] shows a lower bound on the query complexity of $\lfloor \frac{3n}{2} \rfloor + 1$ (and Pan [Pan66] has given a nearly-matching upper bound). Miltersen [Mil95] proves that the trivial algorithm (with query complexity n) is essentially optimal when the field size is exponentially large and the data structure is limited to polynomial size, and he conjectures that this lower bound holds for smaller fields as well (this is in an algebraic model that does not permit the modular operations we employ). Finally, Gál and Miltersen [GM07] show a lower bound of $\Omega(n/\log n)$ on the product of the *additive redundancy* (in the data structure size) and the query complexity, thus exhibiting a tradeoff that rules out low query complexity when the data structure is required to be very small (i.e., significantly smaller than $2n$).

5 Fast modular composition, and its transpose

We now obtain fast algorithms for MODULAR COMPOSITION and MODULAR POWER PROJECTION via the reduction of Theorem 2.5, and the transposition principle.

5.1 Modular composition

By applying the reduction in Theorem 2.5, we obtain a nearly-linear time algorithm for MODULAR COMPOSITION. We emphasize that to achieve this running time only requires invoking Algorithm MULTIMODULAR-FOR-EXTENSION-RING with $t = 2$, which makes the overall algorithm (arguably) practical and implementable. Indeed, use of a single round of multimodular reduction is quite common in practice; for instance, Shoup's NTL library [Sho] uses multimodular reduction for most basic arithmetic involving multiprecision integer polynomials.

Theorem 5.1. *Let R be a finite ring of cardinality q given as $(\mathbb{Z}/r\mathbb{Z})[Z]/(E(Z))$ for some monic polynomial $E(Z)$. For every $\delta > 0$, if we have access to $n^{1+O(\delta)}$ distinct elements of R whose differences are units in R, then there is an algorithm for* MODULAR COMPOSITION *over R running in $n^{1+\delta}\log^{1+o(1)} q$ bit operations, for sufficiently large n.*

Proof. Let c be a a sufficiently large constant (depending on δ), and set $d = n^{1/c}$ and $m = c$. Then applying Theorem 2.5, we obtain an algorithm for MODULAR COMPOSITION with running time $n^{1+2/c}\log^{1+o(1)} q \cdot \text{poly}\log(n, m, d) + T(d, m, N)$, where $N \leq nmd^2 \leq cn^{1+2/c}$, and $T(d, m, N)$ is the time for MULTIVARIATE MULTIPOINT EVALUATION with parameters d, m, N. We solve this instance via Theorem 3.4 with $t = 2$. □

Corollary 5.2. *For every $\delta > 0$, there is an algorithm for* MODULAR COMPOSITION *over $\mathbb{F}_q$ running in $n^{1+\delta}\log^{1+o(1)} q$ bit operations, for sufficiently large n.*

Proof. Construct an extension field $\mathbb{F}_{q'}$ of $\mathbb{F}_q$ with cardinality at least $n^{1+O(\delta)}$, then apply Theorem 5.1 with $R = \mathbb{F}_{q'}$. □

Remark. In the running times claimed in Corollaries 3.3, 3.5, 5.2, and Theorem 5.1, we have chosen to present bounds that interpret "almost linear in x" as meaning "for all $\delta > 0$, there is an algorithm running in time $x^{1+\delta}$ for sufficiently large x." In all cases, it is possible to choose δ to be a sub-constant function of the other parameters, giving stronger, but messier, bounds.

5.2 Fast modular power projection

In this section, we consider the "transpose" of MODULAR COMPOSITION, defined next:

Problem 5.3 (MODULAR POWER PROJECTION). *Given a linear form $\pi : R^n \to R$, and polynomials $g(X), h(X)$ in $R[X]$, each with degree at most $n-1$, and with the leading coefficient of h a unit in R, output $\pi(g(X)^i \bmod h(X))$ for $i = 0, 1, \ldots, n-1$.*

One can view MODULAR COMPOSITION as multiplying the $n \times 1$ column vector of coefficients of f on the left by the $n \times n$ matrix $A_{g,h}$, whose columns are the coefficients of $g(X)^i \bmod h(X)$ for $i = 0, 1, \ldots, n-1$. Then MODULAR POWER PROJECTION is the problem of multiplying the column vector of coefficients of π on the left by the transpose of $A_{g,h}$.

By a general argument (the "transposition principle"), *linear* straight-line programs computing a linear map yield linear straight-line programs with essentially the same complexity for computing the transposed map.

Theorem 5.4 ([BCS97, Thm. 13.20]). *Let $\phi : R^n \to R^m$ be a linear map that can be computed by a linear straight-line program of length L and whose matrix in the canonical basis has z_0 zero rows and z_1 zero columns. Then the transposed map $\phi^t : R^m \to R^n$ can be computed by a linear straight-line program of size $L - n + m - z_0 + z_1$.*

If our algorithm for MODULAR COMPOSITION computed only linear forms in the coefficients of polynomial f then we would have a similarly fast algorithm for MODULAR POWER PROJECTION via the above theorem. Unfortunately, the lifting to characteristic 0 followed by modular reduction is not algebraic, and so we cannot apply Theorem 5.4 directly. However, with some care, we can isolate the nonalgebraic parts of the algorithm into preprocessing and postprocessing phases, and apply the transposition principle to algebraic portions of the algorithm. Before considering MODULAR POWER PROJECTION, we consider the transpose of MULTIVARIATE MULTIPOINT EVALUATION.

Theorem 5.5. *Let R be a finite ring of cardinality q given as $(\mathbb{Z}/r\mathbb{Z})[Z]/(E(Z))$ for some monic polynomial $E(Z)$. There is an algorithm for the transpose of* MULTIVARIATE MULTIPOINT EVALUATION *with parameters satisfying $N = d^m$, with running time at most that claimed in Theorem 3.4.*

Proof. We view Algorithm MULTIMODULAR-FOR-EXTENSION-RING as computing the linear map $\phi : R^{d^m} \to R^N$ which computes the evaluations of f at evaluation points $\alpha_0, \alpha_1, \ldots, \alpha_{N-1}$. This is computed by a preprocessing phase (Steps 1 and 2), which produces $\overline{f}$ and $\overline{\alpha}_0, \overline{\alpha}_1, \ldots, \overline{\alpha}_{N-1}$, with the coefficients of $\overline{f}$ and the coordinates of each $\overline{\alpha}_i$ in $\mathbb{Z}/r'\mathbb{Z}$. Algorithm MULTIMODULAR then computes in t successive multimodular reductions a collection of instances of MULTIVARIATE MULTIPOINT EVALUATION over $\mathbb{F}_p$, for small primes p. Each of these is a map from $\phi_p : \mathbb{F}_p^{d^m} \to \mathbb{F}_p^N$, which is computed rapidly using Theorem 3.1. The transpose map ϕ_p can be computed in the same time bound, by Theorem 5.4, or directly by observing that the transpose of the DFT computed in Step 2 in the proof of Theorem 3.1 can again be computed rapidly using the FFT.

In the original algorithm, a postprocessing phase (successive applications of Step 5 of Algorithm MULTIMODULAR) we recover the evaluations of $\overline{f}$ in t successive rounds of reconstruction using the Chinese Remainder Theorem. Finally the evaluations of f are reconstructed in Step 4 of Algorithm MULTIMODULAR-FOR-EXTENSION-RING. In our algorithm for the transpose problem ϕ^t, we perform the same successive rounds of reconstructions applied to the output from computing the various ϕ_p^t maps.

In the original problem, correctness in each round of reconstruction comes from choosing primes for each multimodular reduction whose product exceeded the magnitude of any evaluation in $\mathbb{Z}$. We argue correctness of these successive rounds of reconstruction in the transpose problem by noting that the magnitude calculation is the same for the transpose problem, when $N = d^m$. This is because the bound is calculated as the product of the number of coefficients of the polynomial (d^m) and the maximum magnitude of any matrix entry in the matrix representation of the linear map. For the transpose problem, a valid bound is the product of N times the maximum magnitude of any matrix entry of the transposed matrix, which is the same. □

Theorem 5.6. *Let R be a finite ring of cardinality q given as $(\mathbb{Z}/r\mathbb{Z})[Z]/(E(Z))$ for some monic polynomial $E(Z)$. For every $\delta > 0$, if we have access to $n^{1+O(\delta)}$ distinct elements of R whose differences are units in R, then there is an algorithm for* MODULAR POWER PROJECTION *over R running in $n^{1+\delta} \log^{1+o(1)} q$ bit operations, for sufficiently large n.*

Proof. Consider first the reduction from MODULAR COMPOSITION to MULTIVARIATE MULTIPOINT EVALUATION of Theorem 2.5. An instance of MODULAR COMPOSITION is specified by degree n polynomials $f(X), g(X), h(X)$. We describe the reduction as the product of linear maps applied to the vector of coefficients of f. Steps 2 and 3 do not involve f, and can be executed in a preprocessing phase.

Step 1 is given by $\phi_1 : R^n \to R^{n'}$ which maps f to f' by permuting the coefficients and padding with 0's. Step 4 is given by $\phi_4 : R^{n'} \to R^N$ which maps f' to its evaluations at the N evaluation points (the α's). Step 5 is given by $\phi_5 : R^N \to R^N$ which maps these evaluations to the coefficients of the univariate polynomial having these values at the βs. Step 6 is given by $\phi_6 : R^N \to R^n$ which maps the resulting degree $N-1$ univariate polynomial to its reduction modulo $h(X)$. All of $\phi_1, \phi_4, \phi_5, \phi_6$ are linear maps, and thus the overall algorithm for MODULAR COMPOSITION (after the preprocessing phase involving $g(X)$ and $h(X)$) can be described as the linear map $\phi_6 \circ \phi_5 \circ \phi_4 \circ \phi_1 : R^n \to R^n$.

We are interested in computing the transposed map $\phi_1^t \circ \phi_4^t \circ \phi_5^t \circ \phi_6^t : R^n \to R^n$. We argue that transposed map can be computed in time comparable to the time required for the nontransposed map. In Theorem 2.5, ϕ_6 is computed rapidly using fast polynomial division with remainder. By the transposition principle (Theorem 5.4), ϕ_6^t can be computed in comparable time. In Theorem 2.5, ϕ_5 is computed rapidly using fast univariate polynomial interpolation. By the transposition principle (Theorem 5.4), ϕ_5^t can be computed in comparable time.

In Theorem 2.5, ϕ_4 is computed rapidly by invoking a fast algorithm for MULTIVARIATE MULTIPOINT EVALUATION. We claim that ϕ_4^t can be computed in the time expended by Algorithm MULTIMODULAR-FOR-EXTENSION-RING to compute ϕ_4. We'd like to apply Theorem 5.5, but that requires $N = d^m$, and in our case N is larger by a factor of dm. But, just as we could have computed ϕ_4 by invoking Algorithm MULTIMODULAR-FOR-EXTENSION-RING dm times with d^m evaluation points each time, we can compute ϕ_4^t by computing the transpose of a dm square instances (via Theorem 5.5) and summing the resulting vectors.

Finally, ϕ_1^t is just a projection followed by a permutation of the coordinates, which can trivially be computed in time comparable to that required for computing ϕ_1. □

Remark. There are explicit algorithms known for ϕ_5^t (transposed univariate interpolation) and ϕ_6^t (transposed univariate polynomial division with remainder) (see, e.g., [BLS03]), and our algorithm in Theorem 5.5 is also explicit. Thus we have an explicit algorithm for MODULAR POWER PROJECTION (whereas in general, use of the transposition principle may produce an algorithm that can only be written down by manipulating the linear straight-line program).

6 Conclusions

We conclude by outlining some applications of our new algorithms, and open problems.

6.1 Applications

Fast algorithms for MODULAR COMPOSITION and MODULAR POWER PROJECTION give rise to improvements in various basic operations with polynomials over finite fields, as indicated already in [Uma08]. Here is an incomplete but indicative list of such problems, with the dependence on the running times for MODULAR COMPOSITION and MODULAR POWER PROJECTION made explicit. Below we use $C(n, q)$ and $P(n, q)$ for the number of bit operations required for MODULAR COMPOSITION and MODULAR POWER PROJECTION, respectively (operating on degree n polynomials, over $\mathbb{F}_q$).

- Univariate polynomial factorization. We are given $f(X) \in \mathbb{F}_q[X]$ of degree n and we must output the irreducible factors. Variants of the Cantor-Zassenhaus method break this problem into three stages: square-free factorization, distinct-degree factorization, and equal-degree factorization. Yun's algorithm for the first stage takes $n^{1+o(1)} \log^{2+o(1)} q$ bit operations; Kaltofen & Shoup's algorithm for the second stage [KS98] takes $n^{0.5+o(1)}C(n, q) + n^{1+o(1)} \log^{2+o(1)} q$ bit operations; von zur Gathen & Shoup's randomized algorithm for the third stage [vzGS92] takes $O(C(n, q)) + n^{1+o(1)} \log^{2+o(1)} q$ bit operations. Thus with our algorithm for MODULAR COMPOSITION, we obtain a randomized algorithm that takes
$$(n^{1.5+o(1)} + n^{1+o(1)} \log q) \log^{1+o(1)} q$$
bit operations for the polynomial factorization problem.

- Irreducibility testing. We are given $f(X) \in \mathbb{F}_q[X]$ of degree n, and we want to determine whether or not it is irreducible. Rabin's algorithm [Rab80] can be implemented to take $(n^{1+o(1)}) \log^{1+o(1)} q + C(n, q) \log^2 n$ bit operations, so we obtain a running time of $n^{1+o(1)} \log^{1+o(1)} q$, which is best-possible up to lower order terms.

- Computing minimal polynomials. We are given $g(X), h(X) \in \mathbb{F}_q[X]$, both of degree at most n, and we must output the minimal polynomial of $g(X)$ in the ring $\mathbb{F}_q[X]/(h(X))$; i.e., the monic polynomial $f(X)$ of minimal degree for which $f(g(X)) \bmod h(X) = 0$. Shoup's randomized algorithm [Sho99] runs in expected time $(n+C(n, q)+P(n, q))n^{o(1)}$, so we obtain a running time of $n^{1+o(1)} \log^{1+o(1)} q$, which is best possible up to lower order terms.

The fact that our algorithm applies to extension rings leads to some additional applications. For instance, if $P(X) \in (\mathbb{Z}/p^n\mathbb{Z})[X]$ is a monic polynomial whose reduction modulo p is monic (of the same degree) and irreducible, then the ring $R = (\mathbb{Z}/p^n\mathbb{Z})[X]/(P(X))$ admits a unique *Frobenius automorphism* $F : R \to R$ satisfying $F(r) \equiv r^p \pmod{p}$ for all $r \in R$. Once one has computed $F(X)$, one can then evaluate F efficiently using modular composition. Such rings R arise as quotients of unramified extensions of the ring of p-adic integers; consequently, fast Frobenius evaluation leads to improvements in certain algorithms based on p-adic numbers. An explicit example was suggested by Hendrik Hubrechts, in his use of deformations in p-adic Dwork cohomology to compute zeta functions of hyperelliptic curves over finite fields; use of our algorithms leads to a runtime improvement by substituting for our modular composition algorithm in [Hub, §6.2].

6.2 Open problems

We briefly mention some open problems. Our algorithm for MULTIVARIATE MULTIPOINT EVALUATION is only optimal up to lower order terms in case $m \leq d^{o(1)}$. It would be interesting to describe a near-optimal algorithm in the remaining cases, or perhaps just the multilinear case to start. It would also be satisfying to give a near-optimal *algebraic* algorithm for MULTIVARIATE MULTIPOINT EVALUATION in arbitrary characteristic ([Uma08] does so for the case of small characteristic).

As noted earlier, the reduction from MODULAR COMPOSITION to MULTIVARIATE MULTIPOINT EVALUATION plays an important role in our work because it is easier to control the growth of integers when solving the lifted version of MULTIVARIATE MULTIPOINT EVALUATION. One wonders whether there are other problems involving polynomials that can exploit the combination of transforming the problem to a multivariate version with smaller total degree, and then lifting to characteristic zero followed by multimodular reduction.

Finally, the reduction to MULTIVARIATE MULTIPOINT EVALUATION can be seen, loosely, as a generalization of the "baby steps/giant steps" approach of [BK78]. We wonder whether this generalization can improve algorithms for other problems whose currently best algorithms use a "baby steps/giant steps" technique, such as *automorphism projection* and *automorphism evaluation* as discussed in [KS98].

7 Acknowledgements

We thank Swastik Kopparty and Madhu Sudan for some references mentioned in Section 4, and Ronald de Wolf and the FOCS 2008 referees for helpful comments.

References

[BCS97] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1997.

[Bel61] E. G. Belaga. Evaluation of polynomials of one variable with preliminary preprocessing of the coefficients. *Problemy Kibernet.*, 5:7–15, 1961.

[Ber] D. J. Bernstein. Fast multiplication and its applications (version of 7 Oct 2004). Preprint available at `http://cr.yp.to/papers.html#multapps`.

[BK78] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *J. ACM*, 25(4):581–595, 1978.

[BLS03] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *ISSAC '03: Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, pages 37–44, New York, NY, USA, 2003. ACM.

[GM07] A. Gál and P. B. Miltersen. The cell probe complexity of succinct data structures. *Theor. Comput. Sci.*, 379(3):405–417, 2007.

[HP98] X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *J. Complexity*, 14(2):257–299, 1998.

[Hub] H. Hubrechts. Point counting in families of hyperelliptic curves (version of 31 Mar 2007). Preprint available at `http://wis.kuleuven.be/algebra/hubrechts/`.

[KS98] E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. *Mathematics of Computation*, 67(223):1179–1197, 1998.

[Mil95] P. B. Miltersen. On the cell probe complexity of polynomial evaluation. *Theor. Comput. Sci.*, 143(1):167–174, 1995.

[NZ04] M. Nüsken and M. Ziegler. Fast multipoint evaluation of bivariate polynomials. In Susanne Albers and Tomasz Radzik, editors, *ESA*, volume 3221 of *Lecture Notes in Computer Science*, pages 544–555. Springer, 2004.

[Pan66] V. Ya. Pan. Methods of computing values of polynomials. *Russian Math. Surveys*, 21(1):105–136, 1966.

[Rab80] M. O. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comput.*, 9(2):273–280, 1980.

[Sho] V. Shoup. NTL 5.4.2. Available at `http://www.shoup.net/ntl/`.

[Sho94] V. Shoup. Fast construction of irreducible polynomials over finite fields. *J. Symb. Comput.*, 17(5):371–391, 1994.

[Sho99] V. Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *ISSAC*, pages 53–58, 1999.

[Sho08] V. Shoup. *A Computational Introduction to Number Theory and Algebra (version 2.3)*. Cambridge University Press, 2008. Available at `http://www.shoup.net/ntb/`.

[Uma08] C. Umans. Fast polynomial factorization and modular composition in small characteristic. In Richard E. Ladner and Cynthia Dwork, editors, *STOC*, pages 481–490. ACM, 2008.

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

[vzGS92] J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Computational Complexity*, 2:187–224, 1992.