The Parallel Complexity of Exponentiating Polynomials over Finite Fields

Faith E. Fich
Martin Tompa

Department of Computer Science, FR-35
University of Washington
Seattle, WA 98195 U.S.A.

1 Exponentiation Problems

There are a number of important algebraic problems that enjoy efficient sequential solutions, but for which the existence of much more efficient parallel solutions is currently unresolved. To put this rather imprecise statement in technical terms, those problems that we wish to consider are in $\mathcal{P}$, but it is not known whether or not they are in $\mathcal{NC}$. Two such problems are particularly important because of their many applications. They are the computation of greatest common divisors, and modular integer exponentiation. This work is motivated by the latter.

Given n-bit integers a, e, and m, the problem of *Modular Integer Exponentiation* is to compute $a^e \bmod m$. To illustrate its many applications, we list here some problems that are $\mathcal{NC}$-reducible to Modular Integer Exponentiation (but whose membership in $\mathcal{NC}$ is likewise unresolved):

1. primality testing (by a slight variation of the Miller-Rabin algorithm [14,16]; in this application, the reduction to Modular Integer Exponentiation is probabilistic),
2. computation of the encryption and decryption functions in Rivest, Shamir, and Adleman's public key cryptosystem [17],
3. inverse computation modulo any prime,
4. Chinese remaindering with respect to prime moduli [3],
5. quadratic residuosity modulo any prime [3], and
6. extraction of square roots modulo any prime congruent to 3 mod 4 [15].

It is easy to show that there is a polynomial time algorithm for Modular Integer Exponentiation [3], but it is unknown whether it is in $\mathcal{NC}$, complete for $\mathcal{P}$, or neither [6]. Von zur Gathen [11] approached this problem by showing that it does have an extremely efficient parallel solution for the special case in which all prime factors of the modulus m are polynomial in n; note, though, that this is not the case in any of the applications listed in the previous paragraph.

Borodin (see [11]) proposed a variant of the problem that is closer to ordinary integer exponentiation: given n-bit integers t, a, and e, the *Integer Exponentiation* problem is to compute the t^{th} bit of a^e. It is no longer clear that this problem is even in $\mathcal{P}$, since the repeated squaring method that works for Modular Integer Exponentiation would have to contend with the exponential length of intermediate values.

The n low order bits of a^e can be determined by computing $a^e \bmod 2^n$, using von zur Gathen's algorithm. Alt [1] has shown that the high order bits of a^e can also be computed quickly in parallel. This is done by using a truncated Taylor series to approximately compute $\ln a$, multiplying by e, and then using a truncated Taylor series to approximately compute $\exp(e \ln a)$).

In this work, we seek a better understanding of the parallel complexities of these exponentiation problems by transcribing them from the integer domain into a polynomial domain: if one thinks of the n bits of a (and the n bits of m, in the Modular Integer Exponentiation problem) as n coefficients of a polynomial over Z_2 (the ring of integers modulo 2), the transcribed problems are polynomial exponentiation problems over Z_2. The benefit of this transcription is that it eliminates the complication of "carries" which arise in the integer versions, allowing us to concentrate on the fundamental exponentiation operation.

These polynomial problems will now be defined precisely, doing so with respect to a general finite field F rather than the specific field Z_2:

- *Modular Polynomial Exponentiation:* Given polynomials $a(x), m(x) \in F[x]$, each of degree at most n, and an n-bit integer e, compute $(a(x))^e \bmod m(x) \in F[x]/(m(x))$.

- *Polynomial Exponentiation:* Given a polynomial $a(x) \in F[x]$ of degree at most n, and n-bit integers e and t, compute the coefficient of x^t in $(a(x))^e \in F[x]$.

As further motivation for studying exponentiation problems in this polynomial domain, note that the Modular Polynomial Exponentiation problem arises in Berlekamp's algorithm for factoring polynomials over finite fields [5]. Von zur Gathen [9] demonstrates a random $\mathcal{NC}$ algorithm for factoring polynomials over finite fields along these lines.

Modular Polynomial Exponentiation certainly has a polynomial time solution, analogous to the repeated squaring solution for Modular Integer Exponentiation. Polynomial Exponentiation, on the other hand, is like Integer Exponentiation, in the sense that a polynomial time solution is not evident, due to the exponential degree of intermediate polynomials. To make matters worse, Rabin and Valiant [personal communication] have observed that the following variant of Polynomial Exponentiation is $\#\mathcal{P}$-complete:

1. The computation takes place over the integers.
2. The polynomial $a(x)$ may have degree up to 2^n, but still has only n nonzero terms.
3. The exponent e, rather than having n bits, need only have value n.

(Details of this result are given in Section 7.)

Section 3 describes a method for solving the Modular Polynomial Exponentiation problem, demonstrating that for fields whose characteristic is polynomial in the problem size, this problem is in $\mathcal{NC}^3$. If the order of the field is restricted to be less than some constant, the problem is in $\mathcal{NC}^2$. In the special case when the modulus $m(x)$ is irreducible over Z_p, this problem is equivalent to exponentiating elements in the large finite field $Z_p[x]/(m(x))$. It is interesting to contrast this with one of von zur Gathen's results [11, theorem 2.3], which shows the impossibility of such a fast parallel algorithm for exponentiation in large finite fields when the elements of that field must be treated atomically. The result of Section 3 has applications to inverse computation, quadratic residuosity, and the extraction of square roots in large finite fields.

In Sections 4 and 5, it will be demonstrated that the Polynomial Exponentiation problem over Z_2 not only has a polynomial time solution, but is in fact in $\mathcal{NC}^2$. This result will follow from a reduction to the problem of counting the number of representations of t in a certain system of redundant representations of the integers, a combinatorial problem that is interesting in its own right. This result is generalized in Section 3 to fields of constant order. We also show that, for fields whose characteristic is polynomial in the input size, the Polynomial Exponentiation problem is in $\mathcal{NC}^3$. the problem is in $\mathcal{NC}^2$.

2 Preliminaries

We begin with a short description of notation and some properties of finite fields. Throughout the paper, $\mathcal{N}$ denotes the set of nonnegative integers and Z denotes the ring of integers.

If F is a finite field, we use p to denote its characteristic and $q = p^\ell$ to denote its order. The finite field of order p is Z_p, the ring of integers modulo p. The usual way to represent an element of this field is as an integer between 0 and $p-1$, written in binary.

The field of order p^ℓ, for $\ell > 1$, can be described as $Z_p[y]/(g(y))$ where $g(y) \in Z_p[y]$ is a monic irreducible polynomial of degree ℓ. Elements of this field are represented by polynomials in $Z_p[y]$ of degree at most $\ell - 1$; i.e. if $f \in Z_p[y]/(g(y))$, then $f = \sum_{i=0}^{\ell-1} f_i y^i$ where $f_i \in Z_p$ for $0 \le i \le \ell - 1$. We specify a field as part of the input by providing p and $g(y)$.

The key property of finite fields that we require in this paper is their "nice" binomial theorem:

Proposition 1 Let F be a field with characteristic p. If $a(x)$ and $b(x)$ are elements of $F[x]$, then $(a(x)+b(x))^p = (a(x))^p + (b(x))^p$.

Proof: For $0 < i < p$, $\binom{p}{i}$ is a multiple of p, and hence is 0 in F. □

Corollary 2 Let F be a field with characteristic p. If $a(x) = \sum_{i=0}^{d} a_i x^i \in F[x]$ and $k \in \mathcal{N}$, then $(a(x))^{p^k} = \sum_{i=0}^{d} (a_i)^{p^k} x^i$.

Proof: This follows by induction from Proposition 1. □

Corollary 3 If F is a field of order q and $a(x) \in F[x]$, then $(a(x))^{q^k} = a(x^{q^k})$ for all $k \in \mathcal{N}$.

Proof: This follows from Corollary 2 and the fact that $a^q = a$ for all $a \in F$. □

We use uniform Boolean circuits of polynomial size as our model of parallel computation. We measure complexity in terms of the depth of these circuits as a function of n, a parameter describing the size of the input.

The following result of Eberly [7] concerning polynomial arithmetic over finite fields is important for analysing the complexity of our algorithms.

Theorem 4 Let F be a field of order q such that $\log q = n^{O(1)}$. Then the multiplication of $n^{O(1)}$ polynomials in $F[x]$ and the division with remainder of two polynomials in $F[x]$, where the polynomials have degree less than $n^{O(1)}$, can be implemented using uniform Boolean circuits of depth $O(\log n \log\log n)$ and polynomial size.

3 Modular Polynomial Exponentiation

In [5], Berlekamp describes an important method for computing $(a(x))^q \bmod m(x)$, where F is a finite field of order q, $m(x) \in F[x]$, d is the degree of $m(x)$, and $a(x) = \sum_{i=0}^{d-1} a_i x^i \in F[x]/(m(x))$. Specifically, he constructs a $d \times d$ matrix Q whose i^{th} row $(Q_{i,0}, Q_{i,1}, \ldots, Q_{i,d-1})$ is the sequence of coefficients of the polynomial $\sum_{j=0}^{d-1} Q_{i,j} x^j \equiv x^{iq} \bmod m(x)$. Then the elements of the row vector $(a_0, a_1, \ldots, a_{d-1}) \times Q$ are the coefficients of $(a(x))^q \bmod m(x)$, because

$$\begin{aligned}(a(x))^q &= \sum_{i=0}^{d-1} a_i x^{iq}, \text{ by Corollary 3} \\ &\equiv \sum_{i=0}^{d-1} \left(a_i \sum_{j=0}^{d-1} Q_{i,j} x^j \right) \bmod m(x), \text{ by definition of } Q \\ &= \sum_{j=0}^{d-1} \left(\sum_{i=0}^{d-1} a_i Q_{i,j} \right) x^j.\end{aligned}$$

More generally, the elements of the row vector

$$(a_0, a_1, \ldots, a_{d-1}) \times Q^k$$

are the coefficients of $(a(x))^{q^k} \bmod m(x)$. By multiplying together an appropriate combination of these powers of $a(x)$, any power of $a(x)$ can be obtained. When the order of the field, q, is polynomial in n, this is an efficient parallel method for performing modular polynomial exponentiation. With minor modifications, this technique can be adapted so that only the characteristic p must be polynomial in n.

Let R be the $d \times d$ matrix whose i^{th} row

$$(R_{i,0}, R_{i,1}, \ldots, R_{i,d-1})$$

is the sequence of coefficients of the polynomial $\sum_{j=0}^{d-1} R_{i,j} x^j \equiv x^{ip} \bmod m(x)$. (Recall that p is the characteristic of F.) Define $R^{(k)}$ to be the the matrix obtained by raising each entry of R to the power p^k i.e. $R^{(k)}_{i,j} = (R_{i,j})^{p^k}$. Finally, let $M^{(k)} = R^{(k-1)} \times \cdots \times R^{(1)} \times R^{(0)}$.

Lemma 5 The i^{th} row $\left(M^{(k)}_{i,0}, M^{(k)}_{i,1}, \ldots, M^{(k)}_{i,d-1}\right)$ of the matrix $M^{(k)}$ is the sequence of coefficients of the polynomial $\sum_{j=0}^{d-1} M^{(k)}_{i,j} x^j \equiv x^{ip^k} \bmod m(x)$.

Proof: The proof is by induction on k. If $k = 1$, then $M^{(1)} = R$ and the lemma is true by the definition of R.

Now let $k \geq 1$ and assume that the i^{th} row

$$\left(M^{(k)}_{i,0}, M^{(k)}_{i,1}, \ldots, M^{(k)}_{i,d-1}\right)$$

of the matrix $M^{(k)}$ is the sequence of coefficients of the polynomial $\sum_{j=0}^{d-1} M^{(k)}_{i,j} x^j \equiv x^{ip^k} \bmod m(x)$. Since $M^{(k+1)} = R^{(k)} \times M^{(k)}$, it follows that

$$\begin{aligned}\sum_{j=0}^{d-1} M^{(k+1)}_{i,j} x^j &= \sum_{j=0}^{d-1} \left(\sum_{h=0}^{d-1} R^{(k)}_{i,h} M^{(k)}_{h,j} \right) x^j \\ &= \sum_{h=0}^{d-1} (R_{i,h})^{p^k} \left(\sum_{j=0}^{d-1} M^{(k)}_{h,j} x^j \right) \\ &\equiv \sum_{h=0}^{d-1} (R_{i,h})^{p^k} x^{hp^k} \bmod m(x) \\ &= \left(\sum_{h=0}^{d-1} R_{i,h} x^h \right)^{p^k}, \text{ by Corollary 2} \\ &\equiv \left(x^{ip} \right)^{p^k} \bmod m(x), \text{ by definition of } R \\ &= x^{ip^{k+1}}.\end{aligned}$$

Thus the lemma is true for $k+1$. By induction, it is true for all $k \geq 1$. □

Corollary 6 If $a(x) = \sum_{i=0}^{d-1} a_i x^i$ then the elements of the row vector $\left((a_0)^{p^k}, (a_1)^{p^k}, \ldots, (a_{d-1})^{p^k}\right) \times M^{(k)}$ are the coefficients of the polynomial $(a(x))^{p^k} \bmod m(x)$.

Proof:

$$\begin{aligned}(a(x))^{p^k} &= \sum_{i=0}^{d-1} (a_i)^{p^k} x^{ip^k} \text{ by Corollary 2} \\ &\equiv \sum_{i=0}^{d-1} (a_i)^{p^k} \left(\sum_{j=0}^{d-1} M^{(k)}_{i,j} x^j \right) \bmod m(x), \text{ by Lemma 5} \\ &= \sum_{j=0}^{d-1} \left(\sum_{i=0}^{d-1} (a_i)^{p^k} M^{(k)}_{i,j} \right) x^j. \quad \square\end{aligned}$$

Theorem 7 For any $n > 0$, given a field F of characteristic p and order $q = p^\ell$, polynomials $m(x) \in F[x]$, with degree d, and $a(x) = \sum_{i=0}^{d-1} a_i x^i \in F[x]/(m(x))$, and a positive binary integer e, such that p, ℓ, d, and $log(e)$ are polynomial in n, the problem of computing $(a(x))^e \bmod m(x)$ can be solved by a Boolean circuit of depth $O((\log n)^2 \log\log n))$ and polynomial size.

Proof: The computation consists of the following steps.

- Convert the binary integer e to base p so that $e = \sum_{k=0}^{r-1} e_k p^k$ and $0 \leq e_k < p$. Since p and $\log e$ are polynomial in n, each p-ary digit of this representation can be computed in parallel by a Boolean circuit of depth $O(\log n \log\log n)$ [2].

- Compute the d rows of R in parallel. To compute $x^{ip} \bmod m(x)$, it suffices to divide the polynomial x^{ip}, which has degree at most $(d-1)p$, by the degree d polynomial $m(x)$. By Theorem 4, this can be accomplished by a Boolean circuit of depth $O(\log dp \log\log dp) = O(\log n \log\log n)$.

- In parallel, compute all of the entries of $R^{(k-1)}$ and the row vectors $\left((a_0)^{p^k}, \ldots, (a_{d-1})^{p^k}\right)$ for $1 \le k \le r-1$. Each of these computations consists of exponentiating a field element. Since $f^q = f$ for all $f \in F$, the exponents can all be assumed to be less than q.

  If $q = p$ (i.e. $F = Z_p$), then by Theorem 4.3 in [4], this computation can be performed by a Boolean circuit of depth $O(\log n)$

  Otherwise, $F \simeq Z_p[y]/(g(y))$ for some irreducible polynomial $g(y) \in Z_p[y]$ of degree ℓ and any element of the field F is actually a polynomial $f(y) \in Z_p[y]$ of degree less than ℓ. Thus, if our theorem is true for the field Z_p, then we may exponentiate $f(y)$ using a Boolean circuit of depth $O\left((\log n)^2 \log\log n\right)$.

- For all $1 \le k \le r-1$ in parallel, compute

$$\left((a_0)^{p^k}, \ldots, (a_{d-1})^{p^k}\right) \times M^{(k)}$$

  to obtain the coefficients of $(a(x))^{p^k} \bmod m(x)$. This can be done using a balanced tree of k matrix multiplications.

  Each matrix multiplication consists of $1 + \log d$ parallel steps; the first is the multiplication of pairs of field elements and the rest are the addition of pairs of field elements. The addition of two field elements can be performed by a Boolean circuit of depth $O(\log\log p) = O(\log\log n)$.

  If $F = Z_p$, the multiplication of two field elements can be accomplished by performing an integer multiplication followed by an integer division with remainder in depth $O(\log\log p \log\log\log p) = O(\log\log n \log\log\log n)$. (The integers involved have $O(\log p)$ bits.)

  Otherwise the multiplication of two field elements consists of a multiplication and a division with remainder of two polynomials of degree $O(\ell)$. By Theorem 4, this can be done in depth $O(\log n \log\log n)$. Hence, each matrix multiplication can be performed by a Boolean circuit of depth $O(\log n \log\log n)$. Since $k < r = n^{O(1)}$, the total depth necessary for a Boolean circuit to perform the matrix multiplications is $O\left((\log n)^2 \log\log n\right)$.

- Since $(a(x))^e = \prod_{k=0}^{r-1} \left((a(x))^{p^k}\right)^{e_k}$, the only remaining task is the multiplication of $\sum_{k=0}^{r-1} e_k = O(p \log e)$ polynomials in $F[x]/(m(x))$. By Theorem 4, this can be accomplished by a Boolean circuit of depth $O(\log n \log\log n)$ because p, $\log e$, and d are all $n^{O(1)}$. □

Corollary 8 The *Modular Polynomial Exponentiation Problem* is in $\mathcal{NC}^3$ for finite fields of polynomial characteristic.

Corollary 9 The *Modular Polynomial Exponentiation Problem* is in $\mathcal{NC}^2$ for finite fields whose order is $O(1)$.

Proof: Observe that, when the order of the field is $O(1)$, the addition and multiplication of field elements can be performed by Boolean circuits of constant depth. □

In the special case when $m(x)$ is monic and irreducible over Z_p, that is, $Z_p[x]/(m(x)) \cong \mathrm{GF}(p^d)$, Corollary 8 has applications to computations in $Z_p[x]/(m(x))$ that are analogous to the applications of Modular Integer Exponentiation outlined in Section 1. Three of these applications are listed below.

Proposition 10 Let p be prime and $m(x)$ be monic, irreducible over Z_p, and of degree n. Then for any $a(x) \in Z_p[x]/(m(x)) - \{0\}$, $(a(x))^{-1} = (a(x))^{p^d-2}$. Hence, computing inverses in $Z_p[x]/(m(x))$ is in $\mathcal{NC}^3$, provided $p = n^{O(1)}$.

Proposition 11 Let p be an odd prime, $m(x)$ be monic, irreducible over Z_p, and of degree n, and $a(x) \in Z_p[x]/(m(x)) - \{0\}$. Then $a(x)$ is a quadratic residue if and only if $(a(x))^{\frac{1}{2}(p^d-1)} = 1$. Hence, determining quadratic residuosity in $Z_p[x]/(m(x))$ is in $\mathcal{NC}^3$, provided $p = n^{O(1)}$.

Proposition 12 Let $p \equiv 3 \pmod 4$ be prime and $m(x)$ be monic, irreducible over Z_p, and of odd degree n. Let $a(x) \in Z_p[x]/(m(x))$ be a quadratic residue. Then $(a(x))^{\frac{1}{4}(p^d+1)}$ is a square root of $a(x)$. Hence, extracting square roots in $Z_p[x]/(m(x))$ is in $\mathcal{NC}^3$, provided $p = n^{O(1)}$.

4 Relating Polynomial Exponentiation to the Redundant Representations of Integers

For expository purposes, the results of this section and the next are presented with respect to the special case $F = Z_2$. In Section 6 we discuss how these results can be generalized to an arbitrary finite field whose characteristic is polynomial in the input size.

Given $e \in \mathcal{N}$, let $e = \sum_{i=0}^{r} e_i 2^i$, where $e_i \in \{0,1\}$. Then $e_{r-1} e_{r-2} \cdots e_0 \in \{0,1\}^r$ is the binary representation of e.

Corollary 13 For $a(x) \in Z_2[x]$, $(a(x))^e = \prod_{i=0}^{r-1} a(x^{2^i})^{e_i}$.

Proof: This follows directly from Corollaries 2 and 3. □

Example 1 This begins a running example of how to compute the coefficient of x^{33} in $(x^0 + x^1 + x^3 + x^4)^{11}$ over Z_2. To begin, Corollary 13 is applied to decompose $(x^0 + x^1 + x^3 + x^4)^{11}$. In this example $e = 11_{10} = 1011_2$, so

$$\begin{aligned}(x^0 + x^1 + x^3 + x^4)^{11} &= (x^0 + x^1 + x^3 + x^4)^{1\times 2^3} \times \\ &\quad (x^0 + x^1 + x^3 + x^4)^{0\times 2^2} \times \\ &\quad (x^0 + x^1 + x^3 + x^4)^{1\times 2^1} \times \\ &\quad (x^0 + x^1 + x^3 + x^4)^{1\times 2^0} \\ &= (x^0 + x^1 + x^3 + x^4)^8 \times \\ &\quad (x^0) \times \\ &\quad (x^0 + x^1 + x^3 + x^4)^2 \times \\ &\quad (x^0 + x^1 + x^3 + x^4)^1 \\ &= (x^0 + x^8 + x^{24} + x^{32}) \times \\ &\quad (x^0) \times \\ &\quad (x^0 + x^2 + x^6 + x^8) \times \\ &\quad (x^0 + x^1 + x^3 + x^4). \quad \square\end{aligned}$$

We now define the redundant representation system to which Section 1 alluded. The notation $D_i D_{i-1} \cdots D_0$ denotes the concatenation of the sets $D_i, \ldots, D_1, D_0$. The i-fold concatenation of D with itself is denoted by D^i.

If $y = y_{i-1} y_{i-2} \cdots y_0 \in \mathcal{N}^i$, define

$$\text{val}(y) = \sum_{j=0}^{i-1} y_j 2^j.$$

Fix sets $D_0, D_1, \ldots, D_i, \ldots \subseteq \mathcal{N}$. For $u, i \in \mathcal{N}$ and $y \in \{0,1\}^i$, define

$$C(u, y) = \#\{z \in D_i D_{i-1} \cdots D_0 \mid \text{val}(z) = \text{val}(uy)\}.$$

In words, $C(0, y)$ is the number of representations of the integer val(y) in a radix 2 system in which the set of allowable digits in the i^{th} least significant position is D_i.

Example 2 $\text{val}(4001) = 4 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 33$. Let $D_3 = D_1 = D_0 = \{0, 1, 3, 4\}$ and $D_2 = \{0\}$. Then $N(4, 001) = 3$, since there are 3 values of $z \in D_3 D_2 D_1 D_0$ with $\text{val}(z) = 33$, namely 4001, 3041, and 3033. □

Theorem 14 provides the promised reduction from the Polynomial Exponentiation problem to computation of the number of redundant representations of the integer t.

Theorem 14 Let $a(x) = a_0 + a_1 x + \cdots + a_d x^d \in Z_2[x]$, let $t, e \in \mathcal{N}$, and let $t_{m-1} t_{m-2} \cdots t_0 \in \{0,1\}^m$ and $e_{r-1} e_{r-2} \cdots e_0 \in \{0,1\}^r$ denote the binary representations of t and e, respectively. Define $e_i = 0$ if $i \geq r$. Then the coefficient of x^t in $\prod_{i \geq 0} a(x^{2^i})^{e_i}$ *when multiplied over* Z is $C(0, t_{m-1} t_{m-2} \cdots t_0)$ where, for all $i \in \mathcal{N}$, the set of allowable digits for the i^{th} position is

$$D_i = \begin{cases} \{k \mid a_k = 1\}, & \text{if } e_i = 1 \\ \{0\}, & \text{otherwise} \end{cases}.$$

Proof: Each term in the expansion of $\prod_{i \geq 0} a(x^{2^i})^{e_i}$ is obtained by selecting one term from each of the factors $a(x^{2^i})^{e_i}$ and multiplying them together. Note that if $e_i = 0$, then $a(x^{2^i})^{e_i} = 1 = x^0$ and, thus, x^0 is the only term which can be selected from this factor.

When $e_i = 1$, the set of terms of the factor $a(x^{2^i})^{e_i} = \sum_{k=0}^{d} a_k (x^{2^i})^k$ is $\{x^{k 2^i} \mid a_k = 1\}$. By our definition of the sets D_i, $\prod_{i \geq 0} x^{z_i 2^i}$ is a term in the expansion of $\prod_{i \geq 0} a(x^{2^i})^{e_i}$ if and only if $z_i \in D_i$ for all $i \geq 0$.

The coefficient of x^t in $\prod_{i \geq 0} a(x^{2^i})^{e_i}$ when multiplied over Z is the number of terms $\prod_{i \geq 0} x^{z_i 2^i}$ in the expansion of $\prod_{i \geq 0} a(x^{2^i})^{e_i}$ which satisfy $t = \sum_{i \geq 0} z_i 2^i$. For $i \geq r$, $e_i = 0$ and $z_i \in D_i = \{0\}$. Thus $t = \sum_{i=0}^{r-1} z_i 2^i = val(z)$. This is exactly the definition of $C(0, t_{m-1} t_{m-2} \cdots t_0)$. □

Example 3 Example 1 is continued here by demonstrating how to use Theorem 14 in order to compute the coefficient of x^{33} in

$$\begin{aligned}P(x) &= (x^0 + x^8 + x^{24} + x^{32}) \times \\ &\quad (x^0) \times \\ &\quad (x^0 + x^2 + x^6 + x^8) \times \\ &\quad (x^0 + x^1 + x^3 + x^4). \quad \square\end{aligned}$$

when multiplied over Z. In the statement of Theorem 14,

$a(x) = x^0 + x^1 + x^3 + x^4$,

$t = 33$, which is 100001 written in binary,

$e = 11$, which is 1011 written in binary,

$D_3 = D_1 = D_0 = \{0, 1, 3, 4\}$, and

$D_6 = D_5 = D_4 = D_2 = \{0\}$.

The conclusion of Theorem 14 is that the coefficient of x^{33} in $P(x)$ is $C(0, 100001)$. The next section shows how to compute this value, but for this small example it is easy to verify that $C(0, 100001) = 3$, using the same 3 representations as in Example 2. To understand at an intuitive level why Theorem 14 is true, note the following natural correspondences:

Representation	Monomial
4001	$(x^4)^8 \cdot (x^0)^4 \cdot (x^0)^2 \cdot (x^1)^1 = x^{32} \cdot x^0 \cdot x^0 \cdot x^1$
3041	$(x^3)^8 \cdot (x^0)^4 \cdot (x^4)^2 \cdot (x^1)^1 = x^{24} \cdot x^0 \cdot x^8 \cdot x^1$
3033	$(x^3)^8 \cdot (x^0)^4 \cdot (x^3)^2 \cdot (x^3)^1 = x^{24} \cdot x^0 \cdot x^6 \cdot x^3$

Thus, for example, the representation 3041 corresponds to selecting x^{24} from the first factor of $P(x)$, x^0 from the second, and x^8 from the third, and x^1 from the fourth. The product of these four selections contributes 1 to the coefficient of x^{33}, just as 3041 contributes 1 to the number of representations of 100001. □

Corollary 15 Let $a(x) \in Z_2[x]$, let $t, e \in \mathcal{N}$, and let $t_{m-1}t_{m-2}\cdots t_0 \in \{0,1\}^m$ be the binary representation of t. Then the coefficient of x^t in $(a(x))^e$ *when computed over* Z_2 is the parity of $C(0, t_{m-1}t_{m-2}\cdots t_0)$, where the set of allowable digits is as in Theorem 14.

Proof: This follows directly from Corollary 13 and Theorem 14. □

Example 4 Continuing Example 3, the coefficient of x^{33} in $(x^0 + x^1 + x^3 + x^4)^{11}$ when computed over Z_2 is $C(0, 100001) \bmod 2 = 1$. □

5 Counting Redundant Representations

Let $t_{m-1}t_{m-2}\cdots t_0 \in \{0,1\}^m$ be input together with $D_{m-1}, D_{m-2}, \ldots, D_0$, where $D_i \subseteq \{0, 1, \ldots, d\}$. (For convenience, assume that $D_j = \{0\}$ if $j \geq m$.) The goal of this section is to derive algorithms that compute the number $C(0, t_{m-1}t_{m-2}\cdots t_0)$ of radix 2 representations of $t = \text{val}(t_{m-1}t_{m-2}\cdots t_0)$ in which D_i is the set of allowable digits in the i^{th} significant position.

Theorem 16 provides a recurrence for computing $C(u, y)$, and is the basis for the efficient algorithms of this section. The intuition underlying the recurrence is very simple. Suppose $u \in \mathcal{N}$, $b \in \{0,1\}$, and $y \in \{0,1\}^i$, and the goal is to rewrite uby in other legal representations. For any $k \in D_{i+1}$, $u - k$ can be "borrowed" from the most significant position, thus replacing u and b by k and $b + 2(u-k)$, respectively.

Theorem 16 For all $u, i \in \mathcal{N}$, $b \in \{0,1\}$ and $y \in \{0,1\}^i$,

$$\begin{aligned} C(u,\epsilon) &= \begin{cases} 1 & \text{, if } u \in D_0 \\ 0 & \text{, otherwise} \end{cases} \\ C(u,by) &= \sum_{\substack{k \in D_{i+1} \\ k \leq u}} C(b + 2(u-k), y). \end{aligned}$$

Proof:

CASE 1 ($C(u,\epsilon)$):

$$\begin{aligned} C(u,\epsilon) &= \#\{z \in D_0 \mid \text{val}(z) = \text{val}(u)\} \\ &= \#\{z \in D_0 \mid z = u\}. \end{aligned}$$

CASE 2 ($C(u,by)$): Let $u, i \in \mathcal{N}$, $b \in \{0,1\}$, and $y \in \{0,1\}^i$. Suppose $k \in \mathcal{N}$ and $z \in \mathcal{N}^{i+1}$ satisfy $\text{val}(kz) = \text{val}(uby)$. Since $\text{val}(uby) < (u+1)2^{i+1}$ and $\text{val}(z) \geq 0$, $k \leq u$. Hence

$$\begin{aligned} C(u,by) &= \#\{kz \mid k \in D_{i+1} \text{ and } z \in D_iD_{i-1}\cdots D_0 \\ &\qquad \text{and } \text{val}(kz) = \text{val}(uby)\} \\ &= \sum_{\substack{k \in D_{i+1} \\ k \leq u}} \#\{z \in D_iD_{i-1}\cdots D_0 \mid \text{val}(kz) = \text{val}(uby)\} \\ &= \sum_{\substack{k \in D_{i+1} \\ k \leq u}} \#\{z \in D_iD_{i-1}\cdots D_0 \mid \\ &\qquad \text{val}(z) = (b + 2(u-k))2^i + \text{val}(y)\} \\ &= \sum_{\substack{k \in D_{i+1} \\ k \leq u}} C(b + 2(u-k), y). \quad \square \end{aligned}$$

In order to make the application of Theorem 16 more efficient, we observe in Lemma 17 that if the most significant digit u becomes too great, then $C(u,y) = 0$, simply because there is not enough "capacity" in the available positions using only allowable digits.

Lemma 17 If $u \geq 2d$ then $C(u,y) = 0$.

Proof: Let $i = |y|$. Then $\text{val}(uy) \geq u2^i \geq d2^{i+1}$, but

$$\max\{\text{val}(z) \mid z \in D_iD_{i-1}\cdots D_0\} \leq \sum_{j=0}^{i} d2^j < d2^{i+1}.$$

Hence,

$$\begin{aligned} C(u,y) &= \#\{z \in D_iD_{i-1}\cdots D_0 \mid \text{val}(z) = \text{val}(uy)\} \\ &= 0. \quad \square \end{aligned}$$

Theorem 16 and Lemma 17 together provide a polynomial time sequential algorithm for computing $C(u, t_{m-1}t_{m-2}\cdots t_0)$, using dynamic programming: for progressively longer suffixes y of $t_{m-1}t_{m-2}\cdots t_0$ and all $0 \leq u' < 2d$, use the recurrence of Theorem 16 to compute $C(u', y)$. This is illustrated in the following example.

Example 5 This example completes Example 3 by showing how to compute $C(0, 100001)$. In applying the recurrence of Theorem 16, recall that

$$D_3 = D_1 = D_0 = \{0,1,3,4\} \text{ and}$$

$$D_6 = D_5 = D_4 = D_2 = \{0\}.$$

The dynamic programming algorithm proceeds by completing the following table row by row:

y	$C(0,y)$	$C(1,y)$	$C(2,y)$	$C(3,y)$	$C(4,y)$	$C(5,y)$	$C(6,y)$	$C(7,y)$
ϵ	1	1	0	1	1	0	0	0
1	1	2	1	1	2	1	0	0
01	1	1	2	0	0	0	0	0
001	1	3	2	1	3	2	0	0
0001	1	2	3	0	0	0	0	0
00001	1	3	0	0	0	0	0	0
100001	3	0	0	0	0	0	0	0

As examples of how these entries are computed from the recurrence of Theorem 16,

$$\begin{aligned} C(4,001) &= C(0,01) + C(2,01) + C(6,01) + C(8,01) \\ &= 1 + 2 + 0 + 0 \\ &= 3 \end{aligned}$$

since $D_3 = \{0,1,3,4\}$, but

$$\begin{aligned} C(4,0001) &= C(8,001) \\ &= 0 \end{aligned}$$

since $D_4 = \{0\}$. □

The following theorem shows that the recurrence of Theorem 16 can be evaluated efficiently in parallel, using a technique similar to that used in Section 2.3.3 of [13].

Theorem 18 Given inputs $t_{n-1}t_{n-2}\cdots t_0 \in \{0,1\}^n$ and $D_{n-1}, D_{n-2}, \ldots, D_0$, where $D_j \subseteq \{0,1,\ldots,n\}$, the problem of computing $C(0, t_{n-1}t_{n-2}\cdots t_0)$ is in $\mathcal{DET}$, and hence in $\mathcal{NC}^2$. ($\mathcal{DET}$ is Cook's class of problems $\mathcal{NC}^1$-reducible to computing the determinant of an integer matrix. See [6] for a discussion.)

Proof: The intuition underlying the proof is as follows. Consider the row vector $V^{(y)} = (C(0,y), \ldots, C(2n-1,y))$ From the recurrence of Theorem 16 it follows that $V^{(ty)}$ can be computed simply by multiplying $V^{(y)}$ by a $2n \times 2n$ matrix with entries from $\{0,1\}$. Hence the problem of computing $C(0, t_{n-1}t_{n-2}\cdots t_0)$ is reduced to the problem of multiplying n such matrices $M^{(0)}, M^{(1)}, \ldots, M^{(n-1)}$ over Z. The theorem then follows from Proposition 5.2 of Cook [6]. The details of this reduction to the matrix product problem are given below.

CONSTRUCTION: For $0 \le i < n$ and $0 \le u, j < 2n$, compute

$$M^{(i)}_{j,u} = \begin{cases} 1 & \text{, if } j \equiv t_i \pmod 2 \text{ and } u - \frac{1}{2}(j - t_i) \in D_{i+1} \\ 0 & \text{, otherwise} \end{cases}$$

Call the oracle to compute $M = M^{(0)} \times \cdots \times M^{(n-2)} \times M^{(n-1)}$. Now $C(0, t_{n-1}t_{n-2}\cdots t_0)$ is computed as the 0^{th} entry of $V^{(\epsilon)} \times M$, where $V^{(\epsilon)}$ is the row vector $V^{(\epsilon)} = (C(0,\epsilon), \ldots, C(2n-1,\epsilon))$.

CORRECTNESS: Let y be a suffix of $t_{n-1}t_{n-2}\cdots t_0$, $i = |y|$, and $0 \le u < 2n$. By induction on i, it will be demonstrated that the u^{th} entry of $V^{(\epsilon)} \times M^{(0)} \times \cdots \times M^{(i-2)} \times M^{(i-1)}$ is $C(u,y)$.

Basis ($i = 0$): The u^{th} entry of $V^{(\epsilon)}$ is $C(u,\epsilon)$, by definition.

Induction: Let y be the suffix of $t_{n-1}t_{n-2}\cdots t_0$ whose length is i. Assume by the induction hypothesis that the j^{th} entry of $V^{(\epsilon)} \times M^{(0)} \times \cdots \times M^{(i-2)} \times M^{(i-1)}$ is $C(j,y)$, for all $0 \le j < 2n$. By the definition of $M^{(i)}$, the u^{th} entry of $(V^{(\epsilon)} \times M^{(0)} \times \cdots \times M^{(i-2)} \times M^{(i-1)}) \times M^{(i)}$ is thus

$$\sum_{\substack{0 \le j \le 2n-1 \\ j \equiv t_i \pmod 2 \\ u - \frac{1}{2}(j-t_i) \in D_{i+1}}} C(j,y).$$

Letting $k = u - \frac{1}{2}(j - t_i)$, this is

$$\begin{aligned} &\sum_{\substack{k \in D_{i+1} \\ u-n+1 \le k \le u}} C(t_i + 2(u-k), y) \\ &= \sum_{\substack{k \in D_{i+1} \\ k \le u}} C(t_i + 2(u-k), y) \text{ (by Lemma 17)} \\ &= C(u, t_i y) \qquad \text{(by Theorem 16).} \end{aligned}$$

ANALYSIS: The following operations are required in the construction, and are all readily seen to be in $\mathcal{NC}^1$: subtracting $O(\log n)$ bit integers, checking if an integer $k < 2n$ is in an input set $D \subseteq \{0,1,\ldots,n\}$, and computing the inner product of 2 vectors each of length $2n$ whose entries are integers of $O(n \log n)$ bits each. □

Corollary 19 Given input strings $e_{n-1}e_{n-2}\cdots e_0$, $t_{n-1}t_{n-2}\cdots t_0 \in \{0,1\}^n$ and a polynomial $a(x) \in Z_2[x]$, where $e = \text{val}(e_{n-1}e_{n-2}\cdots e_0)$, $t = \text{val}(t_{n-1}t_{n-2}\cdots t_0)$ and the degree of $a(x) = n$, the problem of computing the coefficient of x^t in $(a(x))^e$, computed over Z_2, is in $\mathcal{DET}$, and hence in $\mathcal{NC}^2$.

Proof: This follows directly from Corollary 15 and Theorem 18. □

6 The Generalization to Arbitrary Finite Fields

Using techniques similar to those of the preceding sections, we demonstrate that the general Polynomial Exponentiation problem is in $\mathcal{NC}^3$ when the characteristic of the field is polynomial in the input size and in $\mathcal{NC}^2$ when the characteristic is constant.

Let F be a finite field of characteristic p. Given $e, t \in \mathcal{N}$, let $e = \sum_{i=0}^{r-1} e_i p^i$ and $t = \sum_{i=0}^{m-1} t_i p^i$ where $0 \le e_i, t_i < p$. Then $e_{r-1}e_{r-2}\cdots e_0$ and $t_{m-1}t_{m-2}\cdots t_0$ are the base p representations of e and t, respectively. For convenience, we define $e_i = 0$ for $i \ge r$.

Consider the polynomial $a(x) = \sum_{j=0}^{d} a_j x^j \in F[x]$ of degree d.

Corollary 20 $[a(x)]^e = \prod_{i\ge 0} c_i(x^{p^i})$ where

$$c_i(x) = \left(\sum_{j=0}^{d} (a_j)^{p^i} x^j\right)^{e_i}.$$

Proof:

$$\begin{aligned}[a(x)]^e &= \prod_{i=0}^{r-1} (a(x))^{e_i p^i} \\ &= \prod_{i=0}^{r-1} \left(\sum_{j=0}^{d} (a_j)^{p^i} (x^{p^i})^j\right)^{e_i}, \text{ by Corollary 3} \\ &= \prod_{i=0}^{r-1} c_i(x^{p^i}).\end{aligned}$$

For $i \ge r$, we have $e_i = 0$ and, hence, $c_i(x^{p^i}) = 1$. Therefore $\prod_{i=0}^{r-1} c_i(x^{p^i}) = \prod_{i\ge 0} c_i(x^{p^i})$. □

Let $c_{i,j}$ be the coefficient of x^j in $c_i(x)$. In particular, if $j > de_i$ or $j < 0$, then $c_{i,j} = 0$.

If $y = y_{i-1}y_{i-2}\cdots y_0 \in \mathcal{N}^i$, define

$$\text{val}_p(y) = \sum_{j=0}^{i-1} y_j p^j.$$

For $u, i \in \mathcal{N}$ and $y \in \{0, 1, \ldots, p-1\}^i$, define $C_p(u, y)$ to be the coefficient of $x^{\text{val}_p(uy)}$ in $\prod_{j=0}^{i} c_j(x^{p^j})$.

Lemma 21 If $t < p^i$ then the coefficient of x^t in $\prod_{j\ge 0} c_j(x^{p^j})$ is equal to $\prod_{j\ge i} c_{j,0}$ times the coefficient of x^t in $\prod_{j=0}^{i-1} c_j(x^{p^j})$.

Proof: For $j \ge i$, every term of $c_j(x^{p^j})$, except its constant term, is a multiple of x^{p^i}. □

Corollary 22 The coefficient of x^t in $\prod_{j\ge 0} c_j(x^{p^j})$ is equal to $\left(\prod_{m<j<r} (a_0)^{e_j p^j}\right) \times C_p(0, t_{m-1}t_{m-2}\cdots t_0)$.

Proof: Since $t < p^{m+1}$, it follows from Lemma 21 that $\prod_{j\ge 0} c_j(x^{p^j})$ is equal to $\left(\prod_{j\ge m+1} c_{j,0}\right) \times C_p(0, t_{m-1}t_{m-2}\cdots t_0)$. For $j \ge r$, we have $e_j = 0$ and $c_{j,0} = 1$. Therefore $\prod_{j\ge m+1} c_{j,0} = \prod_{m<j<r} (a_0)^{e_j p^j}$. □

Another way to view $C_p(0, t_{m-1}t_{m-2}\cdots t_0)$ is as a weighted count of the redundant representations of the integer $t \ge 0$ in a radix p system. More precisely, for each $j \ge 0$, we define the weight function $w_j : \mathcal{N} \to F$ by $w_j(k) = c_{j,k}$. (Recall that $c_{j,k}$ is the coefficient of x^k in $c_j(x)$.) The weight of a string $y = y_i \cdots y_0 \in \mathcal{N}^{i+1}$, is defined to be

$$w(y_i \cdots y_0) = \prod_{j=0}^{i} w_j(y_j).$$

Then, for $u, i \in \mathcal{N}$ and $y \in \{0, 1, \ldots, p-1\}^i$, $C_p(u, y) = \sum\{w(z) \mid z \in \mathcal{N}^{i+1} \text{ and } \text{val}(z) = \text{val}(uy)\}$.

Note that if $z = z_i \cdots z_0$ and $c_{j,z_j} = 0$ for some $0 \le j \le i$, then $w(z) = 0$ and, hence, the representation z does not contribute to the weighted count. For the field Z_2,

$$w(z) = \begin{cases} 1 & \text{if } z_i \in D_i \text{ for } 0 \le i \le k \\ 0 & \text{otherwise} \end{cases}$$

and thus $C_2(u, y)$ is the parity of the function $C(u, y)$ used in Section 4.

Example 6 Consider $a(x) = (x^0 + 2x^1 + 2x^2 + x^4) \in Z_3[x]$ and $e = 46$, which has the base 3 representation 1201. Then

$$\begin{aligned}[a(x)]^e &= (x^0 + 2x^1 + 2x^2 + x^4)^{46} \\ &= (x^0 + 2x^1 + 2x^2 + x^4)^{1\cdot 27} \times \\ &\quad (x^0 + 2x^1 + 2x^2 + x^4)^{2\cdot 9} \times \\ &\quad (x^0 + 2x^1 + 2x^2 + x^4)^{0\cdot 3} \times \\ &\quad (x^0 + 2x^1 + 2x^2 + x^4)^{1\cdot 1} \\ &= (x^0 + 2x^1 + 2x^2 + x^4)^{27} \times \\ &\quad (x^0 + x^1 + 2x^2 + 2x^3 + x^5 + x^6 + x^8)^9 \times \\ &\quad (x^0) \times \\ &\quad (x^0 + 2x^1 + 2x^2 + x^4).\end{aligned}$$

In this redundant representation system, the integer $t = 82$, which is 10001 expressed in ternary, can be represented in exactly four ways: 10001, 3001, 2301, and 1601. The weights associated with these representations are 0, 0, $8 \equiv 2 \bmod 3$, and $4 \equiv 1 \bmod 3$, respectively. Thus $C_3(4, 10001) \equiv 0 \bmod 3$.

Notice that $C_p(u, \epsilon) = c_{0,u}$ for all $u \in \mathcal{N}$. More generally, $C_p(u, y)$ can be computed by means of the following recurrence.

Theorem 23 For $u, i \in \mathcal{N}$, $b \in \{0, 1, \ldots, p-1\}$, and $y \in \{0, 1, \ldots, p-1\}^i$,

$$C_p(u, by) = \sum_{0 \le k < dp} c_{i+1,k} C_p(b + p(u-k), y).$$

Proof: Let $z = \mathrm{val}_p(uby)$. The value of $C_p(u, by)$ is the coefficient of x^z in $\prod_{j=0}^{i+1} c_j(x^{p^j}) = c_{i+1}(x^{p^{i+1}}) \left(\prod_{j=0}^{i} c_j(x^{p^j}) \right)$. This is the sum of the coefficients of x^z in the polynomials $c_{i+1,k} x^{kp^{i+1}} \left(\prod_{j=0}^{i} c_j(x^{p^j}) \right)$. Since the coefficient of x^z in $c_{i+1,k} x^{kp^{i+1}} \left(\prod_{j=0}^{i} c_j(x^{p^j}) \right)$ is $c_{i+1,k}$ times the coefficient of $x^{z-kp^{i+1}}$ in $\prod_{j=0}^{i} c_j(x^{p^j})$ and $z - kp^{i+1} = \mathrm{val}((b + p(u-k))y)$, it follows that $C_p(u, by) = \sum_k c_{i+1,k} C_p(b + p(u-k), y)$.

Now $c_{i+1}(x)$ is a polynomial with degree less than dp, so $c_{i+1,k} = 0$ for $k \ge dp$. We may therefore assume that the limits of the summation are $0 \le k < dp$. □

Theorem 24 For any $n > 0$, given a field F of characteristic p and order $q = p^\ell$, a polynomial $a(x) \in F[x]$, of degree d, and $e, t \in \mathcal{N}$, such that p, ℓ, d, $\log(e)$, and $\log(t)$ are polynomial in n, the problem of computing the coefficient of x^t in $(a(x))^e$ can be solved by a Boolean circuit of depth $O((\log n)^2 \log\log n))$ and polynomial size.

Proof: Let $t_{m-1} t_{m-2} \cdots t_0$ denote the base p representation of t. Consider the row vector

$$V^{(y)} = (C_p(0, y), ..., C_p(dp-1, y)).$$

From the recurrence of Theorem 23, it follows that $V^{(by)}$ can be computed by simply multiplying $V^{(y)}$ by a $dp \times dp$ matrix with entries in F.

Specifically, let $M^{(i)}$ be the $dp \times dp$ matrix such that $M^{(i)}_{k,u} = c_{i+1,i p(u-k)}$. Then $V^{(t_{i-1} t_{i-2} \cdots t_0)} = V^{(t_{i-1} \cdots t_0)} \times M^{(i)}$ so, by induction, $V^{(t_{m-1} t_{m-2} \cdots t_0)} = V^{(\epsilon)} \times M^{(0)} \times \cdots \times M^{(m-1)}$. Notice that $C_p(0, t)$ is the 0^{th} element of $V^{(t_{m-1} t_{m-2} \cdots t_0)}$.

The computation proceeds as follows.

- Convert the binary integer e to base p. As in Theorem 7, since $\log e$ and $\log t$ are polynomial in n, this can be done in $O(\log n \log\log n)$ steps.
- Compute the coefficients of the polynomials $c_i(x)$, for $0 \le i \le m-1$. This can be done by raising each coefficient of $a(x)$ to the power p^i, as in Theorem 7, via a Boolean circuit of depth $O((\log n)^2 \log\log n)$ and then raising the resulting degree d polynomial to the power $e_i < p$, which, from Theorem 4, can be done in $O(\log n \log\log n)$ depth, since d and p are polynomial in n.
- Compute the product $V^{(\epsilon)} \times M^{(0)} \times \cdots \times M^{(m-1)}$. (Recall that the elements of $V^{(\epsilon)}$ and $M^{(0)}, \ldots, M^{(m-1)}$ are coefficients of the polynomials $c_0(x), \ldots, c_{m-1}(x)$.) Since $m = O(\log t)$, d, and p are polynomial in n, these matrix multiplications can be performed in parallel, as in Theorem 7, by a Boolean circuit of depth $O((\log n)^2 \log\log n)$.
- Multiply $C_p(0, t_{m-1} t_{m-2} \cdots t_0)$ (which is the 0^{th} entry of the row vector $V^{(\epsilon)} \times M^{(0)} \times \cdots \times M^{(m-1)}$) by $\prod_{m+1 \le i < r} (a_0)^{c_i p^i}$. As above, this can be computed by a Boolean circuit in depth $O((\log n)^2 \log\log n)$. □

The following two corollaries are analogues of Corollaries 8 and 9 in Section 3.

Corollary 25 The *Polynomial Exponentiation Problem* is in $\mathcal{NC}^3$ for finite fields of polynomial characteristic.

Corollary 26 The *Polynomial Exponentiation Problem* is in $\mathcal{NC}^2$ for finite fields whose order is $O(1)$.

7 #P-Completeness of a Variant

As mentioned in Section 1, if the degree of the base polynomial $a(x)$ is changed from n to exponential in n (but $a(x)$ still is allowed to have only n nonzero terms), the effect is to change the complexity of the problem: instead of being in $\mathcal{NC}^3$, it is now $\#\mathcal{P}$-hard. This remains true even if the exponent is changed from exponential in n to n. (See Garey and Johnson [8] for a discussion of $\#\mathcal{P}$.)

Theorem 27 (Rabin, Valiant [personal communication]) Let

$$a(x) = x^0 + x^{w_1} + x^{w_2} + \cdots + x^{w_n},$$

where $w_1, w_2, \ldots, w_n$ are n-bit integers. Given $a(x)$ and an n-bit integer t, the problem of computing the coefficient of x^t in $(a(x))^n$ over Z is $\#\mathcal{P}$-complete.

Proof: To demonstrate inclusion in $\#\mathcal{P}$, a counting machine would make n independent nondeterministic guesses, each guess being one of $a(x)$'s $n+1$ terms. The machine accepts if and only if the product of these n guessed terms is x^t. The number of accepting computations is the coefficient of x^t in $(a(x))^n$.

To demonstrate $\#\mathcal{P}$-hardness, notice that the problem is a slight variant of the counting version of a knapsack problem with weights $w_1, w_2, \ldots, w_n$ and knapsack capacity t; the only difference is that the same weight can be selected more than once, as long as the total number of selections is still at most n. By returning to the reduction from the exact cover problem to the knapsack problem [8], it is straightforward to verify that, for those instances of the knapsack problem that arise in the reduction, no solution exists in which a weight is selected more than once. □

It is even easier to see that computing the coefficient of x^t in the product of n sparse polynomials is $\#\mathcal{P}$-complete, by considering $(1 + x^{w_1})(1 + x^{w_2}) \cdots (1 + x^{w_n})$. A product of sparse polynomials arises as $\prod_{i=0}^{r-1} a(x^{2^i})$ in Section 4; what seems to make these easier to compute is the special relationship (a geometric sequence) of the exponents among the constituent polynomials.

Notice that Theorem 27 remains true when Z is replaced by Z_p, where $p > (n+1)^{n-1}$, since the number of solutions of this version of the knapsack problem is at most $(n+1)^{n-1}$. If, instead of having the base field fixed in advance, we allow the field to be specified as part of the input, then, by the Chinese Remainder Theorem, this sparse polynomial exponentiation problem is still $\#\mathcal{P}$-complete with the restriction that characteristic of the input field is $O(n \log n)$. This problem is even closer to that discussed in Section 6.

Acknowledgements

We thank Les Goldschlager for discussions that led to the problem of exponentiation in $Z_2[x]$, and Michael Rabin and Les Valiant for observations on the $\#\mathcal{P}$-completeness of the sparse polynomial problems discussed in Section 7. We are particularly indebted to Don Coppersmith who pointed out Berlekamp's Q matrix method, transforming what had previously been a nonuniform algorithm for Modular Polynomial Exponentiation into a uniform one.

This paper is based upon work supported by the National Science Foundation under Grants DCR-8301212, DCR-8352093, and MCS-8402676. The first author was also supported by an IBM Faculty Development Award and the University of Washington Graduate School Research Fund.

References

[1] H. Alt, "Comparison of Arithmetic Functions with respect to Boolean Circuit Depth", *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, Washington, D.C., 1984, 466-470.

[2] H. Alt and N. Blum "On the Boolean Circuit Depth of Division Related Functions", Department of Computer Science, Pennsylvania State University, 1983.

[3] D. Angluin, "Lecture Notes on the Complexity of Some Problems in Number Theory", Technical Report #243, Yale University, August 1982.

[4] P. W. Beame, S. A. Cook, and H. J. Hoover, "Log Depth Circuits for Division and Related Problems", *Proceedings of the Twenty Fifth Annual IEEE Symposium on Foundations of Computer Science*, Singer Island, Florida, 1984, 1-6.

[5] E. R. Berlekamp, "Factoring Polynomials over Large Finite Fields", *Math. Comput.* **24**, 1970, 713-735.

[6] S. A. Cook, "A Taxonomy of Problems with Fast Parallel Algorithms", University of Toronto, June 1984.

[7] W. Eberly, "Very Past Parallel Matrix and Polynomial Arithmetic" *Proceedings of the Twenty Fifth Annual IEEE Symposium on Foundations of Computer Science*, Singer Island, Florida, 1984, 21-30.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[9] J. von zur Gathen, "Parallel Algorithms for Algebraic Problems", *SIAM Journal on Computing* **13**, November 1984, 802-824.

[10] J. von zur Gathen, 'Parallel Powering", *Proceedings of the Twenty Fifth Annual IEEE Symposium on Foundations of Computer Science*, Singer Island, Florida, 1984, 31-36.

[11] J. von zur Gathen, "Computing Powers in Parallel", preprint, December 1984.

[12] R. M. Karp, "Reducibility among Combinatorial Problems", *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, 85-103.

[13] D. J. Kuck, *The Structure of Computers and Computation*, volume 1, Wiley, New York, 1978.

[14] G. Miller, "Riemann's Hypothesis and Tests for Primality", *Journal of Computer and System Sciences* **13**, 1976, 300-317.

[15] M. O. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", Technical Report MIT/LCS/TR-212, M.I.T., January 1979.

[16] M. O. Rabin, "Probabilistic Algorithm for Testing Primality", *Journal of Number Theory* **12**, 1980, 128-138.

[17] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Crypto-Systems", *Communications of the ACM* **21**, 1978, 120-126.