ISSN 0105-8517

The Computational Efficacy of Finite Field Arithmetic

Gudmund Frandsen
Carl Sturtivant

DAIMI PB – 227
August 1987

AARHUS UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
Ny Munkegade 116 – DK 8000 Aarhus C – DENMARK
Telephone: + 45 6 12 83 55 *Telex: 64767 aausci dk*

PB – 227 Frandsen & Sturtivant: The Computational Efficacy …

The Computational Efficacy of Finite Field Arithmetic

Carl Sturtivant & Gudmund Frandsen
Computer Science Department
Aarhus University, Aarhus, Denmark

Abstract

We show that there exists an interesting non-uniform model of computational complexity within characteristic two finite fields. This model regards all problems as families of functions whose domain and co-domain are characteristic two fields. The model is both a structured and a fully general model of computation.

We ask if the same is true when the characteristics of the fields are unbounded. We show that this is equivalent to asking whether arithmetic complexity over the prime fields is a fully general measure of complexity.

We show that this reduces to whether or not a single canonical function is "easy" to compute using only modulo p arithmetic.

We show that the arithmetic complexity of the above function is divided between two other canonical functions, the first to be computed modulo p and the second with modulo p^2 arithmetic.

We thus have tied the efficacy of finite field arithmetic to specific questions about the arithmetic complexities of some fundamental functions.

Introduction

Let p be a prime number. Then the integers in the range 0 to $p-1$ with modulo p arithmetic operations constitute a field. It is a remarkable fact that there exists an element of this field g with the property that the powers g^i for i between 1 and $p-1$ run through all of the non-zero elements of the field. All finite fields contain such a g, which is known as a primitive element [Lidl 83].

Now consider the function $exp_g(x) = g^x$, which is an exponential function within the integers modulo p. Ignoring zero, since g is a primitive element, this function has an inverse called the discrete logarithm $dlog_g$.

The function exp_g is "easy" to compute by "repeated squaring" [Dixon 84, Knuth 81] modulo p, whereas the function $dlog_g$ appears to be "hard" [Odlyzko 84], and therefore may be a one-way function. Many cryptographic schemes are based upon this [e.g. Blum 84, El Gamal 85, Goldreich 86, Odlyzko 84, Yao 82]. The computational complexity of $dlog_g$ is being heavily investigated [Adleman 79, Coppersmith 84]. This and all other useful candidates for one-way functions and other cryptographic objects seem to arise in an algebraic or number-theoretic context [Yao 82].

We pose the question: do all efficient (i.e. polynomial time) algorithms have a purely arithmetic equivalent? If so, then in investigating $dlog_g$ within a finite field one need only consider arithmetic computations. Note that in investigating exp_g, it is not at all clear that there is an efficient ("easy") purely arithmetic computation.

To answer this question, we will consider first the restricted case of fields with bounded characteristic. We show that for such fields all polynomial time computations have a purely arithmetic equivalent.

Consider the following non-uniform model of computation: all finite fields with their arithmetic operations only, where increasing the input size is accommodated by increasing the field size. Thus a computational problem is a family of functions $\{f_i\}$ where for each i, the domain and co-domain of f_i is a finite field. (Contrast this with the boolean model in [Skyum 85]).

The finite field model is unique. All other models of computation provide that an increase in the input size is accommodated by an increase in the number of variables (e.g. the number of bits or the number of symbols in some alphabet). In the finite field model there is always only one variable. In fact, there is exactly one input and one output.

Obviously, since we have only one variable we need a measure of the input size. The input size of f_i is naturally the base two logarithm of the size of the domain of f_i. This is precisely the minimum number of bits required to distinguish all of the field elements.

Now consider any reasonable compact and efficient representation of all finite fields by the booleans. By this we mean assignment of "reasonably short" bit-strings uniquely to every finite field element, in such a way that "small" boolean circuits exist to implement the finite field arithmetic. This definition is made precise in the next section.

Such a representation can provide a means of interpreting an arbitrary boolean computational problem as a finite field computational problem. This can be achieved by interpreting the bit-string input and output as the field element it represents, provided that all bit-strings are used to represent field elements (some technicalities are being deferred here).

The issue then arises as to whether the existence of an efficient boolean solution to the computational problem implies the existence of an efficient arithmetic solution. Here by "efficient solution" we mean the existence of polynomial size families of circuits to solve the problem.

Following [Borodin 82] we define two classes of non-uniform models of computation. First, we define a general model of computation as one that gives a complexity measure that is polynomially related to boolean circuit-size. Second, we define a structured model of computation as one in which there is no fixed bound on the size of the values manipulated in one atomic operation. (Here by size we mean the number of bits in a reasonable bit representation). These definitions differ from those in [Borodin 82] in order to make the defined concepts independent.

Let us apply these terms to some well-known models of computation. Straight line programs over the complex numbers [Strassen 73, Valiant 82] is a structured but not general model of computation. Boolean circuits is a not structured but general model. Monotone boolean circuits is possibly neither structured nor general. Clearly our finite field model of computation is a structured model. The above question of the efficacy of arithmetic is equivalent to asking whether or not it is a general model of computation.

When restricted to characteristic two, the finite field model is both a structured and a general model of computation. The status of the finite field model for unbounded characteristic is unresolved. However, we will show that it is a general model of computation if and only if a certain canonical function has polynomial size arithmetic circuits.

The existence of a simultaneously structured and general model of computation is of interest because it shows that a macroscopic structure can be imposed upon boolean computations without loss of efficiency. In particular, only bit-strings representing characteristic two field elements need be manipulated in a boolean computation, and the only manipulations required may be implemented by standard boolean circuits that simulate the addition and multiplication of the field. All boolean computations can be done efficiently by "macrogates" whose interactions with each other are well defined and elegant. These macrogates represent arithmetic in a field and thus are subject to the commutative, associative and distributive laws.

Using macrogates is an extremely restrictive structural imposition: first, only data of size equal to the input size can be manipulated; second, there can be only a small set of standard circuits used; and third, there can be no loss of efficiency. If this can be achieved, macrogates open up the large toolbox already available to the study of finite fields and make it applicable to the complexity theorist in search of boolean lower bounds. Concepts such as primes, factoring, irreducibility, orders of elements, that were essentially invisible in the original model are now available. Obviously, this is a massive conceptual gain.

We will show that the finite field model of computation introduced above is an inherently sequential model; in the sense that circuit depth is <u>not</u> preserved in going over from boolean computations to computations within our model, and indeed may get exponentially bigger. Therefore from the point of view of efficient sequential computation, large numbers of irrelevant computations are automatically excluded. This is hopeful from a "sequential lower bound" point of view, where we want to <u>exclude</u> the possibility of the existence of certain computations. If there are fewer to begin with, this can only help.

Consider a boolean computation with inputs $x_1, \ldots, x_n$. We may write the function computed in the SOPE normal form [Savage 76] as follows:

$$\sum_{A \in S} \left(\prod_{i \in A} x_i \right)$$

where S is a subset of the power set of $\{1, \ldots, n\}$, where sum and product are *exclusive-or* and *and* respectively, and where the empty product is taken to be 1. This is clearly a normal form because $\{0,1\}$ with sum and product as above is a field where all elements satisfy $x^2 = x$. The above expression can give all functionally distinct polynomials in n variables. Without loss of generality we may consider the boolean computation to involve only *exclusive-or* and *and* operations because adding 1 is permitted. We may now track the computation formally, by regarding the inputs $x_1, \ldots, x_n$ as indeterminates and defining the outcome of an operation as a polynomial equal to the formal sum or product of the incoming polynomials as appropriate. The result of the computation is then some polynomial with zero-one coefficients that we denote by $p(x_1, \ldots, x_n)$. Its relation to the normal form is as follows:

$$p(x_1, \ldots, x_n) = \sum_{A \in S} (\prod_{i \in A} x_i) + \sum_{i=1}^{n} \phi_i(x_1, \ldots, x_n)(x_i^2 - x_i)$$

where the ϕ_i are arbitrary polynomials with zero-one coefficients.

Compare this redundancy in the computation with the redundancy in the finite field model set up above. Suppose we are computing over a field with q elements. There is only one input x. Clearly, any function can be written by interpolation as a polynomial of degree $q-1$ at most. In fact, every field element satisfies $x^q = x$ (see [Lidl 83]). Thus the natural normal form is:

$$\sum_{i=0}^{q-1} a_i x^i$$

where the a_i are field elements. If $p(x)$ is the formal polynomial resulting from tracking the computation formally, then it is related to the normal form by:

$$p(x) = \sum_{i=0}^{q-1} a_i x^i + \phi(x)(x^q - x)$$

where $\phi(x)$ is an arbitrary polynomial with coefficients from the field. This is the same as:

$$p(x) \equiv \sum_{i=0}^{q-1} a_i x^i \; mod \; (x^q - x)$$

The question as to whether or not there is a uniform relationship between the boolean and finite field models of computation depends upon the uniform complexity of fundamental algebraic problems such as primality and the irreducibility of polynomials over finite fields. Such problems can be solved in random polynomial time, the randomization being some sort of "poor man's non-uniformity" [Adleman 78, Bennett 81, Gill 77]. (For a discussion of uniformity versus non-uniformity see [Borodin 77, Karp 80, Karp 82, Pippenger 79, Ruzzo 81, Valiant 83].)

If there is a uniform relationship, then uniform complexity theory can be carried into the model *en masse*, with the benefit that the subtle algorithms involved in establishing the existence of the uniform relationship implicitly add conceptual force to this new view of computation.

Conversely, if there is no uniform relationship, then attempts to prove negative results in the finite field model start from an inherently non-uniform angle. This is because any lower bound result is nevertheless implicitly parametrized by the input size. Adding one more bit to the input does not radically change the nature of the computational process. However, increasing the input size by one by increasing the field size, and thus using new, and uniformity-wise unrelated arithmetic operations, makes a radical change by including some intrinsic non-uniformity to an arithmetic lower bound.

The existence or non-existence of polynomial-time algorithms for problems such as factoring integers, factoring polynomials over finite fields, and related number-theoretic complexity investigations, clearly has great impact on the nature and character of computations in the proposed model, and therefore must now be considered of direct relevance to general boolean computations, a fact that had heretofore not been established with such force. See [Berlekamp 68, Dixon 84, Rabin 80, Riesel 86] for results in these areas.

In the usual combinatorial scenario, most natural problems in NP seem to be either in P or to be NP-complete [Garey 79]. This phenomenon is probably merely a reflection of the way problems are usually posed (i.e. in a relatively unstructured setting). Over a finite field with q elements, a problem is naturally posed as $\sum_{i=1}^{q-1} a_i x^i$ where the a_i are given in a natural way as a function of i. For example, the discrete logarithm can be naturally posed as:

$$dlog_g(x) = \sum_{i=1}^{p-2} \frac{x^i}{1-g^i}$$

in the field of the integers modulo a prime p (this can be easily obtained by interpolation, as described in [Lidl 83]).

It seems likely that within the new model (because of the irregular relationship with the booleans), the distribution of natural problems across the complexity classes may be quite different from usual. This supposition is supported by the known difficulty of classifying many natural algebraic and number-theoretic problems in a combinatorial setting (e.g. primality testing, factoring integers, and group isomorphism not fitting in neatly with P versus NP [Garey 79]). Thus the finite field model may prove to be useful in investigating the structure of NP, and in particular for cryptographically motivated complexity.

In the next sections we will show that a finite field model of computation is fully general if and only if any reasonable representation of the field elements as bit-strings is efficiently accessible within the field. That is to say iff there exist small arithmetic circuits with one input and many outputs which, when a field element is input, compute the bits of its representation as zero-one field values. We define a representation as strong iff it is accessible within the fields in the above sense. We define one representation to be equivalent to another iff there exists a family of small boolean circuits that translate from one to the other and *vice-versa*. Note that boolean circuits can easily be efficiently simulated by field arithmetic on zero-one values.

Next we show that if a strong representation exists then all reasonable representations are strong and equivalent to each other. Thus the issue becomes that of whether or not the usual ("standard") representation of the finite fields is strong. We show that for bounded characteristic this is indeed the case. Thus in the case of characteristic two, arithmetic is adequate for all polynomial time computations. In the case of unbounded characteristic we show that the problem reduces to that for the *prime* fields (i.e. for modulo p arithmetic).

We define the function $f(x)$ by:

$$f(x) = (\frac{x - x^p}{p}) \, mod \, p$$

where the arithmetic within the parentheses is *integer* arithmetic. Since $x - x^p \equiv 0 \, mod \, p$, the division by p is well defined over the integers, and f defines a function whose domain and co-domain are both the integers modulo p. We then show that the standard representation of the fields of the integers modulo p is strong if and only if the function f is easy to compute with modulo p arithmetic. The function f is not the only one with this property, but is canonical in its form and expressive power. We find the degree $p - 1$ polynomial representing f: its coefficients involve

Bernoulli numbers modulo p, thus suggesting a fundamental connection with other problems in number theory [Borevich 66].

We define the function $g(x)$ by:

$$g(x) = (\frac{1 + x^p - (1 + x)^p}{p})\ mod\ p$$

where, just as with f, the arithmetic within the parenthesis is integer arithmetic. This function has a simple expansion as a polynomial of degree $p - 1$ within the field:

$$g(x) = \sum_{i=1}^{p-1} \frac{(-1)^i x^i}{i}$$

(The division here is within the field of integers modulo p). Note the similarity to the power series for the logarithm characteristic zero. For this reason we refer to g as the pseudo-logarithm. Note also that both f and g can be computed efficiently outside the field. (This can be achieved by using modulo p^2 arithmetic and repeated squaring).

We define the function $m(x)$ over the integers modulo p^2 by $m(x) = x mod p$ with the intuitively obvious meaning, and we define the arithmetic complexity of m to be its arithmetic complexity in the ring of integers modulo p^2. We show that m is equal to the Bernoulli polynomial B_p taken modulo p^2, apart from a few small terms of trivial arithmetic complexity.

Finally, we show that the arithmetic complexity of f lies between that of m and the product of the arithmetic complexity of m with the arithmetic complexity of g. Therefore, the arithmetic complexity of f can be determined by finding the arithmetic complexities of m and g. Since neither m nor g seem as powerful as f, this must be regarded as a true simplification.

Arbitrary vs. Arithmetic Computations

The following are well known facts about finite fields (see [Lidl 83] for details).

- All finite fields with identical cardinality are isomorphic.
- There exists a finite field of cardinality n, precisely when n is a prime-power.

On this basis the finite field of cardinality $q = p^n$ for some prime p and natural number $n \geq 1$ is denoted $\mathbf{F}_q$, in particular $\mathbf{F}_p$ when $q = p$. In the latter case, $\mathbf{F}_p$ is said to be a prime field.

- All elements of $\mathbf{F}_q$ satisfy $x^q = x$.

Thus division is unnecessary as $x^{-1} = x^{q-2}$ when x is non-zero. The latter can be computed in $O(log\, q)$ multiplications by repeated squaring. In fact, it is possible to eliminate division with only a constant loss of efficiency for functions with at least linear arithmetic complexity [Bøgestrand 87]. Subtraction can also be eliminated with only constant loss of efficiency by observing that $a + b$ is equal to $a + (-1)b$.

We may now define the finite field model of computation:

i) A finite field computational problem Δ is a family of functions, one for every finite field: $\Delta = \{\delta_q \mid q \text{ is a prime power}\}$ such that $\delta_q : \mathbf{F}_q \rightarrow \mathbf{F}_q$.

ii) A family of arithmetic circuits $\{a_q \mid q \text{ is a prime power}\}$ contains for each finite field $\mathbf{F}_q$ a circuit a_q that uses $\mathbf{F}_q$-arithmetic $(+, \cdot)$ and $\mathbf{F}_q$-constants.

iii) An arithmetic solution to Δ consists of a family $\{a_q\}$ of arithmetic circuits such that a_q computes a unary function and a_q and δ_q are functionally identical for all q.

iv) The input size for an $\mathbf{F}_q$-problem, or for a circuit indexed by q, is taken to be $\log q$ since the cardinality of $\mathbf{F}_q$ is q.

v) The complexity of a family of circuits is taken to be the circuit size as a function of input size. A family of circuits is said to be p-bounded if the complexity is bounded by some polynomial in the input size.

Observe that solutions are nonuniform because each circuit only works for a specific field (*i.e.*, input size). If uniformity is desired, it can be imposed by requiring structural similarity between the individual circuits that constitute a solution, *e.g.*, via constructing all of the infinite family of circuits by a single Turing Machine of specified complexity.

It should be noted that every finite field problem has an arithmetic solution. This follows from the fact that any function in $\mathbf{F}_q$ can be expressed as a polynomial over $\mathbf{F}_q$.

Having defined the arithmetic model of computation, we shall see that we may restrict ourselves to deal with prime fields in the sense that arithmetic in any finite field can be simulated efficiently by arithmetic in its underlying prime subfield:

Theorem 1-a

A finite field $\mathbf{F}_q (q = p^n)$ may be represented in terms of $\mathbf{F}_p$ because there exist two p-bounded families of circuits $\{\alpha_q, \mu_q : \mathbf{F}_p^n \times \mathbf{F}_p^n \rightarrow \mathbf{F}_p^n\}$ all using $\mathbf{F}_p$-arithmetic and a semantic bijection $\phi_q : \mathbf{F}_p^n \rightarrow \mathbf{F}_q$ such that for all q:

$$\begin{aligned} \phi_q(\alpha_q(\bar{x}, \bar{y})) &= \phi_q(\bar{x}) + \phi_q(\bar{y}) \\ \phi_q(\mu_q(\bar{x}, \bar{y})) &= \phi_q(\bar{x}) \cdot \phi_q(\bar{y}) \end{aligned}$$

Proof of Theorem 1-a:

Given $q = p^n$, there exists an irreducible polynomial $f(x) \in \mathbf{F}_p[x]$ of degree n. It is the case that $\mathbf{F}_q \cong \mathbf{F}_p[x]/f(x)$ [Lidl 83]. Thus $\mathbf{F}_q$ may be regarded as a vector space over $\mathbf{F}_p$. Since $(1, x, x^2, \cdots, x^{n-1})$ is a basis for this vector space, we may define $\phi_q(a_0, a_1, \cdots, a_{n-1}) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$. Obviously, addition may be performed componentwise in this representation and the p-bounded family of circuits $\{\alpha_q\}$ is constructable. Multiplication may be performed as a convolution followed by a modulo-f operation. Hence the p-bounded family $\{\mu_q\}$ can be constructed.

$\square$

Theorem 1-b

Given the representation of $\mathbf{F}_q$ in terms of $\mathbf{F}_p$ as defined above (called the standard representation), there exist two p-bounded families of arithmetic circuits $\{i_q : \mathbf{F}_q^n \to \mathbf{F}_q\}$, $\{o_q : \mathbf{F}_q \to \mathbf{F}_q^n\}$ such that $i_q(\bar{x}) = \phi_q(\bar{x})$ and $\phi_q(o_q(y)) = y$ for $\bar{x} \in \mathbf{F}_p^n$ and $y \in \mathbf{F}_q$.

Proof of Theorem 1-b:

The family of circuits $\{i_q\}$ is easily constructed as a p-bounded family, since $\phi_q(a_0, a_1, \cdots, a_{n-1}) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$. In the case of $\{o_q\}$, we note that $(\sum_{i=0}^{n-1} a_i \cdot x_i)^p = \sum_{i=0}^{n-1} a_i(x_i^p)$ for $a_i \in \mathbf{F}_p$, $x_i \in \mathbf{F}_q$, since $\mathbf{F}_q$ has characteristic p and $a = a^p$ for $a \in \mathbf{Z}_p$. This leads to a system of linear equations for finding the coordinates $(a_0, \ldots, a_n)$ of an element α in $\mathbf{F}_q$:

$$\begin{bmatrix} 1 & x & x^2 & x^3 & \ldots & x^{n-1} \\ 1 & x^p & x^{2p} & x^{3p} & \ldots & x^{(n-1)p} \\ 1 & x^{p^2} & x^{2p^2} & x^{3p^2} & \ldots & x^{(n-1)p^2} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x^{p^{n-1}} & x^{2p^{n-1}} & x^{3p^{n-1}} & \ldots & x^{(n-1)p^{n-1}} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha^p \\ \alpha^{p^2} \\ \vdots \\ \alpha^{p^{n-1}} \end{bmatrix}$$

Since $(1, x, \ldots, x^{n-1})$ is a basis for $\mathbf{F}_q$ over $\mathbf{F}_p$, the involved matrix is invertible [Lidl 83]. Consequently we find:

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \bar{A} \begin{bmatrix} \alpha \\ \alpha^p \\ \vdots \\ \alpha^{p^{n-1}} \end{bmatrix}$$

for some constant matrix $\bar{A}$ over $\mathbf{F}_q$). In addition, each of the powers $\alpha^p, \alpha^{p^2}, \ldots, \alpha^{p^{n-1}}$ may be computed using at most $O(\log q)$ multiplication by repeated squaring. Hence $\{o_q\}$ may be constructed as a p-bounded family of circuits.

In fact as is shown in [Bøgestrand 87], $\{o_q\}$ may be constructed as a family of circuits of size only $O(n\ log^2 n\ log\ log\ n)$ in the case of fields of characteristic two, using results from [Aho 74] and [Schönhage 77]. This means that in this representation, arithmetic is as efficient as boolean operations for any problem with complexity worse than $O(n\, log^2 n\, log log\, n)$.

□

Since $\mathbf{F}_p$ is a subfield of $\mathbf{F}_q$ [Lidl 83], theorem 1-a establishes a representation of $\mathbf{F}_q$ in terms of itself. In addition, 1-b asserts that it is possible to translate back and forth efficiently between an element in $\mathbf{F}_q$ and its representation in $\mathbf{F}_q^n$ (*i.e.*, $\mathbf{F}_p^n$) by $\mathbf{F}_q$-arithmetic.

In order to discuss the relation between arithmetic and general computations, we define the latter in terms of Boolean algebra:

i) Let the Booleans be given by $\mathbf{B} = \{0, 1\}$ with the three operators $\wedge, \vee, \neg$ where they have the usual interpretations.

ii) A general computational problem Γ is a family of functions: (one for every natural number n) $\Gamma = \{\gamma_n\}$ such that $\gamma_n : \mathbf{B}^n \to \mathbf{B}^n$.

iii) A family of Boolean circuits $\{b_n\}$ contains for every natural number n a circuit b_n that uses boolean operations $(\wedge, \vee, \neg)$ and constants $(0, 1)$.

iv) A general solution to a problem $\Delta = \{\gamma_n\}$ consists of a family of boolean circuits $\{b_n\}$ such that b_n computes a unary function on $\mathbf{B}^n$ and b_n and γ_n are functionally identical. Thus we allow non-uniform solutions.

v) The input size for a $\mathbf{B}^n$-problem or for a circuit indexed by n is taken to be n. Thus the previous definitions of complexity and p-boundedness apply also to Boolean circuits.

Since $0, 1$ are elements of any field, we may identify $\mathbf{B} = \{0, 1\}$ with the corresponding subset of any finite field.

Theorem 2

Given a general problem $\Gamma = \{\gamma_n\}$, there exists an arithmetic problem $\Delta = \{\delta_q\}$, an assignment l from natural numbers to finite field indices (prime powers), and two p-bounded families of circuits $\{i_n : \mathbf{B}^n(\subseteq \mathbf{F}_{l(n)}^n) \to \mathbf{F}_{l(n)}\}, \{o_n : \mathbf{F}_{l(n)} \to \mathbf{F}_{l(n)}^n(\subseteq \mathbf{B}^n)\}$ using $\mathbf{F}_{l(n)}$-*arithmetic* such that $\gamma_n = o_n \circ \delta_{l(n)} \circ i_n$ for all n.

Proof of Theorem 2:

Choose the assignment $l(n) = 2^n$ and the corresponding subsets of the circuit families $\{i_q\}, \{o_q\}$ from Theorem 1-b as $\{i_n\}, \{o_n\}$. Then Δ can be chosen as a trivial extension of $\{i_n \circ \gamma_n \circ o_n\}$ to a finite field problem.

$\square$

We now consider representations of the finite fields using the booleans. Since efficiency is an issue, we choose those representations that are reasonably compact, and have small boolean circuits to implement the arithmetic operations. We define a *p*-representation as follows:

i) Let the p-representation be a 6-tuple
$R = (l, \{S_q\}, \{\alpha_q\}, \{\mu_q\}, \{\zeta_q\}, \{\phi_q\})$.

ii) l is a p-bounded assignment of natural numbers to finite field indices.

iii) $S_q \subseteq \mathbf{B}^{l(q)}$ is the set of bit string representations of $\mathbf{F}_q$-elements.

iv) $\{\alpha_q, \mu_q : S_q \times S_q \rightarrow S_q, \ \ \zeta_q : S_q \rightarrow \mathbf{B}\}$ are p-bounded families of *boolean* circuits.

v) $\phi_q : S_q \rightarrow \mathbf{F}_q$ defines the semantics of the representation:

$$\begin{array}{rcl} \phi_q(\alpha_q(\bar{x}, \bar{y})) & = & \phi_q(\bar{x}) + \phi_q(\bar{y}) \\ \phi_q(\mu_q(\bar{x}, \bar{y})) & = & \phi_q(\bar{x}) \cdot \phi_q(\bar{y}) \\ \zeta_q(\bar{x}) = 1 & \text{iff} & \phi_q(\bar{x}) = 0 \end{array}$$

Apart from arithmetic operations, a representation must include a zero-recognizer. This requirement follows from the fact that a single field element may have exponentially many different bit string representations within a single p-representation although the p-boundedness does assume that only "short" bit string representations are possible. Thus it is possible to compute predicates. Without this facility it may not be possible to recognise the output at all, in which case the computation cannot be regarded as meaningful.

Theorem 3

There exists a p-representation of the finite fields.

Proof of Theorem 3:

By Theorem 1-a, we need only consider the representation of prime fields. For p prime, $\mathbf{F}_p$ is isomorphic to $\mathbf{Z}_p$ (see [Lidl 83]). Let $l(p) = \lceil logp \rceil$, $S_p = \{b \in \mathbf{B}^{l(p)} \mid b \leq p - 1$, when b is interpreted as a binary number$\}$, which is the standard binary representation of elements in $\mathbf{Z}_p$. Addition and multiplication are implemented as the corresponding operations for binary numbers followed by a modulo p operation. This is easily done by p-bounded boolean circuits. The zero-recognizer is immediate since $(0,0,\ldots,0)$ is the only representation of 0.

$\square$

The representation constructed in the above proof is denoted the standard representation.

To address the question of efficiency we must compare general and arithmetic solutions to a finite field problem. Therefore we define:

- A general solution to a finite field computational problem $\Delta = \{\delta_q\}$ consists of a p-representation $R = (l, \{S_q\}, \{\alpha_q\}, \{\mu_q\}, \{\zeta_q\}, \{\phi_q\})$, and a family of boolean circuits $\{b_q\}$ such that b_q computes a unary function on S_q and $\phi_q \circ b_q = \delta_q \circ \phi_q$.

Arithmetic is efficient if it is possible to simulate a general solution by arithmetic with only a polynomial increase in circuit size. Arithmetic can simulate the boolean operations efficiently (see proof of Theorem 4). Therefore, all that is left to consider is whether arithmetic can be used to compute a p-representation of a field element efficiently. More formally:

i) A p-representation $R = (l, \{S_q\}, \{\alpha_q\}, \{\mu_q\}, \{\zeta_q\}, \{\phi_q\})$ is strong iff there exists two p-bounded families of arithmetic circuits

$$\{i_q : \mathbf{B}^{l(q)} (\subseteq \mathbf{F}^{l(q)}) \rightarrow \mathbf{F}_q\}$$

and

$$\{o_q : \mathbf{F}_q \rightarrow \mathbf{B}^{l(q)} (\subseteq \mathbf{F}^{l(q)})\}$$

such that $i_q(\bar{x}) = \phi_q(\bar{x})$ and $\phi_q(o_q(y)) = y$ for $\bar{x} \in S_q \subseteq \mathbf{F}^{l(q)}$ and $y \in \mathbf{F}_q$.

ii) A p-representation $R = (l, \{S_q\}, \{\alpha_q\}, \{\mu_q\}, \{\zeta_q\}, \{\phi_q\})$ p-reduces to another p-representation $R = (l', \{S'_q\}, \{\alpha'_q\}, \{\mu'_q\}, \{\zeta'_q\}, \{\phi'_q\})$ ($R \leq_p R'$) iff there exists a p-bounded family of circuits $\{\rho_q : S_q \rightarrow S'_q\}$ using boolean operations such that $\phi_q = \phi'_q \circ \rho_q$.

iii) R is p-equivalent to R' ($R \equiv_p R'$) iff ($R \leq_p R'$ and $R' \leq_p R$).

Intuitively, a strong p-representation is one where arithmetic can efficiently access the bits of the representation. For later use, we note that a strong representation is p-equivalent to any other representation:

Lemma 1

Given p-representations R, R' such that R is strong, then $R \equiv_p R'$.

Proof of Lemma 1:

$\underline{R \leq_p R'}$
If R is strong then a p-bounded family of arithmetic circuits $\{i_q\}$ exists. If i_q is implemented using representation R', this gives a p-bounded family of boolean circuits that takes the R' representation of the R bit string representing a field element and produces the R' representation of that element. All that remains is to choose two bit strings a_q and b_q with $\phi'_q(a_q) = 0$ and $\phi'_q(b_q) = 1$, and to prefix each input to the new circuit where an R' representation of zero or one is required by a small circuit that takes a boolean zero or one and switches into the old input either a_q or b_q as appropriate.

$\underline{R' \leq_p R}$
R is strong, so that $\{o_q\}$ exists for R as in the definition. If o_q is implemented in representation R', the input is now the R' representation of a field element, and the output is the R' representation of the bit-string that that is the R representation of the same field element. Now use ζ'_q to build a small circuit that will recognise R' representations of zero and one and output the corresponding boolean value. Appending one of these to each output where a zero or one in representation R' will appear achieves the desired result.

$\square$

It will turn out that arithmetic access to the standard p-representation is crucial to assessing the efficiency of arithmetic. For this reason we define:

- The last bit problem for *prime* fields $L = \{l_p : \mathbf{F}_p \to \{0,1\} \subseteq \mathbf{F}_p\}$, where $l_p(a)$ is the last bit in the standard representation of a; i.e., if we identify $\mathbf{F}_p$ with $\{0, 1, \ldots, p-1\} \subseteq \mathbf{N}$, l_p is zero on even numbers and one on odd numbers.

Theorem 4

The following statements are equivalent:

(1) Arithmetic over finite fields is efficient.

(2) L has a p-bounded arithmetic solution.

(3) There exists a strong p-representation.

(4) All p-representations are strong.

Proof of Theorem 4:

(1) $\Rightarrow$ (2): Using the standard representation, L has a constant size general solution and therefore by (1) a p-bounded arithmetic solution.

(2) $\Rightarrow$ (3): Assuming (2), the standard representation is strong. By Theorem 1 we need only consider the prime field case. $\{o_p\}$ is built using $O(log\, p)$ copies of a p-bounded arithmetic solution for L, whereas $\{i_p\}$ is directly constructed.

(3) $\Rightarrow$ (4): Given an arbitrary p-representation R' and a strong one R, then $R \equiv_p R'$ by Lemma 1. This implies that R' is strong since $\{i'_q\}$ and $\{o'_q\}$ can be obtained from $\{i_q\}$ and $\{o_q\}$ by prefixing and postfixing respectively arithmetic simulations of the small boolean circuits that translate from R' to R and from R to R'.

(4) $\Rightarrow$ (1): Let $\{\delta_q\}$ be an arithmetic problem which has a general solution consisting of a representation R and a family of boolean circuits $\{b_q\}$ of complexity f. Let $\{tr(b_q)\}$ be a family of arithmetic circuits simulating the boolean circuits $\{b_q\}$ on zero-one field values. This can easily be achieved by using multiplication for *and* and $1 - x$ to compute *not* x. Let $\{i_q\}$ and $\{o_q\}$ be the arithmetic circuits given by the definition of strong for R. In this case, $\{i_q \circ tr(b_q) \circ o_q\}$ is an arithmetic solution to $\{\delta_q\}$ of complexity at most $t + 3f$, for some polynomial t.

□

We have now established that the efficiency of the finite field arithmetic depends upon the arithmetic complexity of the last bit problem for prime fields. For this reason, we will hereafter confine our attention to prime fields.

We shall see later that l_p written as a polynomial over $\mathbf{Z}_p$ has degree at least $p-1$. Thus it follows that any arithmetic circuit to compute l_p will have depth at least $O(log_2 p)$. This is because in an arithmetic circuit the degree can at most double as the depth is increased by one. Similar remarks hold for the polynomials in Theorem 1. Even at bounded characteristic, results asserting the efficiency of arithmetic using circuit depth as a complexity measure do not exist. Thus the finite field model of computation is inherently sequential.

Note also that constants are not necessary in arithmetic computations in $\mathbf{Z}_p$. Given a circuit using constants, the constants can be replaced by outputs from an additional circuit that computes them from 1 using short addition chains (see [Knuth 81]). Then 1 can be replaced by the output of a small circuit that computes x^{p-1} when the input to the original circuit is x. This always has the value 1 except when x is 0 (see [Lidl 83]). Thus the new circuit computes correctly except on input zero, where it must compute zero as it has no constants. Restricting our attention to functions h with $h(0) = 0$ is no great constraint. It is easy to verify that the above construction involves only a small increase in circuit size. In fact, for functions that can be computed without constants, constants are not necessary in order to compute them efficiently over any finite field [Bøgestrand 87].

We shall see that some common operators over prime fields can replace the last bit problem in Theorem 4, apparently due to these operators' dependence on the standard representation:

- An n-ary prime field operator $A = \{\alpha_p\}$ is a family of functions, $\alpha_p : \mathbf{F}_p^n \to \mathbf{F}_p$. Observe that the last bit problem L is a unary prime field operator. Given the standard representation $R = (\ldots, \{\phi_p\})$, we know ϕ_p is invertable. This fact allows us to define the binary prime field operators:

$$\begin{array}{ll} min\text{:} & min_p(a,b) = \phi_p(min_{\mathbf{Z}}(\phi_p^{-1}(a), \phi_p^{-1}(b))) \\ exp\text{:} & exp_p(a,b) = a^{\phi_p^{-1}(b)} \end{array}$$

In order to state the equivalence results formally, we need some definitions:

- A circuit family $\{a_p\}$ consists of A-arithmetic circuits if a_p is allowed to use the operator α_p apart from $\mathbf{F}_p$-arithmetic ($A = \{\alpha_p\}$).
- An n-ary prime field operator $A = \{\alpha_p\}$ reduces to an operator B arithmetically ($A \leq_a B$) iff there exists a p-bounded family of B-arithmetic circuits $\{a_p : \mathbf{F}_p^n \rightarrow \mathbf{F}_p\}$ such that a_p and α_p are functionally identical.
- A and B are arithmetically equivalent ($A \equiv_a B$) iff ($A \leq_a B$ and $B \leq_a A$).

Theorem 5

$op \in \{min, exp\}$ implies $op \equiv_a L$.

Proof of Theorem 5:

Observe that $\phi_p^{-1} : \mathbf{F}_p \rightarrow \mathbf{B}^{\lceil log\, p \rceil} \subseteq \mathbf{F}_p^{\lceil log\, p \rceil}$ is realized by a p-bounded family of L-arithmetic circuits, from which it follows that $op \leq_a L$ for $op \in \{min, exp\}$.

Conversely:

$$\begin{aligned} l_p(a) &= \frac{2}{a}[min_p(a, \frac{a}{2}) - \frac{a}{2}] \text{ since } min_p(a, \frac{a}{2}) = \begin{cases} a, & \text{“}a\text{ odd”} \\ \frac{a}{2}, & \text{“}a\text{ even”} \end{cases} \\ l_p(a) &= \frac{1}{2}[1 - exp_p(p-1, 2)] \text{ since } exp_p(p-1, a) = \begin{cases} p-1, & \text{“}a\text{ odd”} \\ 1, & \text{“}a\text{ even”} \end{cases} \end{aligned}$$

$\square$

Consequently, the arithmetic complexity of any of the operators, L, min, or exp determines the efficiency of arithmetic.

The efficiency of arithmetic is also connected to the possibility of doing modulo-arithmetic within the prime fields:

- Let the function $f : \mathbf{N} \rightarrow \mathbf{N}$ be chosen such that $f(p) \neq p$ for all primes p. A prime field representation of modulo f-arithmetic is a tuple:

  $$R_f = (l, \{S_p\}, \{\oplus_p\}, \{\odot_p\}, \{\ominus_p\}, \{\oslash_p\}, \{ⓩ_p\}, \{i_p\}, \{o_p\}, \{\psi_p\})$$

  such that:

$$
\begin{aligned}
&l : \mathbf{N} \to \mathbf{N} \\
&S_p \subseteq \mathbf{F}_p^{l(p)} \\
&\oplus_p, \ominus_p, \odot_p, \oslash_p : S_p \times S_p \to S_p \\
&ⓩ_p : S_p \to \{0,1\} \subseteq \mathbf{F}_p \\
&i_p : \mathbf{F}_p \to S_p \\
&o_p : S_p \to \mathbf{F}_p
\end{aligned}
$$

The semantics is given by $\psi_p : S_p \to \mathbf{Z}_{f(p)}$.

$$
\begin{aligned}
\psi_p(\bar{x} \oplus_p \bar{y}) &= [\psi_p(\bar{x}) + \psi_p(\bar{y})] \bmod f(p) \\
\psi_p(\bar{x} \ominus_p \bar{y}) &= [\psi_p(\bar{x}) - \psi_p(\bar{y})] \bmod f(p) \\
\psi_p(\bar{x} \odot_p \bar{y}) &= [\psi_p(\bar{x}) \cdot \psi_p(\bar{y})] \bmod f(p) \\
\psi_p(\bar{x} \oslash_p \bar{y}) &= \begin{cases} [\psi_p(\bar{x}) / \psi_p(\bar{y})] \bmod f(p) & \text{if } \psi_p(\bar{y}) \text{is a unit in } \mathbf{Z}_{f(p)} \\ \psi_p(\bar{x}) & \text{otherwise} \end{cases} \\
ⓩ_p(a) = 1 \ \text{ iff } \ \psi_p(a) = 0
\end{aligned}
$$

In the following, let ϕ_p denote the semantic function of the standard prime field representation:

$$
\begin{aligned}
\psi_p(i_p(a)) &= \quad \phi_p^{-1}(a) \bmod f(p) \quad \text{for } a \in \mathbf{F}_p \\
o_p(\bar{x}) &= \quad \phi_p(\psi_p(\bar{x}) \bmod p) \quad \text{for } \bar{x} \in S_p
\end{aligned}
$$

By generalizing arithmetic equivalence from dealing with single operators on $\mathbf{F}_p$ to encompass sets of operators on $\mathbf{F}_p^n$ (in the obvious way) we may state:

Theorem 6

For any function $f : \mathbf{N} \to \mathbf{N} \setminus \{0,1\}$ such that $f(p) \neq p$ for all primes p, let $R_f = (\ldots, \{\oplus_p\}, \{\odot_p\}, \{\ominus_p\}, \{\oslash_p\}, \{ⓩ_p\}, \{i_p\}, \{o_p\})$ be an arbitrary prime field representation of modulo f arithmetic. In such case:

$$L \leq_a \{\{\oplus_p\}, \{\odot_p\}, \{\ominus_p\}, \{\oslash_p\}, \{ⓩ_p\}, \{i_p\}, \{o_p\}\}$$

Proof of Theorem 6:

For

$\underline{f(p) > p}$: $l_p(a) = 1 - \textcircled{z}_p(2 \odot_p i_p(\frac{a}{2}) \ominus_p i_p(a))$

$\underline{f(p) < p,\ f(p) \text{ even}}$: $l_p(a) = \textcircled{z}_p(\frac{f(p)}{2} \odot_p i_p(a) \ominus_p \frac{f(p)}{2})$

$\underline{f(p) < p,\ f(p) \text{ odd}}$: Observe that $l_p((o_p(i_p(a))))$ is zero iff $2 \cdot o_p(i_p(a) \oslash_p 2) - o_p(i_p(a))$ is zero. Since $o_p(i_p(a)) =$ "$a \bmod f(p)$", we can get the parity of all digits in a base $f(p)$ representation of a in $log_{f(p)}p \leq log_2(p)$ steps as above.

$\square$

$\mathbf{Z}_{p^2}$-arithmetic and Witt-vectors

The result just obtained establishes that arithmetic is efficient if *mod p*-arithmetic can efficiently simulate $mod\, f(p)$-arithmetic. This section deals specifically with simulations of $\mathbf{Z}_{p^2}$-arithmetic by means of $\mathbf{Z}_p$-arithmetic.

By using a nonstandard Witt-vector representation of $\mathbf{Z}_{p^2}$, the last bit problem is split into two apparently easier problems: one of which is computing remainder modulo p in $\mathbf{Z}_{p^2}$, the other being computing the additive carry in Witt-vector representation of $\mathbf{Z}_{p^2}$.

In what follows, we identify $\mathbf{F}_p$ (and $\mathbf{Z}_p$) with the standard representation, which is the natural numbers $\{0, 1, \ldots, p-1\}$ equipped with $mod p$-arithmetic. Similarly $\mathbf{Z}_{p^2}$ is identified with the set $\{0, 1, \ldots, p^2-1\}$ equipped with *mod* p^2-arithmetic. We shall henceforth assume p always denotes an odd prime. Ignoring the prime 2 is no severe restriction, since any algorithm or reduction can easily be fixed for a finite number of input sizes.

The subscript p will be omitted in the following when no ambiguity arises.

The standard representation of $\mathbf{Z}_{p^2}$ is the following:

$$S_2(\mathbf{F}_p) \;=\; (l, \{S_p\}, \{\oplus_p\}, \{\odot_p\}, \{\ominus_p\}, \{\oslash_p\}, \{\circledz_p\}, \{i_p\}, \{o_p\}, \{\psi_p\})$$

where:

i) l is identically 2

ii) $S_p = \mathbf{F}_p \times \mathbf{F}_p$

iii) $\psi_p : \mathbf{F}_p^2 \rightarrow \mathbf{Z}_{p^2} \qquad \psi_p(x_0, x_1) = x_0 + p x_1$

(i)-(iii) determine the representation uniquely. Insertion and retraction are easy to compute:

$$\begin{aligned} i_p : \mathbf{F}_p \rightarrow \mathbf{F}_p^2, &\quad i_p(x) = (x, 0) \\ o_p : \mathbf{F}_p^2 \rightarrow \mathbf{F}_p, &\quad o_p(x_0, x_1) = x_0 \end{aligned}$$

Since ψ_p is a bijection, the identity relation (specifically $\circledz_p$) is also easy to compute. Arithmetic is more difficult. Let the multiplicative and additive carries be represented by binary prime field operators π, σ such that:

$$\begin{aligned}(x_0, x_1) \oplus (y_0, y_1) &= (x_0 + y_0,\ x_1 + y_1 + \sigma(x_0, y_0)) \\ (x_0, x_1) \odot (y_0, y_1) &= (x_0 \cdot y_0,\ x_0 \cdot y_1 + x_1 \cdot y_0 + \pi(x_0, y_0))\end{aligned}$$

It is fairly obvious that:

$$l_p(x) = 0 \quad \text{iff} \quad i_p(2) \odot_p i_p(\tfrac{x}{2}) = i_p(x) \ \text{iff} \quad \pi(2, \tfrac{x}{2}) = 0$$

and

$$l_p(x) = 0 \quad \text{iff} \quad i_p(\tfrac{x}{2}) \oplus_p i_p(\tfrac{x}{2}) = i_p(x) \ \text{iff} \quad \sigma(\tfrac{x}{2}, \tfrac{x}{2}) = 0$$

Consequently $S_2(\mathbf{F}_p)$ is easily realizable by $\mathbf{F}_p$-arithmetic except possibly for the carry-functions:

Theorem 7

$L \equiv_a \pi \equiv_a \sigma$

Witt-vectors are an ingeneous construction for representing Abelian extensions of fields. The theory of Witt-vectors is described in [Encyclopedia 80, Greenberg 69, Hasse 49, Jacobson 64, Witt 37]. For the present, we simply render the Witt-vector representation of $\mathbf{Z}_{p^2}$ in a way that emphasizes the carry-less nature of this representation. This property makes the representation specifically attractive for our purposes.

The Witt representation of $\mathbf{Z}_{p^2}$ is:

$$W_2(\mathbf{F}_p) \ = \ (l, \{S_p\}, \{\oplus_p\}, \{\odot_p\}, \{\ominus_p\}, \{\oslash_p\}, \{ⓩ_p\}, \{i_p\}, \{o_p\}, \{\psi_p\})$$

where

i) l is identically 2

ii) $S_p = \mathbf{F}_p \times \mathbf{F}_p$

iii) $(x_0, x_1) \odot_p (y_0, y_1) = (x_0 \cdot y_0,\ x_0 \cdot y_1 + x_1 \cdot y_0)$

iv) $(0, x_1) \oplus_p (0, y_1) = (0,\ x_1 + y_1)$

v) $(x_0, 0) \oplus_p (0, y_1) = (x_0, y_1)$

vi) $\psi_p(0, 1) = p$

We will verify later that properties i-vi determine the representation uniquely. Properties i and ii are shared with the standard representation; whereas, iii specifies multiplication as a convolution without carry. Property iii can only determine the representation up to an automorphism with respect to the multiplicative structure on $\mathbf{Z}_{p^2}$. Addition cannot be specified without carry, since the resulting structure would be of characteristic p. However, $\mathbf{Z}_{p^2}$ contains a unique additive subgroup of order p, consisting of the non-units. Property iv forces a simple addition for this subgroup. Property v specifies addition between some units and non-units. Because of properties iii-v, $W_2(\mathbf{F})$ offers a very simple (if not the simplest possible) simulation of $\mathbf{Z}_{p^2}$-arithmetic by means of $\mathbf{F}_p$-arithmetic. Property vi ensures that the characteristic is p^2.

The additive carry in $W_2(\mathbf{F}_p)$ is the binary prime field operator g defined by:

$$g_p(x, y) \overset{p}{\equiv} \frac{x^p + y^p - (x + y)^p}{p}$$

where $\overset{p}{\equiv}$ denotes congruence modulo p. The righthandside expression is p-integral since:

$$x^p + y^p - (x + y)^p = -\sum_{n=1}^{p-1} \binom{p}{n} x^n y^{p-n}.$$

and the binomial coefficients are all divisible by p. Similarly the semantics of $W_2(\mathbf{F}_p)$ is expressed by means of a unary prime field operator f, defined by:

$$f_p(x) \overset{p}{\equiv} \frac{x - x^p}{p}$$

In proving the correctness of the representation, we need a lemma stating that g has the required carry properties:

Lemma 2

A) $g(x, 0) = 0$

B) $g(x, -x) = 0$

C) $g(x, y) = g(y, x)$

D) $g(x,y)+g(x+y,z)=g(x,y+z)+g(y,z)$

E) $g(xy,xz)=xg(y,z)$

Proof of Lemma 2:

All identities (A)-(E) are simple consequences of the definition of g.

$\square$

The correctness of $W_2(\mathbf{F}_p)$ may now be stated and proved.

Theorem 8

(a) $W_2(\mathbf{F}_p)$ uniquely determines a representation of the ring $\mathbf{Z}_{p^2}$.

(b) $(x_0,x_1)\oplus_p(y_0,y_1)=(x_0+y_0,x_1+y_1+g_p(x_0,y_0))$

(c) $\psi_p(x_0,x_1)=x_0+p(x_1-f_p(x_0))$

Proof of Theorem 8:

We shall begin by establishing (b), (c) from which the uniqueness of the representation follows, and end by establishing existence. In what follows, we drop special representation syntax and write $+,\cdot,-,/$ for $\oplus_p,\odot_p,\ominus_p,\oslash_p$ when no ambiguity arises; similarly, we drop the index p for f,g. Initially the consequences of iii are explored.

Observe that the representation of 0 and 1 are uniquely determined by $0\cdot x=0$ and $1\cdot x=x$, giving:

(1) $\psi_p(0,0)=0$

(2) $\psi_p(1,0)=1$

In addition, the non-units of $\mathbf{Z}_{p^2}$ are precisely the elements of the set $\{x \mid x^2=0\}$, which is easily recognized in the Witt-representation:

(3) $\{(0,x)\mid x\in\mathbf{Z}_p\}$ represents $\{xp\mid x\in\mathbf{Z}_p\}$

The multiplicative subgroup of $\mathbf{Z}_{p^2}$ has order $p(p-1)$ and is cyclic. Consequently there exist unique subgroups of orders $2,p,p-1$. By noting inductively that $(x_0,x_1)^k=(x_o^k,kx_0^{k-1}x_1)$, we get:

(4) $\psi_p(-1,0)=p^2-1$, since $\{(-1,0),(1,0)\}$ represents $\{x\mid x^2=1\}$

(5) $\{(x,0) \mid x \in \mathbf{Z}_p^*\}$ represents $\{x \mid x^{p-1} = 1\}$

(6) $\{(1,x) \mid x \in \mathbf{Z}_p\}$ represents $\{x \mid x^p = 1\}$ which is $\{1 + xp \mid x \in \mathbf{Z}_p\}$

An implication of (4) is:

(7) $-(x_0, x_1) = (-x_0, -x_1)$

We also note:

(7.5) $(x_0, x_1)^{-1} = (x_0^{-1}, -x_0^{-2}. x_1)$ for $x_0 \neq 0$

Next we investigate consequences of iv. By computing $(0,1)\cdot((x_0,0)+(y_0,0))$ in two different ways, we get:

(8) $(x_0,0)+(y_0,0) = (x_0+y_0, \gamma(x_0,y_0))$ for some binary prime field function γ.

By use of v we get:

(9) $(x_0,x_1) + (y_0,y_1) = (x_0 + y_0, x_1 + y_1 + \gamma(x_0,y_0))$

By (2) and (9) it follows that $\psi_p(x_0,x_1) \overset{p}{\equiv} x_0$, and that specifically $x_0^p \overset{p^2}{\equiv} \psi_p(x_0,0)$ by (5). In addition by vi and iv, $\psi_p(0,x_1) = x_1 p$. Hence the semantics of $W_2(\mathbf{F}_p)$ is:

(10) $\psi_p(x_0,x_1) = \psi_p(x_0,0) + \psi_p(0,x_1) \overset{p^2}{\equiv} x_0^p + x_1 p \overset{p^2}{\equiv} x_0 + p(x_1 - f(x_0))$

and consequently:

(11) $\psi_p(x_0, f(x_0)) = x_0$

We have now established the uniqueness part of Theorem 8 and may confirm that $\gamma = g$ using the $W_2(\mathbf{F}_p)$-representation:

$$\begin{aligned} g(x_0,y_0) &= \frac{1}{p}[x_0^p + y_0^p - (x_0+y_0)^p] \\ &= \frac{1}{p}[(x_0,0) + (y_0,0) - (x_0+y_0,0)] \\ &= \frac{1}{p}[(0,\gamma(x_0,y_0))] = \gamma(x_0,y_0) \end{aligned}$$

Finally, we must verify that $W_2(\mathbf{F})$ is a commutative ring: this is checked by trivial computations using Lemma 2, iii, (b) and (1), (2), (7) and (7.5).

$\square$

By exploiting commutative diagrams between $W_2(\mathbf{F}_p)$ and $S_2(\mathbf{F}_p)$ we may reduce L and the carry-functions to f:

Theorem 9

1) $g \leq_a f$:
$$g(x,1) = \begin{cases} f(x+1) - f(x) & \text{when } x \neq -1 \\ 0 & \text{otherwise} \end{cases}$$

2) $\pi \leq_a f$:
$$\pi(x,y) = x \cdot f(y) + y \cdot f(x) - f(xy)$$

3) $\sigma \leq_a f$:
$$\sigma(x,y) = f(x) + f(y) + g(x,y) - f(x+y)$$

4) $L \leq_a f$:
$$l(x) = 2f(\frac{x}{2}) + \frac{x}{2}f(2) - f(x)$$

Proof of Theorem 9:

1) The stated identity is a simple consequence of the definitions of f and g. The full reduction ($g \leq_a f$) follows by the distributivity of g (Lemma 2).

2) Routing the pair $(x_0, x_1), (y_0, y_1)$ through the following commutative diagram in two different ways establishes 2 (where $\Psi(x_0, x_1) = (x_0, x_1 - f(x_0))$ is the canonical isomorphism between the two representations of $\mathbf{Z}_{p^2}$):

$$\begin{tikzcd}
W_2(\mathbf{F}) \times W_2(\mathbf{F}) \arrow[r, "\Psi \times \Psi"] \arrow[d, "\odot"] & S_2(\mathbf{F}) \times S_2(\mathbf{F}) \arrow[d, "\odot"] \\
W_2(\mathbf{F}) \arrow[r, "\Psi"] & S_2(\mathbf{F})
\end{tikzcd}$$

3) Similar to 2, except using addition.

4) Follows from 2, by observing that $l(x) = \pi(2, \frac{x}{2})$.

$\square$

Viewed as a computational problem, f certainly has a p-bounded general solution, which implies the existence of possibly non-uniform reductions from f to L and σ. Conversely (by Theorem 9), there exists trivial uniform reductions from L, π, σ and g to f. This indicates that f is canonical among the problems that are arithmetically equivalent to L. Clearly, the determination of f's arithmetic complexity resolves the question of the efficiency of finite field arithmetic.

Similarly, g may be viewed as a computational problem. Apparently there is no simple reduction of f to g. By Theorem 9 such a reduction would imply summation of the g function. Summation is related to the Bernoulli polynomials and by using $W_2(\mathbf{F}_p)$ we can split the f-problem into two hopefully simpler problems: namely g – the additive carry in $W_2(\mathbf{F}_p)$, and m – the modulo-p operation in $\mathbf{Z}_{p^2}$. It turns out that g is a pseudo-logarithm and m is almost a Bernoulli polynomial.

We start by transferring our previous definitions of an arithmetic model of computation to the family of rings $\mathbf{Z}_{p^2}$:

i) A $\mathbf{Z}_{p^2}$-computational problem $\Delta = \{\delta_p\}$ is a family of functions δ_p: $\mathbf{Z}_{p^2} \rightarrow \mathbf{Z}_{p^2}$, one function for every prime.

ii) A family of $\mathbf{Z}_{p^2}$-arithmetic circuits $\{a_p\}$ contains for every prime a circuit a_p that uses $\mathbf{Z}_{p^2}$-arithmetic $(+, \cdot)$ and $\mathbf{Z}_{p^2}$-constants.

iii) An arithmetic solution to Δ consists of a family $\{a_p\}$ of $\mathbf{Z}_{p^2}$-arithmetic circuits such that a_p computes a unary function and a_p and δ_p are functionally identical for all p.

iv) The input size for a $\mathbf{Z}_{p^2}$-problem or circuit indexed by p is taken to be *log p*. Given this, the previous definitions of complexity and p-boundedness apply.

v) The modulo-problem $m = \{m_p\}$, is given by $m_p(a + bp) = a$ for $a, b \in \mathbf{Z}_p$.

Ignoring the fact that some $\mathbf{Z}_{p^2}$-problems have no arithmetic solutions, it seems clear that the modulo problem plays a role in $\mathbf{Z}_{p^2}$ similar to the role of the last bit problem in $\mathbf{Z}_p$. In fact f, and thus L, may be reduced to g and m. It should be noted that some $\mathbf{Z}_{p^2}$-problems have no arithmetic solutions. As an example consider $\{\delta\}$, where $\delta_p(a+bp)=b$, for a, b in $\mathbf{Z}_p$.

Theorem 10

1. If f has a p-bounded solution, so has m.
2. If m and g both have p-bounded arithmetic solutions, then f has a p-bounded arithmetic solution.

Proof of Theorem 10:

1. Assume $\{\varphi_p\}$ is a p-bounded arithmetic solution to f, and construct a family of $\mathbf{Z}_{p^2}$-arithmetic circuits $\{\hat{\varphi}_p\}$ identical to $\{\varphi_p\}$ except for the change of arithmetic basis. Observe that $m_p(x) \overset{p^2}{\equiv} x^p + p\hat{\varphi}_p(x)$. This follows from the observation that in the Witt representation $m_p((x_0,x_1)) = (x_0, f(x_0))$.
2. Let $\{\mu_p\}$ be a p-bounded arithmetic solution to m, i.e. μ_p uses $\mathbf{Z}_{p^2}$-arithmetic. Assuming g has a p-bounded arithmetic solution, we may find p-bounded $\mathbf{Z}_p$-arithmetic circuits for $\oplus_p, \odot_p$ in the $W_2(\mathbf{F}_p)$ representation of $\mathbf{Z}_{p^2}$. Substitute these into $\{\mu_p\}$ to obtain a p-bounded family of $\mathbf{Z}_p$-arithmetic circuits
$\{\hat{\mu}_p : \mathbf{Z}_p \times \mathbf{Z}_p \to \mathbf{Z}_p \times \mathbf{Z}_p\}$. A p-bounded arithmetic solution to f follows immediately by observing that $f_p(x)$ equals the second component of $\hat{\mu}_p(x,0)$.

$\square$

In the following, m and g will be related respectively to the Bernoulli polynomials and the natural logarithm. These results indicate that m and g have quite different properties. This suggests that the reduction of f to $\{m,g\}$ is a true simplification.

A thorough introduction to Bernoulli polynomials may be found in [Lang 78, Nörlund 24]. First we restate the basic definitions:

- The infinite set of Bernoulli numbers $\{B_k\} \subseteq \mathbf{Q}$ is defined by

$$\frac{t}{e^t-1} = \sum_{k=0}^{\infty} B_k \frac{t^k}{k!}$$

and

- the infinite set of Bernoulli polynomials $\{B_k(x)\} \subseteq \mathbf{Q}[x]$ is defined by

$$\frac{te^{tx}}{e^t - 1} = \sum_{k=0}^{\infty} B_k(x) \frac{t^k}{k!}$$

And second, the basic propositions:

1. $B_0 = 1, B_1 = -\frac{1}{2}, B_{2k+1} = 0$ for $k \geq 1$.
2. Let α be a residue class mod $p-1$ and let $k, l \in \alpha$, and k, l different from zero:
   - (Kummer's Congruence:) if $0 \notin \alpha$ then B_k, B_l are p-integral and $\frac{B_k}{k} \overset{p}{\equiv} \frac{B_l}{l}$
   - (Von Staudt's Congruence:) if $0 \in \alpha$ then $pB_k \overset{p}{\equiv} -1$
3. $B_k(x) = \Sigma_{i=0}^{k} \begin{pmatrix} k \\ i \end{pmatrix} B_{k-i} x^i$
4. $\frac{d}{dx} B_k(x) = k \cdot B_{k-1}(x)$ and $B_k(x+1) - B_k(x) = k \cdot x^{k-1}$
5. $B_k(x) = N^{k-1} \Sigma_{a=0}^{N-1} B_k(\frac{x+a}{N})$, for $N \geq 1$ and
   $B_k(x) = (-1)^{k-1} B_k(1-x)$

Proposition 2 assures that $B_1, \ldots, B_{p-2}$ and pB_{p-1} are defined in $\mathbf{Z}_p$ and $\mathbf{Z}_{p^2}$. In addition Proposition 3 assures that $B_k(x)$ is well-defined in $\mathbf{Z}_p$ when $k < p-1$.

Using Propositions 1-5, m may be characterized:

Theorem 11

1. $m_p(x) \overset{p^2}{\equiv} B_p(x+1) + (1-p)x$
2. $m_p(x) = \Sigma_{n=1}^{p} a_n x^n$, where $a_n = p \cdot \frac{B_{p-n}}{n}$ for $2 \leq n \leq p-1$, $a_p = 1$ and $a_1 = p \cdot B_{p-1} + (1-p) = p \Sigma_{k=1}^{p-2} \frac{B_k}{k}$

Proof of Theorem 11:

1. By Proposition 4, $B_p(x+1) = p \Sigma_{n=0}^{x} n^{p-1}$, since $B_p(0) = 0$. However $\Sigma_{n=0}^{x} n^{p-1} \overset{p}{\equiv} a - b$ for $x = a + bp$, where $a, b \in \mathbf{Z}_p$. Hence, $B_p(x+1) \overset{p^2}{\equiv} px + m_p(x) - x$ from which the result follows.

2. The power series expansion follows from 1. and Propositions 1, 3 and 4 when using $\begin{pmatrix} p-1 \\ i \end{pmatrix} \overset{p}{\equiv} (-1)^i$. The alternative definition of a_1 comes from $m_p(1) = 1$.

□

This result implies that identities for the Bernoulli polynomials such as Proposition 5 may be exploited when computing $m_p(x)$. The power series for m easily gives a power series for f:

Theorem 12

$f_p(x) = \sum_{n=1}^{p-1} b_n x^n$, where

$$\begin{aligned} b_n &= \frac{B_{p-n}}{n}, \text{ for } 2 \le n \le p-1 \text{ and} \\ b_1 &= \frac{pB_{p-1}+1}{p} - 1 = \sum_{k=1}^{p-2} \frac{B_k}{k} \end{aligned}$$

Proof of Theorem 12:

This follows from Theorem 11(2) and the relation between m and f used in the proof of Theorem 10, i.e. $m_p(x) \overset{p^2}{\equiv} x^p + pf_p(x)$.

□

The identities for $B_p(x)$ can be interpreted as f identities, e.g. $f(x) + f(-x) = x^{p-1}$.

Next we turn to g. The natural logarithm will be denoted by ln. The facts that are needed to characterize g are as follows (see [Stewart 83]):

- $ln(1+x) = -\sum_{n=1}^{\infty} \frac{1}{n}(-x)^n, \; -1 < x \le 1$
- $ln(x \cdot y) = ln(x) + ln(y), \; x, y > 0$

This leads to the following result:

Theorem 13

(a) $g(x,1) = \sum_{n=1}^{p-1} \frac{1}{n}(-x)^n$

(b) $g(x,1) \overset{p}{\equiv} (x+1)[\frac{1}{p} ln(1+x^p) - ln(1+x)]$

Proof of Theorem 13:

(a) $g(x,y) \overset{p}{\equiv} -\frac{1}{p}\Sigma_{n=1}^{p-1} \begin{pmatrix} p \\ n \end{pmatrix} x^n y^{p-n} =$

$- \Sigma_{n=1}^{p-1} \frac{1}{n} \begin{pmatrix} p-1 \\ n-1 \end{pmatrix} x^n y^{p-n} \overset{p}{\equiv} \Sigma_{n=1}^{p-1} \frac{1}{n}(-x)^n y^{p-n}$

(b) $\frac{1}{p} ln(1+x^p) - ln(1+x) = -\frac{1}{p}\Sigma_n \frac{1}{n}(-x^p)^n + \Sigma_n \frac{1}{n}(-x)^n = \Sigma_{n \neq p} \frac{1}{n}(-x)^n$

$= \Sigma_{k=0}^{\infty} \Sigma_{n=1}^{p-1} \frac{1}{n+pk}(-x)^{n+pk} \overset{p}{\equiv} (\Sigma_{k=0}^{\infty}(-x)^k)(\Sigma_{k=1}^{p-1} \frac{1}{n}(-x)^n) \overset{p}{\equiv} \frac{1}{1+x} g(x,1)$

$\square$

Define a unary prime field operator k by

$$k(x) = \frac{g(1,-x)}{1-x} \quad \text{where } x \neq 1$$

This k fulfils:

Theorem 14

(a) $k \equiv_a g$

(b) $k(\frac{1}{x}) = k(x)$

(c) Let $a \neq 1$ have n distinct n'th roots $r_1, \ldots, r_n$. Then $k(a) = \Sigma_{k=1}^{n} k(r_i)$.

Proof of Theorem 14:

Observe that $k(x) \overset{p}{\equiv} \frac{1}{p} ln(\frac{x^p - 1}{(1-x)^p})$. Then

(a) is obvious.

(b) $k(\frac{1}{x}) \overset{p}{\equiv} \frac{1}{p} ln(\frac{1-(\frac{1}{x})^p}{(1-\frac{1}{x})^p}) = \frac{1}{p} ln(\frac{x^p-1}{(x-1)^p}) \overset{p}{\equiv} k(x)$

(c) The polynomial $(x^p - a)$ factors as $(x - r_1)\ldots(x - r_n)$. Specifically as $(1-a) = \Pi_{i=1}^{n}(1-r_i)$. Furthermore, $r_1^p, \ldots, r_n^p$ are the n'th roots of a^p, from which it follows that $1 - a^p = \Pi_{i=1}^{n}(1-r_i^p)$. Hence, $k(a) \overset{p}{\equiv} \frac{1}{p} ln(\frac{1-a^p}{(1-a)^p}) = \Sigma_{i=1}^{n} \frac{1}{p} ln(\frac{1-r_i^p}{(1-r_i)^p}) \overset{p}{\equiv} \Sigma_{i=1}^{n} k(r_i)$.

$\square$

Conclusion

We have shown that

- Finite field arithmetic is adequate for all efficient computations over fields of bounded characteristic.
- Finite field arithmetic is adequate for all efficient computations if and only if the function $f(x) \overset{p}{\equiv} (\frac{x-x^p}{p})$ has polynomial size arithmetic circuits over $\mathbf{Z}_p$.
- The function $f(x)$ has polynomial size arithmetic circuits if and only if the functions $g(x,y) \overset{p}{\equiv} (\frac{x^p+y^p-(x+y)^p}{p})$ and $m(x) \overset{p^2}{\equiv} x \ mod \ p$ both have polynomial size arithmetic circuits over $\mathbf{Z}_p$ and $\mathbf{Z}_{p^2}$ respectively.
- $m_p(x) \overset{p^2}{\equiv} B_p(x+1) + (1-p)x$ where B_p is the p'th Bernoulli polynomial.
- The power series for f is obtained by using the identity $m_p(x) \overset{p^2}{\equiv} x^p + p(f(x))$.
- $g(x,y) = -\sum_{i=1}^{p-1} \frac{(-1)^i x^i y^{p-i}}{i}$ and is thus formally related to the natural logarithm.

These findings suggest that:

- Because of the obscure nature of arithmetic computations for the function f, it is wise to consider more than just arithmetic operations when designing algorithms to compute the discrete logarithm.
- In the case of characteristic 2 finite fields, there is a natural bijection with bit strings. By using this, arithmetic lower bounds may be used to provide lower bounds on boolean circuit size.
- The status of arithmetic in finite fields can be resolved by investigating the complexity of m and g.

References

Adleman 78 ADLEMAN, L. Two theorems on random polynomial time. *Proceedings* 19th *IEEE Symp. on Foundations of Computer Science, pp. 75-83.* IEEE Computer Society, Los Angeles, 1978.

Adleman 79 ADLEMAN, L. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. *Proc.* 20th *IEEE Symp. on Foundations of Computer Science, pp. 55-60.* IEEE Computer Society, Los Angeles, 1979.

Aho 74 AHO, A. V., HOPCROFT, J. E. and ULLMAN, J. D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.

Bennett 81 BENNETT, C. H. and GILL, J. Relative to a random oracle $A, P^A \neq NP^A \neq co - NP^A$ with probability 1. *SIAM Journal on Computing* **10** *(1981), pp. 96-113.*

Berlekamp 68 BERLEKAMP, E. R. *Algebraic Coding Theory.* McGraw-Hill, New York, 1968.

Blum 84 BLUM, M. and MICALI, S. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing* **13** *(1984), pp. 850-864.*

Bøgestrand 87 BØGESTRAND, K. and LUND, C. Computations in Finite Fields. *Technical report DAIMI* **IR-71***, Computer Science Department, Aarhus University, Denmark, 1987.*

Borevich 66 BOREVICH, Z. I. and SHAFAREVICH, I. R. *Number Theory.* Academic Press, New York, 1966.

Borodin 77 BORODIN, A. On relating Time and Space to Size and Depth. *SIAM Journal on Computing* **6** *(1977), pp. 733-744.*

Borodin 82 BORODIN, A. Structured vs general models in computational complexity. *L'Enseignement Mathématique* **28** *(1982), pp. 171-189.*

Coppersmith 84 COPPERSMITH, D. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory* **IT-30** *(1984), pp. 587-594.*

Dixon 84 DIXON, J. D. Factorization and Primality Tests. *The American Mathematical Monthly* **91** *(1984), pp. 333-352.*

El Gamal 85 EL GAMAL, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* **IT-31** *(1985), pp. 469-472.*

Encyclopedia 80 ENCYCLOPEDIC DICTIONARY OF MATHEMATICS. *By the Mathematical Society of Japan.* Edited by Shôkichi Iyanaga and Yukiyosi Kawada. MIT-press, Cambridge, Mass., 1980.

Garey 79 GAREY, M. R. and JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York, 1979.

Gill 77 GILL, J. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing* **6** *(1977), pp. 675-695.*

Goldreich 86 GOLDREICH, O., GOLDWASSER, S. and MICALI, S. How to construct random functions. *Journal of the ACM* **33** *(1986), pp. 792-807.*

Greenberg 69 GREENBERG, M. J. *Lectures on Forms in many Variables.* W. A. Benjamin, New York, 1969.

Hasse 49 HASSE, H. *Zahlentheorie.* Akademie Verlag, Berlin, 1949.

Jacobson 64 JACOBSON, N. *Lectures in Abstract Algebra* **3**. *Theory of Fields and Galois Theory.* D. van Nostrand Company, Princeton, New Jersey, 1964.

Karp 80 KARP, R. M. and LIPTON, R. J. Some connections between nonuniform and uniform complexity classes. *Proceedings* 12th *ACM Symp. on Theory of Computing, pp. 302-309.* ACM, New York, 1980.

Karp 82 KARP, R. M. and LIPTON, R. J. Turing Machines that take advice. *L'Enseignement Mathématique* **28** *(1982), pp. 191-209.*

Knuth 81 KNUTH, D. E. *The Art of Computer Programming,* **2**: *Seminumerical Algorithms.* Addison-Wesley, Reading, Mass., 1981.

Lang 78 LANG, S. *Cyclotomic Fields.* Springer Verlag, New York, 1978.

Lidl 83 LIDL, R. and NIEDERREITER, H. *Finite Fields.* Encyclopedia of Mathematics and its Applications **20**. Addison-Wesley, Reading, Mass., 1983.

Nörlund 24 NÖRLUND, N. E. *Vorlesungen über Differenzenrechnung.* Springer Verlag, Berlin, 1924.

Odlyzko 84 ODLYZKO, A. M. Discrete logarithms in finite fields and their cryptographic significance. *Advances in Cryptology: Proceedings EUROCRYPT '84, pp. 224-314.* Lecture Notes in Computer Science **209**, Springer Verlag, Berlin, 1985.

Pippenger 79 PIPPENGER, N. On simultaneous resource bounds (preliminary version). *Proceedings* 20^{th} *IEEE Symp. on Foundations of Computer Science, pp. 307-311.* IEEE Computer Society, Los Angeles, 1979.

Rabin 80 RABIN, M. O. Probabilistic algorithms in finite fields. *SIAM Journal on Computing* **9** *(1980), pp. 273-280.*

Riesel 86 RIESEL, H. *Prime Numbers and Computer Methods for Factorization.* Birkhäuser, Boston, 1985.

Ruzzo 81 RUZZO, W. L. On uniform circuit complexity. *Journal of Computer and System Sciences* **22** *(1981), pp. 365-383.*

Savage 76 SAVAGE, J. E. *The Complexity of Computing.* Wiley, New York, 1976.

Schönhage 77 SCHÖNHAGE, A. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Informatica* **7** *(1977), pp. 395-398.*

Skyum 85 SKYUM, S. and VALIANT, L. G. A complexity Theory Based on Boolean Algebra. *Journal of the ACM* **32** *(1985), pp. 484-502.*

Stewart 83 STEWART, I. and TALL, D. *Complex Analysis.* Cambridge University Press, 1983.

Strassen 73 STRASSEN, V. Vermeidung von Divisionen. *J. Reine und Angewandte Mathematik* **264** *(1973), pp. 184-202.*

Valiant 82 VALIANT, L. G. Reducibility by algebraic projections. *L'Enseignement Mathématique* **28** *(1982), pp. 253-268.*

Valiant 83 VALIANT, L. G. An algebraic approach to computational complexity. *Proceedings of the International Congress of Mathematicians (1983), pp. 1637-1643.* North Holland, Amsterdam, 1984.

Witt 37 WITT, E. Zyklische Körper und Algebren der Characteristik p vom Grad p^n. *J. Reine und Angewandte Mathematik* **176** *(1937), pp. 126-140.*

Yao 82 YAO, A. C. Theory and applications of trapdoor functions. *Proceedings* 23rd *IEEE Symp. on Foundations of Computer Science, pp. 80-91.* IEEE Computer Society, Los Angeles, 1982.