Technical Report CS76001-R

THE USES OF FINITE FIELDS

By

T. C. Wesselkamper

January 1976

Language Research Center
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

The Uses of Finite Fields

T. C. Wesselkamper
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

Abstract

The paper is tutorial in nature, although some of the results are new. It reviews some of the elementary facts about the structure and construction of finite fields and hypothesizes a computer whose fundamental instruction set consists of the Galois field operations. Each total function is shown to be defined by a unique polynomial and this normal representation is also the minimal polynomial representation. A method is presented, due to Newton, for constructing the coefficients of the defining polynomial using divided differences. It is shown that under certain circumstances a total function may be more efficiently evaluated by a rational form with non-zero denominator. Finally a rational form representation is shown to be a natural representation for each partial function. In the light of these considerations the process of producing code for the hypothetical machine is almost entirely automated.

CR categories: 5.12, 6.1

AMS(MOS) classification: 12C05, 39A05, 02C05

The work reported herein was supported in part by the National Science Foundation Grant No. DCR74-18108.

I. Preliminaries

In this paper we limit our discussion to matters which involve computers with fixed word length. We do not limit our concern to binary machines. Ternary machines have been produced experimentally for some time [1, 2]. Recent results at Laval and at Paris show that ternary and even quinary machines can be fabricated from commercially available components [3, 4]. These results move ternary and quinary architecture into the realm of things economically feasible.

This paper is concerned with providing some answers to the question: Is there a general approach to machine instruction sets and therefore to programming which is independent of the base of the machine? An analogous question can be asked about primitive components for circuit design.

The results surveyed and reported in this paper demand that the reader be familiar with the mathematical notion of a field. Specifically, that if p is a prime, then the integers $\{0, 1, \ldots, p-1\} = E(p)$ form a finite or Galois field (of p elements) provided that addition and multiplication over E(p) are respectively defined to be integer addition and multiplication modulo p. This field $< E(p), +, * >$ is called GF(p).

Given a field GF(p) and an irreducible polynomial P(x) of degree n over GF(p), one may construct a finite field $GF(p^n)$ in the following fashion:

1. Let α be some symbol not in E(p).

2. Let the elements of $E(p^n)$ be of the form:

   $a_0 + a_1\alpha + a_2\alpha^2 + \ldots + a_{n-1}\alpha^{n-1}$, where for all i.

   $a_i \varepsilon E(p)$. That is,

   $$E(p^n) = \{\sum_{i=0}^{p-1} a_i\alpha^i \mid a_i \varepsilon E(p)\}.$$

3. Define addition over $E(p^n)$ to be coordinate-wise.

   That is, $(a_0 + a_1\alpha + \ldots + a_{n-1}\alpha^{n-1}) +$

   $(b_0 + b_1\alpha + \ldots + b_{n-1}\alpha^{n-1}) = (a_0 + b_0) + (a_1 + b_1)\alpha$

   $+ \ldots + (a_{n-1} + b_{n-1})\ \alpha^{n-1}$.

4. Define multiplication over $E(p^n)$ to be polynomial multiplication modulo $P(\alpha)$.

In the simplest case, when p = 2 and n = 2, there is only one monic irreducible polynomial of degree 2 over GF(2), namely, $P(x) = x^2 + x + 1$. E(4) is the set $\{0, 1, \alpha, \alpha + 1\}$. Addition and multiplication over GF(4) are given by the tables:

+	0	1	α	α+1
0	0	1	α	α+1
1	1	0	α+1	α
α	α	α+1	0	1
α+1	α+1	α	1	0

*	0	1	α	α+1
0	0	0	0	0
1	0	1	α	α+1
	0	α	α+1	1
+1	0	α+1	1	α

In practice the elements of E(4) would be represented as: 00, 01, 10, and 11.

The fields $GF(p^n)$ are the only finite fields. For fixed p and fixed n there are as many representations of $GF(p^n)$ as there are irreducible polynomials of degree n over GF(p). The structure of these different representations is the same, that is, each copy is isomorphic to each of the other copies.

We deal throughout this paper with a hypothetical machine. The machine is a base p machine (where p is a prime) and has word length n. Hence the space of words representable on the machine has p^n elements. Further, we hypothesize that the machine has as fundamental operations the operations of addition, subtraction, multiplication and division over $GF(p^n)$.

From an engineering point of view the choice of the irreducible polynomial used to define Galois multiplication greatly affects the complexity of the circuit design. We see later in the paper that this is purely a design decision which may be entrusted to the engineer, since it has no effect upon the quality of life of the programmer.

We call such a hypothetical machine a Galois machine. The bulk of this paper is concerned with theoretical and practical questions involved in programming for such a machine. (This assumes that the reader believes that there is a practical side to discussing a hypothetical machine.)

This paper excludes all reference to the extensive use of finite fields in coding theory.

II. Total Functions

A significant part of the use of computers is involved with the evaluation of total functions. The focal problem of circuit design is the physical realization of total functions.

The mathematization of circuit design has developed from the work of Shannon [5]. An extensive literature has grown up which addresses itself to the problem of minimizing a circuit developed in this traditional fashion.

Another approach by Reed [6] and Muller [7] uses the operations of addition and multiplication over GF(2), which are the "exclusive or" and "and" operations, as primitive operations. In his seminal paper Reed suggests that the techniques therein employed be extended to finite fields other than GF(2).

This approach is particularly attractive in the light of a sequence of unique representation theorems. Firstly, if $f: E(p^n) \to E(p^n)$ is a function then there exists a unique polynomial over $GF(p^n)$ which at each point of $E(p^n)$ defines f. Specifically,

$$f(x) = \sum_{i=0}^{p^n-1} a_i x^i, \text{ where for all } i,\ a_i \in E(p^n).$$

Analogously, if $g: E^2(p^n) \to E(p^n)$, then there exists a unique polynomial over $GF(p^n)$ which defines g:

$$g(x,y) = \sum_{i,j=0}^{p^n-1} a_{ij} x^i y^j.$$

In general the analogous result holds for a function of any dimensionality.

The above noted uniqueness is an advantage in that the normal representation is also the minimal representation. The user still needs an effective procedure to determine the coefficients of the defining polynomial. Such a procedure was developed by Newton and published in 1675, [8]. It has been less elegantly rediscovered at regular intervals since. The classical work on divided difference methods was developed for the rational and real fields, but the methods carry over directly to finite fields and gain in simplicity in the transition. We describe the method here only for functions of one variable.

Let $k = p^n$ and let $x_0, x_1, \ldots, x_{k-1}$ be some permutation of the elements of $E(k)$. Define,

$$[x_0 x_1] = \frac{f(x_0) - f(x_1)}{x_0 - x_1}, \text{ and in general,}$$

$$[x_i x_{i+1}] = \frac{f(x_i) = f(x_{i+1})}{x_i - x_{i+1}}.$$

$$[x_0 x_1 x_2] = \frac{[x_0 x_1] - [x_1 x_2]}{x_0 - x_2}, \text{ and in general,}$$

$$[x_i x_{i+1} \cdots x_j] = \frac{[x_i x_{i+1} \cdots x_{j-1}] - [x_{i+1} x_{i+2} \cdots x_j]}{x_i - x_j}.$$

Newton's Theorem states:

$$f(x) = f(x_0) + \sum_{i=1}^{k-1} [x_0 x_1 \ldots x_i] (x-x_0)(x-x_1)\ldots(x-x_{i-1}).$$

The coefficients $[x_0 x_1 \ldots x_i]$ may be most easily computed by developing a table of the form:

x_0	$f(x_0)$			
		$[x_0 x_1]$		
x_1	$f(x_1)$		$[x_0 x_1 x_2]$	
		$[x_1 x_2]$		...
x_2	$f(x_2)$		$[x_1 x_2 x_3]$	
		$[x_2 x_3]$		...
.	.		.	
		.		...
.	.		.	
		.		...
.	.			
			$[x_{k-3} x_{k-2} x_{k-1}]$	
		$[x_{k-2} x_{k-1}]$		
x_{k-1}	$f(x_{k-1})$			

The first stated unique representation theorem ensures that the polynomial which results from "multiplying out" Newton's representation is independent of which particular permutation of $E(p^n)$ is chosen for the points: $x_0, x_1, \ldots, x_{k-1}$. This method of Newton's is computationally far simplier than some methods published recently [9]. Some detailed examples of polynomial development for functions of one and two variables are in [13].

Earlier we noted that for each p and n there are several isomorphic representations of $GF(p^n)$. It is natural to have concern that the choice of representation of $GF(p^n)$ will affect the number of nonzero coefficients in the polynomials which represent a given function. The choice has no effect. To see this let us consider the following situation:

1. $f: E(p^n) \to E(p^n)$ is a function.
2. F and F' are two isomorphic copies of $GF(p^n)$.
3. ϕ is the isomorphism $\phi: F \to F'$ (and consequently ϕ^{-1} is the isomorphism $\phi^{-1}: F' \to F$).
4. $P \in F[x]$ and $Q \in F'[x]$.
5. For all $x \in E(p^n)$, $f(x) = P(x) = Q(x)$.
6. $P(x) = \sum_{i=0}^{k-1} p_i x^i$ and $Q(x) = \sum_{i=0}^{k-1} q_i x^i$.

Since ϕ and ϕ^{-1} are the respective isomorphisms, we have,

$$P(x) = \phi^{-1} Q \phi\, x$$

$$= \phi^{-1} \Sigma\, q_i (\phi\, x)^i$$

$$= \phi^{-1} \Sigma\, q_i \phi\, (x^i)$$

$$= \Sigma\, \phi^{-1} q_i \phi^{-1} \phi\, x^i$$

$$= \Sigma\, \phi^{-1} q_i x^i.$$

That is to say that for all i, $p_i = \phi^{-1} q_i$ and $q_i = \phi p_i$. Since each isomorphism maps zero onto itself, the number of non-zero coefficents is the same in P(x) and Q(x). Exactly the same argument applies to functions of several variables.

The above results have two practical consequences. If Galois addition and multiplication are used as fundamental gates for circuit design then the processes of circuit design for a total function may be automated. If Galois addition and multiplication are used as operations of a (still hypothetical) machine then the process of programming for the evaluation of total functions may be automated.

In using a polynomial representation of a function, we utilize only the operations of Galois addition and multiplication. We do not utilize the operations of subtraction and division over $GF(p^n)$.

If p is small, subtraction is not particularly helpful. Each field $GF(p^n)$ is of characteristic p. This is to say that for each $x \in E(p^n)$ the sum of p copies of x is zero: $\underbrace{x + x + \ldots + x}_{p \text{ times}} = 0$.

Specifically in each $GF(2^n)$, $-x = x$, and in each $GF(3^n)$, $-x = x + x$.

The operation of division is another matter altogether. Let $k = p^n$, and let $x_0, x_1, \ldots, x_{k-1}$ denote a permutation of the elements of E(k). Let

$$Z(x) = \prod_{i=0}^{k-1} (x - x_i).$$

This Z(x) is a polynomial of degree k which is constantly zero. There exists no polynomial of degree less than k and greater than zero which is constantly zero. Similarly, Z(x) + 1 is constantly 1.

Let $P(x)$ be either an irreducible polynomial or a product of irreducible polynomials. Suppose that the degree of $P(x)$ is less than k. Since an irreducible polynomial is nowhere zero, $1/P(x)$ is everywhere defined and nowhere zero. Thus $1/P(x)$ defines a function and that function has a unique representation as a polynomial of degree less than k, say $1/P(x) = Q(x)$. Hence we may write: $P(x)Q(x) = Z(x) + 1$, that is $P(x) = (Z(x) + 1)/Q(x)$. Such a formal division of polynomials in a finite field never produces an infinite series, as it would over the rational or real fields. Now if we let $| P(x) |$ denote the degree of $P(x)$, then we have:

$$| P(x)Q(x) | = | P(x) | + | Q(x) | = k, \text{ or } | P(x) | = k - | Q(x) |.$$

Thus, if a function is nowhere zero and is represented by a polynomial with a very large number of non-zero terms, then a much more efficient evaluation of the function results from the implementation of the inverse function. If a function has zeros, then the linear terms must be factored out and appear in the numerator of the eventual rational form.

A complete theory of the representation of functions by rational forms over finite fields has never been developed. Significant problems exist in the case of functions of several variables. Research is going on in this area.

III. Partial Functions

To say that $p: E^m(k) \to E(k)$ is a partial function is to say that there exists a subset $A \subset E^m(k)$, such that $p: A \to E^m(k)$ is a total function.

There has been a tendency among writers to identify the evaluation of partial functions with recursively defined procedures which, for some input values, do not terminate. This identification is regrettable for it clouds the deeper implications of the nontermination of an algorithm.

In complex analysis a distinction is made between two types of discontinuous behavior of functions. A function is said to have a pole at a point a if the Laurent series expansion of the function about the point a contains finitely many terms of the form: $a_i/(z - a)^i$. If the Laurent expansion contains infinitely many terms of the above form the discontinuity is said to be essential. In a finite space a partial function may be regarded simply as a function (in the sense of complex analysis) with finitely many poles. Consider the following situation.

Suppose that $f: E(p^n) \to E(p^n)$ is a function, and suppose that a partial function $p: E(p^n) \to E(p^n)$ is defined:

$$p(x) = \begin{cases} f(x), \text{ if } x \neq a; \\ \text{undefined, if } x = a. \end{cases}$$

By the unique representation theorem there is a polynomial $\chi_a(x)$ such that:

$$\chi_a(x) = \begin{cases} 0, \text{ if } x = a; \\ 1, \text{ of } x \neq a. \end{cases}$$

The partial function p(x) is defined by the rational form:

$$p(x) = f(x)/\chi_a(x).$$

If A is a set of points at which p is undefined, then

$$\chi_A(x) = \begin{cases} 0, \text{ if } x \in A; \\ 1, \text{ if } x \quad A; \end{cases}$$

is given by a polynomial and,

$$p(x) = f(x)/\chi_A(x).$$

There is an analog to the Mittag-Leffler Theorem [10, pp. 304-7] of complex analysis which says that p(x) may be represented in the form:

$$p(x) = g(x) + \sum_{a \in A} b_a/(x - a),$$

where g(x) is a polynomial and the coefficients b_a are elements of $E(p^n)$.

Although this development is brief, it is clear that rational forms over a finite field provide a convenient vehicle for the representation of partial functions.

Several other classical research directions deserve investigation in the context of function representation over finite fields; specifically the representation of functions as continued fractions [11] and the representation of functions as products rather than as series [10, pp. 291-7]. The techniques of harmonic analysis have been applied to the fields $GF(2^n)$ [12], but not to other finite fields.

In closing, consider again our Galois machine. It is not reasonable to expect that applications programmers would ever program in terms of Galois operations. However, as an underlying hardware upon which a variety of virtual machines are to be implemented, there is much to commend the Galois machine. Newton's method provides a mechanical means of code generation. If the defining polynomial has too many terms for convenience, the programmer may mechanically generate a rational form involving fewer operations. These techniques are applicable to both total and partial functions. In the light of all of this it seems worthwhile to devote attention to the areas we have noted as condidates for further research.

References

1. Zhogolev, Y. A., "The Order Code and Interpretive System for the SETUN Computer", USSR Comp. and Math. Physics I, no. 3 (1962) pp. 563-78.

2. G. Frieder, A. Fong, and C. Y. Chao, "A Balanced Ternary Computer", Conference Record of the 1973 International Symposium on Multiple-valued Logic (Toronto, Canada) May 1973, pp. 68-88.

3. H. T. Mouftah and I. B. Jordan, "A Design Technique for an Integrable Ternary Arithmetic Unit", Proc. 1975 Int. Symp. on Multiple-valued Logic, (Bloomington, Indiana, May 1975), pp. 359-372.

4. D. Etiemble and M. Israel, "Implementation of a Complete Ternary Algebra - Application to Ternary Flip Flop", Proc. 1975 Int. Symp. on Multiple-valued Logic, (Bloomington, Indiana, May 1975), pp. 316-329.

5. Claude E. Shannon, " A Symbolic Analysis of Relay and Switching Circuits", Trans. Am. Inst. Elec. Eng. 57 (1938), pp. 713-23.

6. Irving S. Reed, "A Class of Multiple-error-correcting codes and the Decoding Scheme", Trans. IRE - Info. Theory PGIT -4 (September 1954), pp. 38-49.

7. D. E. Muller, "Application of Boolean Algebra to Switching Circuit Design and to Error Correction", IRE Trans. - Elec. Comp. EC-3, No. 3 (September 1954), pp. 6-12.

8. Isaac Newton, "Approaches to a General Theory of Finite Differences" [1675-6] in D. T. Whiteside (editor), The Mathematical Papers of Isaac Newton, vol. 4, (Cambridge, The University Press, 1971), pp. 14-73.

9. Boonsieng Benjauthrit and Irving S. Reed, "Galois Switching Functions and Their Applications" IEEETC C-25, no. 1 (January 1976) pp. 78-86.

10. Richard A. Silverman, Introductory Complex Analysis, (Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1967)

11. D. C. Olds, Continued Fractions, (New York: Random House, 1963)

12. Robert J. Lechner, "Harmonic Analysis of Switching Functions", Recent Developments in Switching Theory (A. Mukhopadhyay, editor), (New York: Academic Press, 1971), pp. 121-227.

13. T. C. Wesselkamper, "Some Classical Mathematical Results Related to the Problems of the Firmware/Hardware Interface" SIGMICRO Newsletter 6, no. 4, (December 1975), pp. 32-38.