J. Symbolic Computation (1997) **24**, 351–369

Lattices of Compatibly Embedded Finite Fields[†]

WIEB BOSMA[‡], JOHN CANNON[§] AND ALLAN STEEL[¶]

Computational Algebra Group, School of Mathematics and Statistics, The University of Sydney, NSW 2006, Australia

The design of a computational facility for finite fields that allows complete freedom in the manner in which fields are constructed, is complicated by the fact that a field of fixed isomorphism type K may be constructed in many different ways. It is desirable that the user be able to perform simultaneous computations in different versions of K in such a way that isomorphisms identifying elements in the different versions are applied automatically whenever necessary. This paper presents a coherent scheme for solving this problem based on an efficient method for compatibly embedding one field within another. This scheme forms a central component of the MAGMA module for finite fields. The paper also outlines the different representations of finite fields employed in the package and comments briefly on some of the major algorithms.

1. Introduction

Finite fields play a crucial role in computational algebra even though, from both a mathematical and computational point of view, they are rather simple structures. The finite field module for the computer algebra system MAGMA (Bosma *et al.*, 1994; Cannon and Playoust, 1996; Bosma *et al.*, 1997) is designed to provide highly efficient arithmetic and structural computation across the range of finite fields that occur in practical applications. This paper identifies some of the key problems that must be addressed when developing such a facility and presents the solutions adopted in the MAGMA system.

A fundamental issue concerns the choice of data structures used to represent finite fields and their elements. This raises mathematical issues (representation of fields and their elements), algorithmic issues (the choice of data structure may have a major effect on the performance of an algorithm) and software engineering issues (fast access, memory management, etc.). For example, in the case of fields of moderate cardinality a choice has to be made between representing field elements as polynomials over a ground field or representing them as powers of a primitive element.

A related issue is the design of efficient algorithms for key operations. For finite fields these include very fast arithmetic, the construction of elements with special properties

[†] This work was supported by the Australian Research Council.
[‡] E-mail: wieb@maths.usyd.edu.au
[§] E-mail: john@maths.usyd.edu.au
[¶] E-mail: allan@maths.usyd.edu.au

0747–7171/97/030351 + 19 $25.00/0 sy970138

(normal, primitive, etc.), the computation of discrete logarithms and the factorization of polynomials. Where standard algorithms have been implemented we will usually refer to original papers for details and omit descriptions of any optimizations we have introduced.

A module in a computer algebra system (CAS) differs from a stand-alone package in that a well-designed CAS allows the user to have many different structures and relationships defined simultaneously. MAGMA, in particular, has been designed to allow a user to work in many (related) finite fields simultaneously, to create arbitrary extensions and subfields, to move between them and to use finite fields as building blocks in the construction of more complicated algebraic structures. A major complication arises from the fact that a given isomorphism type of finite field may be realized in many different ways. We require a scheme that allows computation to proceed simultaneously in two different versions of the one field in such a way that an isomorphism is created and applied automatically as required.

MAGMA provides an internal mechanism which allows a user to work with distinct lattices of finite fields of different characteristics at the same time. Beginning with a single field, such a lattice of fields is constructed by repeatedly applying a sequence of operations of the following types:

- Explicitly embedding one field in a second field so as to ensure compatibility (provided that the embedding is allowed) (Subsection 5.1);
- Creating an extension field of a current field (Subsection 5.2);
- Creating a subfield field of a current field (Subsection 5.3).

Complete freedom is given as to the choice of irreducible polynomials for extensions and generating elements for subfields. The irreducible polynomial used to define a field is not required to be primitive.

The paper is organized as follows. Section 2 deals with the critical issue of compatibility. An axiomatic characterization of lattices of compatibly-embedded finite fields is used to present an effective method of embedding fields. Section 3 describes the representation of individual finite fields in MAGMA (leaving aside the issue of compatibility). Of particular note are the *optimized* representations. Section 4 describes how subfield lattice information is represented in MAGMA (the information associated with subfield relationships). Section 5 describes the implementation of operations which change a lattice, showing how the lattice properties are preserved under each operation. Section 6 contains some brief comments on some of the non-trivial algorithms for field elements implemented in MAGMA. Finally, section 7 presents an extended example of a MAGMA session which demonstrates many of the features discussed in the paper.

We do not attempt to give a complete description of the finite field facilities in MAGMA or provide details of the relevant functions—we refer the reader instead to the MAGMA Handbook (Bosma and Cannon, 1996).

The reader is assumed to be familiar with the basic theory of finite fields. An account of the relevant theory may be found in works such as Jungnickel (1993), Lidl and Niederreiter (1986). The following conventions are used throughout the paper. The symbol p denotes a rational prime, q denotes a power of p (which may equal p), and $\mathbf{F}_q$ denotes a finite field of q elements. The symbol $\partial(F)$ denotes the absolute degree (over the prime field) of a field F and $\partial(F/E)$ denotes the relative degree of F over E. The degree of F refers to its absolute degree. If α is an element of a field F and S is a subfield of F, $\mathrm{Min}_S(\alpha)$ denotes the minimal polynomial of α over S.

2. Compatibility

2.1. BACKGROUND

Let E be a finite field of cardinality p^e, and F a field of cardinality p^f. If e divides f then E 'is contained in' F in the sense that F contains a subfield of cardinality p^e which is isomorphic to E. Since this isomorphism is, in general, not unique, an *explicit embedding of E in F* will refer to the choice of one particular (explicit) isomorphism. Suppose that the field S has been explicitly embedded in both E and F. It is desirable to choose the explicit embedding of E in F in such a way that the resulting diagram commutes: if an element of S is expressed as an element of F, it should not matter whether this is done directly or via E. Informally, a lattice of finite field extensions is termed *compatible* if all diagrams representing explicit embeddings commute.

We present an efficient solution to the problem of maintaining the compatibility of a lattice of finite field extensions as new fields are added.

2.2. EXPLICIT EMBEDDINGS AND COMPATIBILITY

Let L be a set of finite fields of characteristic p and let Φ be a collection of field monomorphisms $\phi_{E\hookrightarrow F}$ between certain elements E and F of L. The field E is said to be *explicitly embedded* in F if there exists $\phi_{E\hookrightarrow F} \in \Phi$. If S is explicitly embedded in both E and F, with corresponding embeddings $\phi_{S\hookrightarrow E}$ and $\phi_{S\hookrightarrow F}$, an explicit embedding $\phi_{E\hookrightarrow F}$ is said to be a *compatible embedding of E in F with respect to S* if $\phi_{S\hookrightarrow F} = \phi_{E\hookrightarrow F} \circ \phi_{S\hookrightarrow E}$. The field E is said to be *compatibly embedded* in F if the explicit embedding $\phi_{E\hookrightarrow F}$ is a compatible embedding with respect to all common explicitly embedded subfields.

We denote by $\preceq$ the relation of being explicitly embedded, so that $E \preceq F$ if and only if there exists $\phi_{E\hookrightarrow F} \in \Phi$. This equips L with a preordering (i.e., a relation that is transitive and reflexive but not necessarily anti-symmetric).

THEOREM 2.1. *Let E, F and S be fields such that $S \preceq E$ and $S \preceq F$ and suppose further that $\partial(E)$ divides $\partial(F)$. Then there exist $\partial(E)/\partial(S)$ distinct compatible embeddings of E in F.*

PROOF. A single compatible embedding ϕ may be constructed as follows. An arbitrarily chosen embedding, $\psi = \psi_{E\hookrightarrow F}$, is made compatible with existing embeddings of S into E and F by composing ψ with an element $\sigma \in \mathrm{Gal}(F)$ having the property that $\sigma(\psi(S)) = \phi_{S\hookrightarrow F}(S)$. Hence, $\phi = \phi_{E\hookrightarrow F} = \sigma \circ \psi$. Any compatible embedding may now be obtained from ϕ by composing it with an automorphism of $\phi(E)$ that leaves $\phi_{S\hookrightarrow F}(S)$ invariant. This yields $\#\,\mathrm{Gal}(\phi(E)/\phi_{S\hookrightarrow F}(S)) = \#\,\mathrm{Gal}(E/S) = \partial(E)/\partial(S)$ possible embeddings. □

The next proposition is a constructive version of the theorem.

PROPOSITION 2.2. *Let E, F and S be fields such that $S \preceq E$, $S \preceq F$ and $\partial(E)$ divides $\partial(F)$. Suppose that S is identified with its isomorphic image $\phi_{S\hookrightarrow E}(S)$. Suppose further that α generates E over S, and let $f_S = \mathrm{Min}_S(\alpha)$. Let $\tau \in F$ be a root of $\phi_{S\hookrightarrow F}(f_S)$. Define $\phi\colon E \to F$ by*

$$\phi\Bigg(\sum_{i=0}^{\partial(E/S)-1} s_i\alpha^i\Bigg) = \sum_{i=0}^{\partial(E/S)-1} \phi_{S\hookrightarrow F}(s_i)\tau^i,$$

using the unique representation $\sum s_i\alpha^i$ *(with* $s_i \in S$*) for an arbitrary element of* E.
Then ϕ *is a field monomorphism from* E *to* F *such that* $\phi \circ \phi_{S\hookrightarrow E} = \phi_{S\hookrightarrow F}$.

PROOF. The existence of τ is clear, since f_S has a root in E and hence also in the subfield of F isomorphic to E. The uniqueness of the representation of elements of E over S implies that the map ϕ is well-defined. Observing that α and τ have the same minimal polynomial over S, it is straightforward to show that ϕ is a homomorphism. Similarly, it can be shown that ϕ is one-to-one. The compatibility condition holds by construction. □

REMARK 2.3. We retain the notation of Proposition 2.2. If α generates E over the prime field P, then α also generates E over $\phi_{S\hookrightarrow E}(S)$. A root τ of $\mathrm{Min}_S(\alpha)$ is, *a fortiori*, a root of $\phi_{P\hookrightarrow F}(f_P)$, where f_P is the minimal polynomial of α over the prime field P. The map ϕ of Proposition 2.2 may be constructed with respect to P rather than S but using the root τ of $\phi_{S\hookrightarrow F}(\mathrm{Min}_S(\alpha))$. The resulting map ϕ will be the same, since both $\phi_{P\hookrightarrow E}$ and $\phi_{P\hookrightarrow F}$ factor over $\phi_{P\hookrightarrow S}$. Thus, the map ϕ may be implemented using the representation of elements of E that arises from regarding E as an algebra over P while still maintaining compatibility with respect to S (this is used in Subsection 4.1 to efficiently represent an embedding map in MAGMA).

2.3. COMPATIBLE POLYNOMIALS

In theory, it is possible to solve the compatibility problem by working solely with so-called compatible polynomials, as introduced by Conway and Parker (Scheerhorn, 1992). A collection of polynomials $f_d \in \mathbf{F}_p[x]$, with degree set D, is said to be *(norm-) compatible* if (a), each polynomial is primitive (in the sense that any root β_d of f_d is a primitive element for $\mathbf{F}_p[x]/f_d(x)$), and (b), for every pair $d_1, d_2 \in D$ such that $d_1|d_2$, it is the case that $\mathbb{N}_{\mathbf{F}_{p^{d_2}}/\mathbf{F}_{p^{d_1}}}(\beta_{d_2}) = \beta_{d_1}$ is a root of f_{d_1}. A compatible embedding is obtained by mapping β_{d_1} to $\beta_{d_2}^k$, where $k = (p^{d_2}-1)/(p^{d_1}-1)$. In this manner compatibility in lattices of fields is ensured by restricting the choice of polynomial used to define a field. The main drawback with this scheme is that, in general, the set of finite fields needed in a computation is not known beforehand, and it is usually difficult to extend a compatible family of polynomials to larger fields. Therefore, in practice this strategy is useful only when working in fairly small fields.

In moderate-sized extensions, compatible collections of polynomials provide a means of constructing *standardized* finite field extensions. By including the additional condition that the polynomials be chosen minimal with respect to some ordering, uniqueness can be achieved. This leads to the following definition (Jansen *et al.*, 1995). The *Conway polynomial* of degree d (over $\mathbf{F}_p$) is the first monic primitive polynomial of degree d that is compatible with all Conway polynomials of smaller degree, where the ordering of monic polynomials of degree d is defined as follows:

$$x^d + (-1)\cdot a_{d-1}\cdot x^{d-1} + \cdots + (-1)^{d-1}\cdot a_1\cdot x + (-1)^d\cdot a_0$$

is less than

$$x^d + (-1)\cdot b_{d-1}\cdot x^{d-1} + \cdots + (-1)^{d-1}\cdot b_1\cdot x + (-1)^d\cdot b_0$$

if and only if

$$(a_{d-1}, a_{d-2}, \ldots, a_1, a_0) < (b_{d-1}, b_{d-2}, \ldots, b_1, b_0)$$

in the lexicographical ordering of tuples of integers, where the a_i, b_j are taken to be representatives in $\mathbb{Z}$ for the appropriate residue classes modulo p satisfying $0 \leq a_i, b_j < p$. It is not obvious from this definition that Conway polynomials exist for every degree and characteristic.

Compatibility in Magma is achieved using a different and more general approach, which is explained in detail in the following sections. The use of arbitrary irreducible polynomials for the representation of finite fields is allowed, but the explicit embeddings will always be made in a compatible way. This is achieved by ensuring that each new explicit embedding is created in such a way that it is compatible with all existing embeddings. The implementation involves only elementary linear algebra and the computation of roots of polynomials.

2.4. Lattices of compatibly embedded finite fields

In this subsection a formal model for a lattice of finite fields is presented. A lattice in this model will consist of a collection of finite fields all having the same characteristic, with explicit embeddings between certain members of the collection.

The pair $\mathcal{L} = (L, \Phi)$ consisting of a collection of finite fields of the same characteristic and a collection of explicit embeddings between members of L is called a *lattice of compatibly embedded finite fields* if the following conditions are satisfied:

CE1 [Unicity] For each ordered pair (E, F) of elements of L, there exists at most one $\phi_{E \hookrightarrow F} \in \Phi$.
CE2 [Reflexivity] For each $E \in L$ the identity $\mathrm{id}_E = \phi_{E \hookrightarrow E}$ is in Φ.
CE3 [Prime subfield] There is exactly one $P \in L$ with $\partial(P) = 1$, and for all $F \in L$ there exists $\phi_{P \hookrightarrow F} \in \Phi$.
CE4 [Invertibility] If $E \preceq F$ and $\partial(E) = \partial(F)$, then $F \preceq E$ and $\phi_{F \hookrightarrow E} = {\phi_{E \hookrightarrow F}}^{-1}$.
CE5 [Transitivity] For any triple (E, F, G) of elements of L, if $E \preceq F$ and $F \preceq G$ then $E \preceq G$ and $\phi_{E \hookrightarrow G} = \phi_{F \hookrightarrow G} \circ \phi_{E \hookrightarrow F}$.
CE6 [Intersection] For each E, F, G in L such that $E \preceq G$ and $F \preceq G$, there exists $S \in L$ such that $\partial(S) = \gcd(\partial(E), \partial(F))$, $S \preceq E$ and $S \preceq F$.

It will be helpful to comment on the conditions CE1–CE6.

It is important to note that if the lattice $\mathcal{L}$ contains fields E and F such that $\partial(E)$ divides $\partial(F)$, then although E is isomorphic to a subfield of F an explicit embedding $\phi_{E \hookrightarrow F}$ may not yet exist. Further, while there are $\partial(E)$ ways of embedding E in F (by Theorem 2.1 with S taken to be the prime field P), only some will be compatible with other embeddings in the lattice. Consequently, E is not regarded as a subfield of F until it is explicitly embedded. This is reflected in CE1.

For convenience, CE2 insists that all identity maps be included so that, in combination with CE1, non-trivial automorphisms are excluded.

Condition CE3 expresses the manner in which the prime field is identified in a finite field.

Condition CE4 expresses the requirement that if one field is explicitly embedded in another and the fields are isomorphic then they are related via embeddings that must compose to the identity map. Note that this is consistent with CE2 and CE3. The fact that $\preceq$ is not required to be anti-symmetric allows for the possibility of having several fields of the same cardinality in L. This is useful in various situations. For example, it

allows the lattice to contain images of a field under Galois automorphisms of a larger field.

Condition CE5 ensures transitivity and compatibility in triangular diagrams.

Finally, condition CE6 is a way of ensuring that implicit compatibility conditions are made explicit. As an illustration, consider the embedding of the field $\mathbf{F}_{p^4}$ of degree 4 and the field $\mathbf{F}_{p^6}$ of degree 6 into a field of degree 24. Each of $\mathbf{F}_{p^4}$ and $\mathbf{F}_{p^6}$ contains a quadratic subfield that has been implicitly embedded and there is now an implicit isomorphism between the two quadratic subfields. This will impose compatibility conditions on future embeddings. If $\mathbf{F}_{p^4}$ and $\mathbf{F}_{p^6}$ are now embedded in a field $\mathbf{F}_{p^{60}}$ of degree 60, the choice of embeddings must respect the isomorphism of the quadratic subfields. Condition CE6 makes this explicit by ensuring that there is a field S of degree 2 with isomorphisms from S to each of the quadratic subfields of $\mathbf{F}_{p^4}$ and $\mathbf{F}_{p^6}$, that, by transitivity, will also be embedded in the field of degree 24, and later in $\mathbf{F}_{p^{60}}$.

Note that if $E, F, G \in L$ are such that $E \preceq G$, $F \preceq G$ and $\partial(E) \mid \partial(F)$, it follows by CE6 that there is a field $S \in L$ with $\partial(S) = \partial(E)$ such that $S \preceq E$ and $S \preceq F$. It then follows from CE4 that $E \preceq S$ and from CE5 that $E \preceq F$. This shows that embeddings into larger fields determine the embeddings into their subfields.

In the model, a lattice may be modified by:

- the creation of a new explicit (compatible) embedding of one field into another;
- the creation of a new field.

The manner in which these operations are implemented in Magma will be described in Section 5. In the remainder of this section, formal aspects of the lattice model will be developed.

In the sequel, unless otherwise stated, to say that E is a subfield of F will be taken to mean that E is an embedded subfield of F, so that $E \preceq F$.

2.5. adding a compatible embedding

Let L be a collection of finite fields of fixed characteristic p and let Φ be a collection of explicit embeddings between members of L. The set Φ is said to form a *compatible collection* for L if for every E, F, S in L such that $S \preceq E$, $S \preceq F$, and $E \preceq F$, E is compatibly embedded in F.

Suppose that E and F in L are such that $E \not\preceq F$. Let $C := \{S \in L \mid S \preceq E \text{ and } S \preceq F\}$ and let E' be the subfield of E generated by $\{\phi_{S \hookrightarrow E}(S) : S \in C\}$. Then E' may be considered as a new field in the lattice $\mathcal{L}$ such that E' is an embedded subfield of E (i.e., $E' \preceq E$) in the following way: the embedding $\phi_{E' \hookrightarrow E}$ is the identity map and for each subfield $S \in C$ the embedding $\phi_{S \hookrightarrow E'}$ is equal to the embedding $\phi_{S \hookrightarrow E}$ with restricted codomain E'. Condition CE6 on the intersection of subfields is automatically fulfilled, since $\partial(E') = \operatorname{lcm}\{\partial(S) \mid S \in C\}$. Analogously, let F' be the subfield of F generated by $\{\phi_{S \hookrightarrow F}(S) : S \in C\}$ and suppose that the respective associated embedding maps have been constructed for F'.

The key observation is that a fully compatible embedding ϕ of E into F is uniquely determined when restricted to the domain E'.

Theorem 2.4. *There exists a unique field isomorphism $\chi : E' \to F'$ which is compatible*

with the set C of common embedded subfields of E' and F', such that $\chi \circ \phi_{S \hookrightarrow E'} = \phi_{S \hookrightarrow F'}$ for each $S \in C$.

PROOF. It is clear that there can be at most one isomorphism $\chi: E' \to F'$, because every element of E' can be expressed in terms of images of elements in the subfields S. By construction, E' and F' have the same degree (the least common multiple of the degrees of the fields $S \in C$), hence there exists an isomorphism χ_0 of E' into F'. Let P be the prime field in C, and let $S' = \phi_{S \hookrightarrow F'}(S)$ for any subfield $S \in C$.

As shown in the proof of Theorem 2.1, the isomorphisms of E' and F' compatible with a subfield S correspond to a coset of $\mathrm{Gal}(F'/S')$ in $\mathrm{Gal}(F'/P')$. The intersection of two such cosets $\sigma_1(\mathrm{Gal}(F'/S_1'))$ and $\sigma_2(\mathrm{Gal}(F'/S_2'))$ is a coset $\sigma(\mathrm{Gal}(F'/S_1') \cap \mathrm{Gal}(F'/S_2'))$, since the embeddings $\phi_{S_1 \hookrightarrow F'}$ and $\phi_{S_2 \hookrightarrow F'}$ are compatible on a subfield of degree $\gcd(\partial(S_1), \partial(S_2))$ by CE6. By induction, there exists an automorphism $\sigma \in \mathrm{Gal}(F'/P')$ such that $\chi = \sigma \circ \chi_0$ is compatible with all subfields $S \in C$. Further, since $\bigcap_{S \in C} \mathrm{Gal}(F'/S') = \{\mathrm{id}\}$, σ (and hence χ) are unique. □

COROLLARY 2.5. *There exists a field monomorphism $\phi: E \hookrightarrow F$ with the property that $\Phi \cup \{\phi\}$ forms a compatible collection for L. In particular, if $\mathcal{L} = (L, \Phi)$ forms a lattice of compatibly embedded fields then there exists a lattice $\mathcal{L}' = (L', \Phi')$ containing $\mathcal{L}$ (in the sense that $L' \supset L$ and $\Phi' \supset \Phi$) such that an embedding of E into F is contained in Φ'.*

PROOF. Let E' and F' be as above. Using the unique isomorphism $\chi: E' \to F'$ constructed in Theorem 2.4, embed E' in F' so that $E' \preceq F$ via $\phi_{F' \hookrightarrow F} \circ \chi$. Applying Remark 2.3 to E and F with S taken to be E' and with α taken to be a generator of E over P yields an embedding ϕ of E in F which is compatible with all existing embeddings for subfields $S \in C$, since $\phi_{|E'} = \chi$. In fact, by Theorem 2.1 there are $\partial(E)/\partial(E')$ different such field monomorphisms. □

The implementation in MAGMA avoids the explicit intersection of cosets in the Galois group of F' through use of a method suggested by the following proposition.

PROPOSITION 2.6. *Let E, E' and C be as above. Suppose that $\alpha \in E$ generates E over P. Then*

$$\mathrm{Min}_{E'}(\alpha) = \gcd\{\mathrm{Min}_{\phi_{S \hookrightarrow E'}(S)}(\alpha) \mid S \in C\}.$$

PROOF. The minimal polynomial of α over P splits over E as

$$\mathrm{Min}_P(\alpha) = \prod_{\sigma \in \mathrm{Gal}(E)} (x - \alpha^\sigma).$$

Hence

$$\mathrm{Min}_{E'}(\alpha) = \prod_{\sigma \in \mathrm{Gal}(E/E')} (x - \alpha^\sigma)$$

and analogously

$$\mathrm{Min}_{\phi_{S \hookrightarrow E'}(S)}(\alpha) = \prod_{\sigma \in \mathrm{Gal}(E/\phi_{S \hookrightarrow E'}(S))} (x - \alpha^\sigma).$$

Since the embeddings $\phi_{S \hookrightarrow E'}$ coincide on intersections after suitable identifications,

$\mathrm{Min}_{E'}(\alpha)$ is the greatest common divisor of the minimal polynomials over the embeddings $\phi_{S\hookrightarrow E'}(S)$. □

Since the unique isomorphism χ identifies E' with F', we have

$$\mathrm{Min}_{\phi_{S\hookrightarrow E'(S)}}(\alpha) = \mathrm{Min}_{\phi_{S\hookrightarrow F'(S)}}(\alpha) = \mathrm{Min}_{\phi_{S\hookrightarrow F(S)}}(\alpha),$$

so that the greatest common divisor need only be computed in $F[x]$. The compatible embedding of E in F may now be created as follows: if

$$f(x) = \gcd\{\phi_{S\hookrightarrow F}(\mathrm{Min}_{\phi_{S\hookrightarrow E'(S)}}(\alpha)) \mid S \in C\}.$$

and τ is any root of $f(x)$ in F, then the required monomorphism $\phi\colon E \to F$ is defined by $\alpha \mapsto \tau$. The correctness of this procedure follows from Remark 2.3, Theorem 2.4 and Proposition 2.6.

Note that in order to preserve the compatibility of a lattice after adding an explicit embedding it may be necessary to add further fields to L so as to ensure that CE6 is satisfied. It will also be necessary to take the transitive closure.

2.6. adding fields to a lattice

A compatibly embedded lattice $\mathcal{L} = (L, \Phi)$ may be enlarged either by adding embeddings to Φ (as in the previous subsection) or by adding fields to L.

Suppose that F is a finite field having the same characteristic as the fields in L, but F is not yet included in L. After modifying L to include F and Φ to include the embedding $\phi_{P\hookrightarrow F}$ of the prime field P into F, it is easy to see that each of the conditions CE1–CE6 is satisfied. The field F may now be embedded in other fields of L or *vice versa*, using the method of the previous subsection. A lattice of fields may be constructed by applying this process iteratively. The process always produces a compatible lattice.

3. Representations of Finite Fields

In this section we describe how individual finite fields are represented in Magma. Note that discussion of the representation of embeddings is deferred until a later section.

3.1. summary of representations

The different internal representations of finite field elements are summarized below. A more detailed discussion of each representation appears in the following subsections.

- The SmallPrime representation applies to prime fields where the prime p fits within a machine 'short'. Elements are represented as integers taken modulo p and arithmetic is performed by table lookup for very small primes.
- The MediumPrime representation applies to prime fields where the prime p fits within a machine 'int'. Elements are represented as integers taken modulo p.
- The BigPrime representation applies to prime fields where the prime p is larger than a machine 'int'. Elements are represented as general multiple-precision integers taken modulo p.
- The PrimePoly representation applies to extensions of prime fields given in the SmallPrime representation. Elements are represented as specialized polynomials over a SmallPrime field. Polynomial arithmetic is optimized.

- The ZECH representation applies to extension fields having cardinality at most 2^{20}. Elements are represented as powers of a primitive element and a table of Zech logarithms is stored for use in addition.
- The ZECHPOLY representation applies to extension fields having cardinality greater than 2^{20}, where the field may be viewed as an extension of a ZECH field. Elements are represented as specialized polynomials over a ZECH field. The polynomial arithmetic is optimized for each particular Zech logarithm table.
- Finally, the GENERALPOLY representation applies to general extensions. Elements are represented as general polynomials defined recursively over any finite field (including fields given in this representation). The highly optimized generic univariate polynomial code of MAGMA is used. The GENERALPOLY case is required internally to create extensions temporarily during the construction of an optimized representation. This representation is usually not employed for finite fields seen by the user.

3.2. PRIME FIELDS

Computation with finite fields $\mathbf{F}_p$ of prime cardinality p is particularly easy, since the elements of $\mathbf{F}_p$ may be identified with the integers modulo p: $\mathbf{F}_p \cong \mathbb{Z}/p\mathbb{Z}$. Arithmetic is simply modular arithmetic.

MAGMA employs three different representations for prime fields: the SMALLPRIME representation where p can be stored in a 'short' ($p < 2^{16}$ for most 32-bit machines); the MEDIUMPRIME representation where p does not fit in a short but can be stored in an 'int' ($p < 2^{32}$ for most 32-bit machines); and the BIGPRIME representation for arbitrarily large p. In the BIGPRIME case, the Montgomery modular representation (Montgomery, 1985) is employed to achieve faster multiplication in the case of large primes. The case where the prime p requires exactly two multiprecision digits (i.e., p is less than 2^{64} for 32-bit machines or less than 2^{128} for 64-bit machines) is also optimized.

3.3. THE ZECH LOGARITHM REPRESENTATION

The fact that the multiplicative group $(\mathbf{F}_q)^*$ of any finite field $\mathbf{F}_q$ is cyclic of order $q-1$ may be exploited to replace multiplication of non-zero field elements by a simple modular addition of exponents with respect to a primitive element g:

$$g^k \cdot g^l = g^{k+l} \tag{1}$$

where the exponents are taken modulo $q-1$. Similarly, exponentiation reduces to multiplication modulo $q-1$. In order to perform addition and subtraction efficiently in this representation, it is necessary to store the logarithm s_r of the successor of g^r for each $0 \le r \le q-2$; that is, the integer s_r with $0 \le s_r \le q-2$ such that $g^{s_r} = g^r + 1$ (Conway, 1968). The addition of elements g^u and g^v is accomplished by looking up $s = s_{v-u}$, since $g^v + g^u = g^u(g^{v-u}+1) = g^u \cdot g^s = g^{u+s}$. The problem of zero is handled by introducing the auxiliary logarithm $q-1$. An entry with $r = q-1$ and $s_r = 0$ is thus added to cater for the successor of 0. Also, if q is odd, an entry with $r = (q-1)/2$ and $s_r = q-1$ indicates that 0 is the successor of -1.

In MAGMA, the ZECH representation is used only when the cardinality q is not prime and $q \le 2^{20}$. However, as explained in the next subsection, a ZECH representation of a subfield may form part of an optimized two-step representation for a field having much larger cardinality.

3.4. optimized representations for field extensions

In this section we describe optimized representations for an arbitrary finite field F of cardinality p^n, for a given prime p and integer $n > 1$.

If p^n is sufficiently small, F is constructed using the Zech representation. As the Zech logarithms for F are computed, a mapping is simultaneously constructed giving the one-to-one correspondence between elements of the field F (given as powers of a primitive element) and the vector space $P^{(n)}$. The embedding of P in F is determined using this map.

Next, suppose that the cardinality of F is outside the range for the Zech representation, but p^n can be written as q^k where $q = p^d$ is the largest possible proper power of p in the Zech range. (Additionally, it is necessary to assume that F is not too large, depending on the machine and characteristic p.) Then a field S of cardinality q is created in terms of the Zech representation, and F is constructed as a degree-k extension of S in terms of the ZechPoly representation. The prime field P is embedded in F by firstly embedding P into S and then using constant polynomials in F over S. An isomorphism between F and $P^{(n)}$ is constructed using the isomorphism between S and $P^{(d)}$ and the fact that elements of F are polynomials over S. This representation is known as the *two-step optimized representation*.

Otherwise, the PrimePoly representation is used to construct F as a degree-n extension of P. The prime field P is embedded in F using constant polynomials. An isomorphism between F and $P^{(n)}$ is found using the fact that the elements of F are simply polynomials over P. This method is used for fields such as $\mathbf{F}_{2^{103}}$, where the degree is a prime so the two-step method cannot be used.

The construction of an optimized representation may be quite expensive for large fields since polynomial factorization is required in order to compute appropriate extension polynomials for the embeddings (see Subsection 5.2). It is possible to suppress the use of an optimized representation in which case the construction of the field will be fast but arithmetic may be much slower. This is useful when it is necessary to perform just a few elementary calculations in an extension field before discarding it.

The use of optimal normal bases will be available soon in Magma.

The following two examples illustrate the use of the various optimized representations.

- Consider what happens if the user first creates $E = \mathbf{F}_{2^8}$ and then creates a degree-14 extension F of E, so $\#F = 2^{16\times 7}$. Magma will create E in the Zech representation, and then create a field $S = \mathbf{F}_{2^{16}}$, again in the Zech representation. Next, F is created as a degree-7 extension of S (using the ZechPoly representation) and finally the embedding machinery is applied so that E appears as a subfield of F (see below for details). Thus, there will be three fields: E (Zech 2^8), S (internal Zech 2^{16}), and F (ZechPoly of degree 7 over S). The Magma statements creating E and F are:

```
> E := FiniteField(2, 8);
> F := ext<E | 14>;
```

- As a more interesting example, consider what happens if the user creates $E = \mathbf{F}_{2^{53}}$ and then builds a degree-10 extension F of E, so that $\#F = 2^{53\times 10}$. Magma will create E in the PrimePoly representation since E is too large for the Zech

representation and the primality of 53 rules out use of the two-step representation. Next, a field $S = \mathbf{F}_{2^{10}}$ is created in the ZECH representation and then F is created as a degree-53 extension of S in the two-step representation. Finally, E is embedded in F. Consequently, although F will appear as a degree-10 extension of E with its elements being printed as polynomials over E of degree less than 10, its internal arithmetic is performed in terms of polynomials over S of degree less than 53. The field is created in this form since the use of polynomials of degree less than 53 over a ZECH field of degree 10 provides *much* faster arithmetic than using polynomials of degree less than 10 over a field represented in terms of polynomials of degree less than 53 taken over the prime field!

4. Representation of Lattice Information

Let $\mathcal{L} = (L, \Phi)$ be a compatibly embedded lattice in MAGMA and suppose that E and F are fields of L such that E is an embedded subfield of F. Throughout this section let P denote the prime field of F, e the degree of E, f the degree of F, and $d = f/e$ the relative degree of F over E.

There are four items associated with the embedding relationship $E \preceq F$ that are stored by MAGMA:

(1) The embedding map $\phi_{E\hookrightarrow F}$.
(2) A generator $\alpha_{F/E}$ for F, considered as an algebra over $\phi_{E\hookrightarrow F}(E)$, called the *generator of F over E*. The generator $\alpha_{F/P}$ of E over the prime field P is called the *prime-field generator* of F and is denoted by α_F.
(3) A vector space isomorphism $\psi_{E^{(d)}\leftrightarrow F}\colon E^{(d)} \to F$. Let ψ_F denote $\psi_{P^{(f)}\leftrightarrow F}$.
(4) The *defining polynomial* $f_{F/E}$ of F over E which is defined to be the minimal polynomial of $\alpha_{F/E}$ over E.

In this section, the calculation and representation of these items will be outlined.

Note that for each finite field F there is a fixed embedded subfield G of F called the *ground field* of F. The field F is created as an extension of G, and F is presented by default as an algebra over G: elements of F are thought of as polynomials in $G[x]$ where x corresponds to the generator $\alpha_{F/G}$ of F over G. Often, of course, the ground field is the prime field P.

4.1. REPRESENTATION OF AN EMBEDDING MAP

Subsection 2.5 gives a method for computing a compatible embedding map $\phi_{E\hookrightarrow F}$. Using Remark 2.3, the construction of the embedding map involves only linear algebra over the *prime* field P. This is of considerable importance since it may be necessary to embed one field in another field where the internal representations of the two fields are quite different. Otherwise, it is difficult to implement a general scheme capable of transferring elements between different representations.

Remark 2.3 thus provides an explicit representation of the embedding $\phi_{E\hookrightarrow F}\colon E \to F$. It is stored as an $e \times f$ matrix M with entries in P, where the ith row of M (numbered from 0 to $e-1$) gives $\phi_{E\hookrightarrow F}(\alpha_E^i)$ as an P-linear combination of the f independent powers

of α_F. That is,

$$\phi_{E\hookrightarrow F}(\alpha_E^i) = \sum_{j=0}^{f-1} M_{i,j}\alpha_F^j.$$

Once the embedding map $\phi_{E\hookrightarrow F}$ has been created and stored, computation with elements of F relative to E is possible and so can be used in the construction of other information.

4.2. relative generator and vector space isomorphism

The next step involves finding a relative generator $\alpha_{F/E}$ for F over E. Let G be the current ground field of F (which will already be compatibly embedded in F). If $\alpha_{F/G}$ generates F over E, then $\alpha_{F/E}$ is taken to be $\alpha_{F/G}$. This is the most common situation, occurring, for example, when the ground field G is the prime field, or, more generally, when G is contained in E. If $\alpha_{F/G}$ does not generate F over E, random elements $\alpha \in F\backslash E$ are chosen until a generator is found. Note that $\alpha \in F$ generates F over E if and only if $\{\alpha^i\}_{0\le i\le d-1}$ is a set of E-linearly independent elements. This is the case if and only if $\{\alpha^i\tau^j\}_{0\le i\le d-1, 0\le j\le e-1}$ is a set of P-linearly independent elements, where τ is $\phi_{E\hookrightarrow F}(\alpha_E)$. This follows from the observation that since α_E generates E over the prime field, τ will generate $\phi_{E\hookrightarrow F}(E)$ over the prime field. Using the map ψ_F^{-1} to write elements of F as vectors in $P^{(f)}$, we see that this condition is equivalent to showing that the matrix with entries $\bar{v}_{ij} = \psi_F^{-1}(\alpha^i\tau^j) \in P^{(f)}$ has full P-rank f.

The structure of F as an E-algebra is entirely determined by $\phi = \phi_{E\hookrightarrow F}$ and $\alpha_{F/E}$:

$$F \cong \phi(E) \oplus \alpha_{F/E}\cdot\phi(E) \oplus \cdots \oplus \alpha_{F/E}^{d-1}\cdot\phi(E) \tag{4.1}$$

where d is the degree of F over E.

Ignoring multiplicative structure, the right-hand side of equation (4.1) gives an E-vector space structure $F \cong E^{(d)} \cong \phi(E)^{(d)}$ for F. The map $\psi_{E^{(d)}\leftrightarrow F}$ is constructed as a composite of three vector space isomorphisms:

$$\psi_{E^{(d)}\leftrightarrow F}\colon E^{(d)} \xrightarrow{(\psi_E^{-1})^d} (P^{(e)})^d \xrightarrow{\eta} P^{(f)} \xrightarrow{\psi_F} F \tag{4.2}$$

where $(\psi_E^{-1})^d$ acts on $E^{(d)}$ in the obvious way through concatenation of the images of ψ_E^{-1} on each of the d components. Furthermore, η is the linear transformation from $P^{(f)}$ to itself obtained by mapping the jth basis vector to $\bar{v}_{rs}$, where $j = re + s$, $0 \le s < e$, and where $(P^{(e)})^d$ is identified with $P^{(f)} = P^{(ed)}$ in the obvious way. The matrix N representing the linear transformation η has the following structure: the first e rows contain the P-coefficients for each of $\psi_F^{-1}(\alpha_{F/E}^0\tau^j)$, for $j = 0, 1, \ldots, e-1$, the next e rows contain those for $\psi_F^{-1}(\alpha_{F/E}^1\tau^j)$, etc., so that generally the $(i\times e + j)$th row corresponds to $\psi_F^{-1}(\alpha_{F/E}^i\tau^j)$.

Proposition 4.1. *The map $\psi_{E^d\leftrightarrow F}$ defined by equation (4.2) defines an E-vector space isomorphism, where F is regarded as a vector space over E through application of the mapping $\phi_{E\hookrightarrow F}$.*

Proof. The proof reduces to a series of verifications:

(i) Since each of the maps in (4.2) is clearly P-linear, so is $\psi = \psi_{E^{(d)}\leftrightarrow F}$.

(ii) Consider the image of $e_{ij} = (0, \ldots, 0, \alpha_E^j, 0, \ldots, 0) \in E^{(d)}$ under ψ, where only the ith component is non-zero. Under ψ_E^{-1}, the image of α_E^j is $f_j = (0, \ldots, 0, 1, 0, \ldots, 0) \in P^{(e)}$, with 1 in the jth position. Under η, the vector $(\bar{0}, \ldots, \bar{0}, f_j, \bar{0}, \ldots, \bar{0}) \in (P^{(e)})^d$ maps to $\psi_F^{-1}(\alpha_{F/E}^i \tau^j)$ by the definition of η. Hence $\psi(e_{ij}) = \alpha_{F/E}^i \tau^j$.

(iii) Since α_E generates E over P, it follows from (i) and (ii) that

$$\psi((\epsilon_0, \ldots, \epsilon_{d-1})) = \sum_{i=0}^{d-1} \phi_{E \hookrightarrow F}(\epsilon_i) \alpha_{F/E}^i$$

for any $(\epsilon_0, \ldots, \epsilon_{d-1}) \in E^{(d)}$. Since $\phi_{E \hookrightarrow F}$ is injective and $\alpha_{F/E}$ is a generator of degree d for F over E, this shows that ψ is bijective.

(iv) If $e = (\epsilon_0, \ldots, \epsilon_{d-1}) \in E^{(d)}$ and $\delta \in E$, then

$$\begin{aligned} \psi(\delta \cdot e) = \psi((\delta\epsilon_0, \ldots, \delta\epsilon_{d-1})) &= \sum_{i=0}^{d-1} \phi_{E \hookrightarrow F}(\delta\epsilon_i) \alpha_{F/E}^i \\ &= \phi_{E \hookrightarrow F}(\delta) \sum_{i=0}^{d-1} \phi_{E \hookrightarrow F}(\epsilon_i) \alpha_{F/E}^i = \phi_{E \hookrightarrow F}(\delta) \psi(e). \end{aligned}$$

Thus, ψ is E-linear.

This proves the proposition. □

4.3. MINIMAL POLYNOMIALS AND DEFINING POLYNOMIAL

The minimal polynomial of an element $\beta \in F$ over E is found by computing the powers $1, \beta, \beta^2, \ldots$ and then applying the isomorphism ${\psi_{E^{(d)} \leftrightarrow F}}^{-1}$ to each of these elements until an E-linear relation is obtained. The fact that E and F may have different representations for their elements does not present any difficulties since application of the isomorphism involves only linear algebra over the prime field P.

The defining polynomial $f_{F/E}$ of F over E is calculated as the minimal polynomial of $\alpha_{F/E}$ over E. Often this polynomial will be known at the outset. This is true, for example, if an extension F is defined in terms of a particular polynomial g, in which case the field F has to be constructed so that its defining polynomial is g (see Subsection 5.2).

5. Modifying a Lattice

In this section we describe our implementation of the operations that modify a lattice $\mathcal{L}$, observing that the lattice conditions are preserved in each case. Note that, since the formal lattice model (and consequently the code) is mutually recursive, it is necessary to employ an inductive form of exposition whereby each of the following subsections may require application of operations described in other subsections (but applied to fields of smaller degree).

A final subsection discusses the construction of *default finite fields* in MAGMA.

5.1. construction of an explicit embedding

Suppose that E and F are fields such that the degree of E divides the degree of F. Assume that E and F are currently stored in a Magma lattice $\mathcal{L}$, but that E is not yet explicitly embedded in F. The subfield E is embedded in F by the following procedure:

(i) Create the map $\phi_{E\hookrightarrow F}$ using the method outlined at the end of Subsection 2.5, where it is shown to yield a compatible embedding of E in F.
(ii) Construct additional information associated with the relationship $E \preceq F$, as described in Section 4.
(iii) For each subfield S of F for which there does not yet exist a common subfield G of E and S with $\partial(G) = g = \gcd(\partial(E), \partial(S))$, construct a subfield G of E of degree g using the method outlined in Subsection 5.3 and explicitly embed G in S by recursively applying this procedure.
(iv) For each field $S \in L$ such that $S \preceq E$, use the mapping $\phi_{S\hookrightarrow F} = \phi_{E\hookrightarrow F} \circ \phi_{S\hookrightarrow E}$ to embed S in F. (This corresponds to forming the transitive closure.)

It is easily seen by induction that the compatibility of $\mathcal{L}$ is preserved, using the inductive assumption that any embeddings arising through application of Subsection 5.3 preserve compatibility.

The Magma procedure

```
> Embed(E, F);
```

performs the explicit embedding of E in F. Note that E may already be explicitly embedded in F, in which case no action is necessary. It is also possible to indicate a preference among the possible compatible embeddings by supplying an image for the generator of E.

5.2. creation of an extension field

Let E be a field currently stored in a Magma lattice $\mathcal{L}$ of finite fields and suppose that $f(x) \in E[x]$ is a polynomial of degree d irreducible over E. The *extension field* F of E by the polynomial $f(x)$ as created in Magma appears to the user as the quotient ring $E[x]/(f)$, i.e., as polynomials over E reduced modulo $f(x)$.

If E is the prime field P, then F is constructed as an arbitrary field of cardinality p^d in the appropriate representation as outlined in Subsection 3.4. The prime field P is automatically embedded in F, and the generator $\alpha_{F/P}$ is taken to be some root of $f(x)$ in $F[x]$ so F appears as an extension of $E = P$ by $f(x)$. Since F has no other relationships to other fields in L as yet, the compatibility of $\mathcal{L}$ is preserved.

Assume then that E has cardinality p^e, where $e > 1$. The following procedure uses the method of Proposition 2.2 and Remark 2.3 to create the extension F of E.

(i) Create an arbitrary field F of cardinality p^{ed} in the appropriate representation.
(ii) Compute the minimal polynomial $t(x) \in P[x]$ of $\alpha_{E/P}$ over P using the vector space isomorphism $\psi_{P^{(e)}\leftrightarrow E}$.
(iii) Lift $t(x)$ into $F[x]$ and find a root τ of $t(x)$ in F (possible since P is automatically embedded in F at construction). This gives the embedding map $\phi_{E\hookrightarrow F}$. Using $\phi_{E\hookrightarrow F}$, embed E in F.

(iv) Lift $f(x)$ into $F[x]$ (now possible since E is embedded in F) and set $\alpha_{F/E}$ to a root of this polynomial in F. Thus the defining polynomial of F over E will be $f(x)$ as desired.
(v) For each field $S \in L$ such that $S \preceq E$ use the mapping $\phi_{S\hookrightarrow F} = \phi_{E\hookrightarrow F} \circ \phi_{S\hookrightarrow E}$ to embed S in F. (This corresponds to forming the transitive closure.)

Observing that condition CE6 is preserved since the only embedded subfields of F are E and its subfields and these satisfy condition CE6 by hypothesis, it is easily seen that the compatibility of $\mathcal{L}$ is preserved.

The following MAGMA statement will construct the extension field F of E generated by the irreducible polynomial $f \in E[x]$:

```
> F<t> := ext<E | f>;
```

The element t of F will be set to be the generator $\alpha_{F/E}$ of F over E; the defining polynomial $f_{F/E}$ of F over E will be set to f; and the ground field of F will be set to E. Alternatively, rather than specifying the polynomial f, an integer $d > 1$ may be specified; MAGMA will then set F to an arbitrary but fixed extension of E of degree d.

5.3. CREATION OF A SUBFIELD

Let F be a field currently stored in a lattice $\mathcal{L}$ of finite fields and suppose that β is a non-zero element of F. MAGMA allows the creation of the *subfield* E of F generated by the element β. Specifying the element β, rather than an integer dividing the degree of F, means that the prime-field generator α_E of E maps to β in F. Note that, as β may generate the whole of F, the user may use this mechanism to construct an alternative presentation of F. The subfield E is constructed by the following procedure:

(i) Compute the minimal polynomial $f(x) \in P[x]$ of β over P.
(ii) Create E in the lattice as an extension of P by the polynomial $f(x)$ as explained in Subsection 5.2.
(iii) Embed E in F so that α_E is mapped to $\beta \in F$. Since the minimal polynomial of α_E over P is $f(x)$ by construction, Remark 2.3 shows that this is a correct embedding.
(iv) For each subfield S of F where there does not yet exist a common subfield G of E and S with $\partial(G) = g = \gcd(\partial(E), \partial(S))$, recursively construct a subfield G of E of degree g and explicitly embed G in S.
(v) For each field $G \in L$ such that $F \preceq G$, embed E in G using the composition $\phi_{E\hookrightarrow G} = \phi_{F\hookrightarrow G} \circ \phi_{E\hookrightarrow F}$.

Again it is easily seen that the compatibility of the lattice $\mathcal{L}$ is thereby preserved.

The MAGMA code for constructing the subfield E of F generated by $b \in F$ is:

```
> E<a> := sub<F | b>;
```

Here the element a of E will equal the element b of F when (automatic) coercion is performed. Instead of the element $b \in F$, an integer d which is a divisor of the degree of F may be specified; MAGMA will then set E to any subfield of F of degree d.

5.4. DEFAULT FINITE FIELDS

For each prime p and degree $n \geq 1$, there is a fixed *default* finite field in MAGMA of cardinality p^n. Default fields are created automatically whenever the function `FiniteField` (taking the characteristic and degree as arguments) is invoked, or when an extension field or subfield construction is applied to an existing default field with specification of the degree only. For example,

```
> F := FiniteField(3, 4);
> E := ext<F | 3>;
> S := sub<E | 2>;
```

constructs default fields of cardinalities 3^4, 3^{12} and 3^6, respectively.

Contrary to the situation with fields created using the general extension field and subfield constructions described in preceding subsections, MAGMA creates embedding maps automatically between default finite fields (when cardinalities are appropriate). Thus it is never necessary to apply the `Embed` procedure to default fields.

Default finite fields are frequently the only ones needed in an application since the precise choice of a generating polynomial is often irrelevant.

6. Algorithms for Finite Field Elements

In this section we briefly mention some of the non-trivial algorithms installed in MAGMA for finite field elements.

A basic operation is the determination of the multiplicative order of a non-zero element. The order of an element in a cyclic group C_m can be determined efficiently provided that the prime factorization of the order m of the group is known. For, if $m = \prod p_i^{k_i}$, an element $c \in C_m$ will have order $\prod p_i^{d_i}$, where $0 \leq d_i \leq k_i$ is the least integer such that

$$c^{m/p^{k_i - d_i}} = 1 \in C_m.$$

This result is employed in MAGMA to find multiplicative orders of an element in a finite field; it requires the factorization of $q-1$, where q is the cardinality of the field. Since q may be very large, the direct factorization of $q-1$ may be very expensive. To avoid this cost whenever possible, MAGMA includes a large database containing information about the factorization of integers of the form $b^n \pm 1$ (Brent and te Riele, 1992). During a MAGMA session, moreover, any factorization of an integer of the form $q-1$ is remembered, so that the cardinality of the multiplicative group of a finite field will never have to be factored twice.

In general, the generator of a finite field need not be primitive; the user may have indicated a preference for a non-primitive generator, and even in the default case the cost of factoring $q-1$ prohibits the search for primitive polynomials in larger fields. For a non-prime field, MAGMA computes a primitive element by simply searching for a random element of maximal order. In the case of a prime field of cardinality p, MAGMA searches for one of the $\phi(p-1)$ elements of order $p-1$. In principle, a random search could be performed, but for compatibility reasons, the elements $x = 2, 3, 4, \ldots$ are examined sequentially so as to find the smallest primitive element a (where the ordering is that of the positive integers). Consequently, the minimal polynomial $x - a$ of a is the Conway

polynomial of degree 1 over $\mathbf{F}_p$ as defined in Section 2. Once found, a primitive element is stored with the field.

Interestingly enough, one of the few problems that can be solved in $\mathbf{F}_p$ without a knowledge of the factorization of $p-1$ is that of finding square roots. MAGMA contains an implementation of the Shanks–Tonelli algorithm (Cohen, 1993) and usually finds square roots of squares modulo p very quickly. For higher order roots for prime fields and for all roots in non-prime fields, an nth root of an element $a \in F$ is found by computing a linear factor of $x^n - a$ in $F[x]$ (if it exists) using a variant of the Cantor–Zassenhaus algorithm (Cantor and Zassenhaus, 1981; Knuth, 1969) to do this.

An element $x \in F$ is said to be *normal over a subfield* S of F if the elements $x, x^q, x^{q^2}, \ldots, x^{q^{d-1}}$ form a basis for F over S, where d is the degree of the extension F/S and q is the cardinality of S. MAGMA locates normal elements by generating elements x at random and testing the first q powers of x for linear independence.

The computation of discrete logarithms, along with the factorization of polynomials, plays an important role in many applications of finite fields, including, for example, cryptography (see Menezes (1993)). There are many similarities between *integer* factorization algorithms and discrete logarithm algorithms in finite fields, and a serious implementation of one of the subexponential algorithms is a major undertaking. Currently, MAGMA includes certain 'square root' methods, where the complexity is dominated by a term of the order of $\sqrt{p}$, for the largest prime p dividing the order $q-1$ of the multiplicative group of $\mathbf{F}_q$. The most important algorithm in this class is the *Pohlig–Hellman* algorithm (Pohlig and Hellman, 1978). This algorithm proceeds by finding logarithms in cyclic groups of prime order, building up to p-Sylow subgroups of the multiplicative group and combining the results using the Chinese Remainder Theorem. Logarithms in cyclic groups of prime order are found using either Pollard's ρ-method (Knuth, 1969; Pollard, 1978) or Shanks' baby-step-giant-step method (Knuth, 1969). Neither will perform satisfactorily if the prime is large (in Shanks' method a table of size $\sqrt{p}$ is needed).

7. An Example

The use of the finite field machinery in MAGMA is illustrated in the context of the subfield-lattice of a finite field of cardinality 5^{36}.

We first assign `F36` with generator `w36` to be the default finite field of cardinality 5^{36} and print the defining polynomial of `F36` (the minimal polynomial of `w36` over the prime field).

```
> F36<w36> := FiniteField(5, 36);
> d<x> := DefiningPolynomial(F36); d;
x^36 + 4*x^33 + x^32 + 3*x^31 + 2*x^29 + 2*x^28 + x^27 + 2*x^26 +
    4*x^25 + x^24 + x^23 + 4*x^20 + x^18 + 4*x^15 + 4*x^13 + 3*x^12 +
    2*x^11 + 3*x^10 + 2*x^9 + 2*x^8 + 2*x^7 + x^6 + x^5 + 3*x^4 +
    4*x^3 + 3*x + 2
```

We next create some subfields of `F36`:

```
> F18<w18> := sub<F36 | 18>;
> F12<w12> := sub<F36 | 12>;
```

```
> F6<w6> := sub<F36 | 6>;
> F1 := sub<F36 | 1>;
```

Note that so far only default finite fields have been created.

The defining polynomial chosen for `F12` (over the prime field) is not the same as the Conway polynomial for $\mathbf{F}_{5^{12}}$. So we assign `G` with generator `g` to be the finite field whose defining polynomial is the Conway polynomial.

```
> ConwayPolynomial(5, 12) eq DefiningPolynomial(F12);
false
> G<g> := ext<F1 | ConwayPolynomial(5, 12)>;
> DefiningPolynomial(G);
x^12 + x^7 + x^6 + 4*x^4 + 4*x^3 + 3*x^2 + 2*x + 2
```

We instruct MAGMA to (compatibly) embed `F12` in `G` and then print `g` as an element of `F12`.

```
> Embed(F12, G);
> F12 ! g;
4*w12^11 + w12^10 + 2*w12^7 + 2*w12^6 + 3*w12^5 + 3*w12^4 +
    3*w12^3 + 3*w12^2 + 4*w12 + 1
```

We check that the minimal polynomial of `g` over `F6` is quadratic. Note that `F6` is now automatically a subfield of `G` by transitivity (condition CE5). Since `g` is primitive (being the root of a Conway polynomial), raising it to the power $(5^{12}-1)/(5^6-1)$ will produce an element `h` whose minimal polynomial over the prime field `F1` has degree 6.

```
> m<x6> := MinimalPolynomial(g, F6);  m;
x6^2 + w6^7484*x6 + w6^3281
> h := g^((5^12 - 1) div (5^6 - 1));  h;
g^11 + 2*g^10 + g^9 + 2*g^8 + g^7 + 2*g^5 + 4*g^3 + 4*g^2 + 4*g + 3
> n<x1> := MinimalPolynomial(h, F1);  n;
x1^6 + x1^4 + 4*x1^3 + x1^2 + 2
```

Since all (embedded) fields of the same degree are isomorphic, `h` must lie in `F6`. So we define `h6` to be the element in `F6` equal to `h`.

```
> h in F6;
true
> h6 := F6 ! h; h6;
w6^3281
```

Now we lift `h` into `F36` in two different ways and check that the "diagram" commutes. Lifting `h6` via `F18` should give the same answer as lifting `h` directly into `F36`.

```
> a := F18 ! h6;
> b := F36 ! a;
> c := F36 ! h;
```

```
> b eq c;
true
```

Note that elements in two different fields (`w18` in `F18` and `g` in `G`) may be compared, combined, etc. since they have a common overfield in the lattice. We calculate the order of $\mathtt{z} = \mathtt{w18} + \mathtt{g}$ (an element of `F36`) in factored form:

```
> FactoredOrder(w18 + g);
[
    <2, 1>, <3, 3>, <7, 1>, <13, 1>, <19, 1>, <31, 1>, <37, 1>,
    <601, 1>, <829, 1>, <5167, 1>, <6597973, 1>
]
```

Acknowledgement

The authors would like to thank Bernd Souvignier for critically reading drafts of this paper and making helpful suggestions which resulted in clarification and simplification of the material in Section 2.

References

Bosma, W., Cannon, J. (1996). *Handbook of MAGMA functions*. School of Mathematics and Statistics, Sydney University.

Bosma, W., Cannon, J., Matthews, G. (1994). Programming with algebraic structures: design of the Magma language. In Giesbrecht, M. (ed.), *Proceedings of the 1994 International Symposium on Symbolic and Algebraic Computation*, pp. 52–57, New York, Oxford: ACM Press.

Bosma, W., Cannon, J., Playoust, C. (1997). The Magma algebra system I: the user language. *J. Symbolic Comput.* **24**, 235–265.

Brent, R., te Riele, H. (1992). Factorizations of $a^n \pm 1$, $13 \leq a < 100$. Technical report NM-R9212, Centrum voor Wiskunde en Informatica, Amsterdam.

Cannon, J., Playoust, C. (1996). Magma: A new computer algebra system. *Euromath Bulletin* **2**(1), 113–144.

Cantor, D., Zassenhaus, H. (1981). A new algorithm for factoring polynomials over finite fields. *Math. Comp.* **36**, 587–592.

Cohen, H. (1993). *A Course in Computational Algebraic Number Theory*. Berlin: Springer.

Conway, J. (1968). A tabulation of some information concerning finite fields. In Churchhouse, R., Herz, J. (eds.), *Computers in Mathematical Research*. Amsterdam: North-Holland.

Jansen, C., Lux, K., Parker, R., Wilson, R. (1995). *An Atlas of Brauer Characters*. Oxford: Oxford Science Publishers.

Jungnickel, D. (1993). *Finite Fields*. Mannheim: BI-Wissenschafts.

Knuth, D. E. (1969). *The Art of Computer Programming. Volume 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley.

Lidl, R., Niederreiter, H. (1986). *Finite Fields*. Cambridge: Cambridge University Press.

Menezes, A. (1993). *Applications of Finite Fields*. Boston: Kluwer.

Montgomery, P. L. (1985). Modular multiplication without trial division. *Math. Comp.* **44**, 519–521.

Pohlig, S., Hellman, M. (1978). An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Info. Th.* **24**, 106–110.

Pollard, J. (1978). Monte carlo methods for index computation mod p. *Math. Comp.* **32**, 918–924.

Scheerhorn, A. (1992). Trace- and norm-compatible extensions of finite fields. *Applicable Algebra in Engineering, Communication and Computing* **3**, 199–209.

Originally received 28 July 1995
Accepted 17 March 1997