

Finite Fields

Emin Karayel

February 6, 2026

Abstract

This entry formalizes the classification of the finite fields (also called Galois fields): For each prime power p^n there exists exactly one (up to isomorphisms) finite field of that size and there are no other finite fields. The derivation includes a formalization of the characteristic of rings, the Frobenius endomorphism, formal differentiation for polynomials in HOL-Algebra, Rabin's test for the irreducibility of polynomials and Gauss' formula for the number of monic irreducible polynomials over finite fields:

$$\frac{1}{n} \sum_{d|n} \mu(d) p^{n/d}.$$

The proofs are based on the books and publications from Ireland and Rosen [3], Rabin [5] as well as, Lidl and Niederreiter [4].

Contents

1	Introduction	2
2	Preliminary Results	3
2.1	Summation in the discrete topology	3
2.2	Polynomials	4
2.3	Ring Isomorphisms	6
2.4	Divisibility	8
2.5	Factorization	9
3	Characteristic of Rings	12
4	Formal Derivatives	19
5	Factorization into Monic Polynomials	21
6	Counting Irreducible Polynomials	26
6.1	The polynomial $X^n - X$	26
6.2	Gauss Formula	28

7	Isomorphism between Finite Fields	30
8	Rabin’s test for irreducible polynomials	31
9	Executable Structures	33
10	Executable Polynomial Rings	36
11	Executable Factor Rings	41
12	Executable Code for Rabin’s Irreducibility Test	43
13	Additional results about Bijections and Digit Representations	45
14	Additional results about PMFs	47
15	Executable Polynomial Factor Rings	47
16	Fast algorithms for Computations of Roots	52
17	Algorithms for finding irreducible polynomials	53

1 Introduction

The following section starts with preliminary results. Section 3 introduces the characteristic of rings with the Frobenius endomorphism. Whenever it makes sense, the definitions and facts do not assume the finiteness of the fields or rings. For example the characteristic is defined over arbitrary rings (and also fields). While formal derivatives do exist for type-class based structures in `HOL-Computational_Algebra`, as far as I can tell, they do not exist for the structure based polynomials in `HOL-Algebra`. These are introduced in Section 4.

A cornerstone of the proof is the derivation of Gauss’ formula for the number of monic irreducible polynomials over a finite field R in Section 6.2. The proof follows the derivation by Ireland and Rosen [3, §7] closely, with the caveat that it does not assume that R is a simple prime field, but that it is just a finite field. This works by adjusting a proof step with the information that the order of a finite field must be of the form p^n , where p is the characteristic of the field, derived in Section 3. The final step relies on the Möbius inversion theorem formalized by Eberl [2].¹

¹Thanks to Katharina Kreuzer for discovering that formalization.

With Gauss' formula it is possible to show the existence of the finite fields of order p^n where p is a prime and $n > 0$. During the proof the fact that the polynomial $X^n - X$ splits in a field of order n is also derived, which is necessary for the uniqueness result as well.

The uniqueness proof is inspired by the derivation of the same result in Lidl and Niederreiter [4], but because of the already derived existence proof for irreducible polynomials, it was possible to reduce its complexity.

The classification consists of three theorems:

- *Existence*: For each prime power p^n there exists a finite field of that size. This is shown at the conclusion of Section 6.2.
- *Uniqueness*: Any two finite fields of the same size are isomorphic. This is shown at the conclusion of Section 7.
- *Completeness*: Any finite fields' size must be a prime power. This is shown at the conclusion of Section 3.

2 Preliminary Results

theory *Finite-Fields-Preliminary-Results*
imports *HOL-Algebra.Polynomial-Divisibility*
begin

2.1 Summation in the discrete topology

The following lemmas transfer the corresponding result from the summation over finite sets to summation over functions which vanish outside of a finite set.

lemma *sum'-subtractf-nat*:
fixes $f :: 'a \Rightarrow nat$
assumes $finite \{i \in A. f\ i \neq 0\}$
assumes $\bigwedge i. i \in A \implies g\ i \leq f\ i$
shows $sum' (\lambda i. f\ i - g\ i)\ A = sum' f\ A - sum' g\ A$
(is ?lhs = ?rhs)
<proof>

lemma *sum'-nat-eq-0-iff*:
fixes $f :: 'a \Rightarrow nat$
assumes $finite \{i \in A. f\ i \neq 0\}$
assumes $sum' f\ A = 0$
shows $\bigwedge i. i \in A \implies f\ i = 0$
<proof>

lemma *sum'-eq-iff*:

fixes $f :: 'a \Rightarrow nat$
assumes $finite \{i \in A. f\ i \neq 0\}$
assumes $\bigwedge i. i \in A \implies f\ i \geq g\ i$
assumes $sum' f\ A \leq sum' g\ A$
shows $\forall i \in A. f\ i = g\ i$
 $\langle proof \rangle$

2.2 Polynomials

The embedding of the constant polynomials into the polynomials is injective:

lemma (**in ring**) *poly-of-const-inj*:
 $inj\ poly-of-const$
 $\langle proof \rangle$

lemma (**in domain**) *embed-hom*:
assumes $subring\ K\ R$
shows $ring-hom-ring\ (K[X])\ (poly-ring\ R)\ id$
 $\langle proof \rangle$

The following are versions of the properties of the degrees of polynomials, that abstract over the definition of the polynomial ring structure. In the theories *HOL-Algebra.Polynomials* and also *HOL-Algebra.Polynomial-Divisibility* these abstract version are usually indicated with the suffix “shell”, consider for example: *domain.pdivides-iff-shell*.

lemma (**in ring**) *degree-add-distinct*:
assumes $subring\ K\ R$
assumes $f \in carrier\ (K[X]) - \{0_{K[X]}\}$
assumes $g \in carrier\ (K[X]) - \{0_{K[X]}\}$
assumes $degree\ f \neq degree\ g$
shows $degree\ (f \oplus_{K[X]}\ g) = max\ (degree\ f)\ (degree\ g)$
 $\langle proof \rangle$

lemma (**in ring**) *degree-add*:
 $degree\ (f \oplus_{K[X]}\ g) \leq max\ (degree\ f)\ (degree\ g)$
 $\langle proof \rangle$

lemma (**in domain**) *degree-mult*:
assumes $subring\ K\ R$
assumes $f \in carrier\ (K[X]) - \{0_{K[X]}\}$
assumes $g \in carrier\ (K[X]) - \{0_{K[X]}\}$
shows $degree\ (f \otimes_{K[X]}\ g) = degree\ f + degree\ g$
 $\langle proof \rangle$

lemma (**in ring**) *degree-one*:
 $degree\ (1_{K[X]}) = 0$

<proof>

lemma (in domain) *pow-non-zero*:
 $x \in \text{carrier } R \implies x \neq \mathbf{0} \implies x [\wedge] (n :: \text{nat}) \neq \mathbf{0}$
<proof>

lemma (in domain) *degree-pow*:
assumes *subring* $K R$
assumes $f \in \text{carrier } (K[X]) - \{\mathbf{0}_{K[X]}\}$
shows $\text{degree } (f [\wedge]_{K[X]} n) = \text{degree } f * n$
<proof>

lemma (in ring) *degree-var*:
 $\text{degree } (X_R) = 1$
<proof>

lemma (in domain) *var-carr*:
fixes $n :: \text{nat}$
assumes *subring* $K R$
shows $X_R \in \text{carrier } (K[X]) - \{\mathbf{0}_{K[X]}\}$
<proof>

lemma (in domain) *var-pow-carr*:
fixes $n :: \text{nat}$
assumes *subring* $K R$
shows $X_R [\wedge]_{K[X]} n \in \text{carrier } (K[X]) - \{\mathbf{0}_{K[X]}\}$
<proof>

lemma (in domain) *var-pow-degree*:
fixes $n :: \text{nat}$
assumes *subring* $K R$
shows $\text{degree } (X_R [\wedge]_{K[X]} n) = n$
<proof>

lemma (in domain) *finprod-non-zero*:
assumes *finite* A
assumes $f \in A \rightarrow \text{carrier } R - \{\mathbf{0}\}$
shows $(\otimes_{i \in A} f i) \in \text{carrier } R - \{\mathbf{0}\}$
<proof>

lemma (in domain) *degree-prod*:
assumes *finite* A
assumes *subring* $K R$
assumes $f \in A \rightarrow \text{carrier } (K[X]) - \{\mathbf{0}_{K[X]}\}$
shows $\text{degree } (\otimes_{K[X]} i \in A. f i) = (\sum i \in A. \text{degree } (f i))$
<proof>

lemma (in ring) *coeff-add*:

assumes *subring* $K R$
assumes $f \in \text{carrier } (K[X])$ $g \in \text{carrier } (K[X])$
shows $\text{coeff } (f \oplus_{K[X]} g) i = \text{coeff } f i \oplus_R \text{coeff } g i$
 <proof>

lemma (*in domain*) *coeff-a-inv*:
assumes *subring* $K R$
assumes $f \in \text{carrier } (K[X])$
shows $\text{coeff } (\ominus_{K[X]} f) i = \ominus (\text{coeff } f i)$ (**is** $?L = ?R$)
 <proof>

This is a version of geometric sums for commutative rings:

lemma (*in cring*) *geom*:
fixes $q:: \text{nat}$
assumes [*simp*]: $a \in \text{carrier } R$
shows $(a \ominus \mathbf{1}) \otimes (\bigoplus_{i \in \{..<q\}} a [\uparrow] i) = (a [\uparrow] q \ominus \mathbf{1})$
 (**is** $?lhs = ?rhs$)
 <proof>

lemma (*in domain*) *rupture-eq-0-iff*:
assumes *subfield* $K R$ $p \in \text{carrier } (K[X])$ $q \in \text{carrier } (K[X])$
shows $\text{rupture-surj } K p q = \mathbf{0}_{\text{Rupt } K p} \longleftrightarrow p \text{ pdivides } q$
 (**is** $?lhs \longleftrightarrow ?rhs$)
 <proof>

2.3 Ring Isomorphisms

The following lemma shows that an isomorphism between domains also induces an isomorphism between the corresponding polynomial rings.

lemma *lift-iso-to-poly-ring*:
assumes $h \in \text{ring-iso } R S$ *domain* R *domain* S
shows $\text{map } h \in \text{ring-iso } (\text{poly-ring } R) (\text{poly-ring } S)$
 <proof>

lemma *carrier-hom*:
assumes $f \in \text{carrier } (\text{poly-ring } R)$
assumes $h \in \text{ring-iso } R S$ *domain* R *domain* S
shows $\text{map } h f \in \text{carrier } (\text{poly-ring } S)$
 <proof>

lemma *carrier-hom'*:
assumes $f \in \text{carrier } (\text{poly-ring } R)$
assumes $h \in \text{ring-hom } R S$
assumes *domain* R *domain* S
assumes *inj-on* h (*carrier* R)
shows $\text{map } h f \in \text{carrier } (\text{poly-ring } S)$

$\langle proof \rangle$

The following lemmas transfer properties like divisibility, irreducibility etc. between ring isomorphisms.

lemma *divides-hom*:

assumes $h \in \text{ring-iso } R S$

assumes $\text{domain } R \text{ domain } S$

assumes $x \in \text{carrier } R y \in \text{carrier } R$

shows $x \text{ divides}_R y \longleftrightarrow (h x) \text{ divides}_S (h y)$ (**is** $?lhs \longleftrightarrow ?rhs$)

$\langle proof \rangle$

lemma *properfactor-hom*:

assumes $h \in \text{ring-iso } R S$

assumes $\text{domain } R \text{ domain } S$

assumes $x \in \text{carrier } R b \in \text{carrier } R$

shows $\text{properfactor } R b x \longleftrightarrow \text{properfactor } S (h b) (h x)$

$\langle proof \rangle$

lemma *Units-hom*:

assumes $h \in \text{ring-iso } R S$

assumes $\text{domain } R \text{ domain } S$

assumes $x \in \text{carrier } R$

shows $x \in \text{Units } R \longleftrightarrow h x \in \text{Units } S$

$\langle proof \rangle$

lemma *irreducible-hom*:

assumes $h \in \text{ring-iso } R S$

assumes $\text{domain } R \text{ domain } S$

assumes $x \in \text{carrier } R$

shows $\text{irreducible } R x = \text{irreducible } S (h x)$

$\langle proof \rangle$

lemma *pirreducible-hom*:

assumes $h \in \text{ring-iso } R S$

assumes $\text{domain } R \text{ domain } S$

assumes $f \in \text{carrier } (\text{poly-ring } R)$

shows $\text{pirreducible}_R (\text{carrier } R) f =$

$\text{pirreducible}_S (\text{carrier } S) (\text{map } h f)$

(**is** $?lhs = ?rhs$)

$\langle proof \rangle$

lemma *ring-hom-cong*:

assumes $\bigwedge x. x \in \text{carrier } R \implies f' x = f x$

assumes $\text{ring } R$

assumes $f \in \text{ring-hom } R S$

shows $f' \in \text{ring-hom } R S$

$\langle proof \rangle$

The natural homomorphism between factor rings, where one

ideal is a subset of the other.

lemma (in *ring*) *quot-quot-hom*:
 assumes *ideal I R*
 assumes *ideal J R*
 assumes $I \subseteq J$
 shows $(\lambda x. (J \langle + \rangle_R x)) \in \text{ring-hom } (R \text{ Quot } I) (R \text{ Quot } J)$
 $\langle \text{proof} \rangle$

lemma (in *ring*) *quot-carr*:
 assumes *ideal I R*
 assumes $y \in \text{carrier } (R \text{ Quot } I)$
 shows $y \subseteq \text{carrier } R$
 $\langle \text{proof} \rangle$

lemma (in *ring*) *set-add-zero*:
 assumes $A \subseteq \text{carrier } R$
 shows $\{\mathbf{0}\} \langle + \rangle_R A = A$
 $\langle \text{proof} \rangle$

Adapted from the proof of *domain.polynomial-rupture*

lemma (in *domain*) *rupture-surj-as-eval*:
 assumes *subring K R*
 assumes $p \in \text{carrier } (K[X])$ $q \in \text{carrier } (K[X])$
 shows *rupture-surj K p q =*
 ring.eval (Rupt K p) (map ((rupture-surj K p) o poly-of-const) q)
 (rupture-surj K p X)
 $\langle \text{proof} \rangle$

2.4 Divisibility

lemma (in *field*) *f-comm-group-1*:
 assumes $x \in \text{carrier } R$ $y \in \text{carrier } R$
 assumes $x \neq \mathbf{0}$ $y \neq \mathbf{0}$
 assumes $x \otimes y = \mathbf{0}$
 shows *False*
 $\langle \text{proof} \rangle$

lemma (in *field*) *f-comm-group-2*:
 assumes $x \in \text{carrier } R$
 assumes $x \neq \mathbf{0}$
 shows $\exists y \in \text{carrier } R - \{\mathbf{0}\}. y \otimes x = \mathbf{1}$
 $\langle \text{proof} \rangle$

sublocale *field < mult-of: comm-group mult-of R*
 rewrites *mult (mult-of R) = mult R*
 and one *(mult-of R) = one R*
 $\langle \text{proof} \rangle$

lemma (in *domain*) *div-neg*:

assumes $a \in \text{carrier } R \ b \in \text{carrier } R$
assumes $a \text{ divides } b$
shows $a \text{ divides } (\ominus b)$
 <proof>

lemma (in domain) *div-sum*:
assumes $a \in \text{carrier } R \ b \in \text{carrier } R \ c \in \text{carrier } R$
assumes $a \text{ divides } b$
assumes $a \text{ divides } c$
shows $a \text{ divides } (b \oplus c)$
 <proof>

lemma (in domain) *div-sum-iff*:
assumes $a \in \text{carrier } R \ b \in \text{carrier } R \ c \in \text{carrier } R$
assumes $a \text{ divides } b$
shows $a \text{ divides } (b \oplus c) \longleftrightarrow a \text{ divides } c$
 <proof>

lemma (in comm-monoid) *irreducible-prod-unit*:
assumes $f \in \text{carrier } G \ x \in \text{Units } G$
shows $\text{irreducible } G \ f = \text{irreducible } G \ (x \otimes f)$ (is ?L = ?R)
 <proof>

end

2.5 Factorization

theory *Finite-Fields-Factorization-Ext*
imports *Finite-Fields-Preliminary-Results*
begin

This section contains additional results building on top of the development in *HOL-Algebra.Divisibility* about factorization in a *factorial-monoid*.

definition *factor-mset* **where** *factor-mset* $G \ x =$
 (THE $f. (\exists \text{ as. } f = \text{fmset } G \ \text{as} \wedge \text{wfactors } G \ \text{as} \ x \wedge \text{set } \text{as} \subseteq \text{carrier } G)$)

In *HOL-Algebra.Divisibility* it is already verified that the multiset representing the factorization of an element of a factorial monoid into irreducible factors is well-defined. With these results it is then possible to define *factor-mset* and show its properties, without referring to a factorization in list form first.

definition *multiplicity* **where**
multiplicity $G \ d \ g = \text{Max } \{(n::\text{nat}). (d \ [\]_G \ n) \text{ divides }_G \ g\}$

definition *canonical-irreducibles* **where**
canonical-irreducibles $G \ A = ($

$$\begin{aligned}
& A \subseteq \{a. a \in \text{carrier } G \wedge \text{irreducible } G a\} \wedge \\
& (\forall x y. x \in A \longrightarrow y \in A \longrightarrow x \sim_G y \longrightarrow x = y) \wedge \\
& (\forall x \in \text{carrier } G. \text{irreducible } G x \longrightarrow (\exists y \in A. x \sim_G y))
\end{aligned}$$

A set of irreducible elements that contains exactly one element from each equivalence class of an irreducible element formed by association, is called a set of *canonical-irreducibles*. An example is the set of monic irreducible polynomials as representatives of all irreducible polynomials.

context *factorial-monoid*
begin

lemma *assoc-as-fmset-eq:*

assumes *wfactors* G *as* a
and *wfactors* G *bs* b
and $a \in \text{carrier } G$
and $b \in \text{carrier } G$
and $\text{set } as \subseteq \text{carrier } G$
and $\text{set } bs \subseteq \text{carrier } G$
shows $a \sim b \iff (\text{fmset } G as = \text{fmset } G bs)$

<proof>

lemma *factor-mset-aux-1:*

assumes $a \in \text{carrier } G$ $\text{set } as \subseteq \text{carrier } G$ *wfactors* G *as* a
shows $\text{factor-mset } G a = \text{fmset } G as$

<proof>

lemma *factor-mset-aux:*

assumes $a \in \text{carrier } G$
shows $\exists as. \text{factor-mset } G a = \text{fmset } G as \wedge \text{wfactors } G as a \wedge$
 $\text{set } as \subseteq \text{carrier } G$

<proof>

lemma *factor-mset-set:*

assumes $a \in \text{carrier } G$
assumes $x \in \# \text{ factor-mset } G a$
obtains y **where**
 $y \in \text{carrier } G$
 $\text{irreducible } G y$
 $\text{assocs } G y = x$

<proof>

lemma *factor-mset-mult:*

assumes $a \in \text{carrier } G$ $b \in \text{carrier } G$
shows $\text{factor-mset } G (a \otimes b) = \text{factor-mset } G a + \text{factor-mset } G b$

<proof>

lemma *factor-mset-unit:* $\text{factor-mset } G \mathbf{1} = \{\#\}$

<proof>

lemma *factor-mset-irred*:

assumes $x \in \text{carrier } G$ *irreducible* G x

shows $\text{factor-mset } G$ $x = \text{image-mset } (\text{assocs } G) \{ \#x\# \}$

$\langle \text{proof} \rangle$

lemma *factor-mset-divides*:

assumes $a \in \text{carrier } G$ $b \in \text{carrier } G$

shows a *divides* $b \iff \text{factor-mset } G$ $a \subseteq\# \text{factor-mset } G$ b

$\langle \text{proof} \rangle$

lemma *factor-mset-sim*:

assumes $a \in \text{carrier } G$ $b \in \text{carrier } G$

shows $a \sim b \iff \text{factor-mset } G$ $a = \text{factor-mset } G$ b

$\langle \text{proof} \rangle$

lemma *factor-mset-prod*:

assumes *finite* A

assumes $f \text{ ' } A \subseteq \text{carrier } G$

shows $\text{factor-mset } G$ $(\bigotimes a \in A. f a) =$
 $(\sum a \in A. \text{factor-mset } G (f a))$

$\langle \text{proof} \rangle$

lemma *factor-mset-pow*:

assumes $a \in \text{carrier } G$

shows $\text{factor-mset } G$ $(a [\wedge] n) = \text{repeat-mset } n$ $(\text{factor-mset } G$ $a)$

$\langle \text{proof} \rangle$

lemma *image-mset-sum*:

assumes *finite* F

shows

$\text{image-mset } h$ $(\sum x \in F. f x) = (\sum x \in F. \text{image-mset } h (f x))$

$\langle \text{proof} \rangle$

lemma *decomp-mset*:

$(\sum x \in \text{set-mset } R. \text{replicate-mset } (\text{count } R$ $x) x) = R$

$\langle \text{proof} \rangle$

lemma *factor-mset-count*:

assumes $a \in \text{carrier } G$ $d \in \text{carrier } G$ *irreducible* G d

shows $\text{count } (\text{factor-mset } G$ $a)$ $(\text{assocs } G$ $d) = \text{multiplicity } G$ d a

$\langle \text{proof} \rangle$

lemma *multiplicity-ge-iff*:

assumes $d \in \text{carrier } G$ *irreducible* G d $a \in \text{carrier } G$

shows $\text{multiplicity } G$ d $a \geq k \iff d$ $[\wedge] k$ *divides* a

(**is** $?lhs \iff ?rhs$)

$\langle \text{proof} \rangle$

lemma *multiplicity-gt-0-iff*:

assumes $d \in \text{carrier } G$ *irreducible* G d $a \in \text{carrier } G$

shows $\text{multiplicity } G$ d $a > 0 \iff d$ *divides* a

<proof>

lemma *factor-mset-count-2*:

assumes $a \in \text{carrier } G$

assumes $\bigwedge z. z \in \text{carrier } G \implies \text{irreducible } G$ $z \implies y \neq \text{assocs } G$ z

shows $\text{count } (\text{factor-mset } G$ $a)$ $y = 0$

<proof>

lemma *factor-mset-choose*:

assumes $a \in \text{carrier } G$ *set-mset* $R \subseteq \text{carrier } G$

assumes $\text{image-mset } (\text{assocs } G)$ $R = \text{factor-mset } G$ a

shows $a \sim (\bigotimes_{x \in \text{set-mset } R. x} [\wedge] \text{count } R$ $x)$ (**is** $a \sim ?rhs$)

<proof>

lemma *divides-iff-mult-mono*:

assumes $a \in \text{carrier } G$ $b \in \text{carrier } G$

assumes *canonical-irreducibles* G R

assumes $\bigwedge d. d \in R \implies \text{multiplicity } G$ d $a \leq \text{multiplicity } G$ d b

shows a *divides* b

<proof>

lemma *count-image-mset-inj*:

assumes *inj-on* f R $x \in R$ *set-mset* $A \subseteq R$

shows $\text{count } (\text{image-mset } f$ $A)$ $(f$ $x) = \text{count } A$ x

<proof>

Factorization of an element from a *factorial-monoid* using a selection of representatives from each equivalence class formed by (\sim) .

lemma *split-factors*:

assumes *canonical-irreducibles* G R

assumes $a \in \text{carrier } G$

shows

finite $\{d. d \in R \wedge \text{multiplicity } G$ d $a > 0\}$

$a \sim (\bigotimes_{d \in \{d. d \in R \wedge \text{multiplicity } G$ d $a > 0\}.}$

d $[\wedge] \text{multiplicity } G$ d $a)$ (**is** $a \sim ?rhs$)

<proof>

end

end

3 Characteristic of Rings

theory *Ring-Characteristic*

```

imports
  Finite-Fields-Factorization-Ext
  HOL-Algebra.IntRing
  HOL-Algebra.Embedded-Algebras
begin

locale finite-field = field +
  assumes finite-carrier: finite (carrier R)
begin

lemma finite-field-min-order:
  order R > 1
  ⟨proof⟩

lemma (in finite-field) order-pow-eq-self:
  assumes x ∈ carrier R
  shows x [^] (order R) = x
  ⟨proof⟩

lemma (in finite-field) order-pow-eq-self':
  assumes x ∈ carrier R
  shows x [^] (order R ^ d) = x
  ⟨proof⟩

end

lemma finite-fieldI:
  assumes field R
  assumes finite (carrier R)
  shows finite-field R
  ⟨proof⟩

lemma (in domain) finite-domain-units:
  assumes finite (carrier R)
  shows Units R = carrier R - {0} (is ?lhs = ?rhs)
  ⟨proof⟩

The following theorem can be found in Lidl and Niederreiter [4,
Theorem 1.31].

theorem finite-domains-are-fields:
  assumes domain R
  assumes finite (carrier R)
  shows finite-field R
  ⟨proof⟩

definition zfact-iso :: nat ⇒ nat ⇒ int set where
  zfact-iso p k = IdlZ {int p} +>Z (int k)

context

```

```

fixes  $n :: \text{nat}$ 
assumes  $n\text{-gt-0}: n > 0$ 
begin

private abbreviation  $I$  where  $I \equiv \text{Idl}_{\mathcal{Z}} \{ \text{int } n \}$ 

private lemma  $\text{ideal-I}: \text{ideal } I \ \mathcal{Z}$ 
   $\langle \text{proof} \rangle$ 

lemma  $\text{int-cosetI}$ :
  assumes  $u \bmod (\text{int } n) = v \bmod (\text{int } n)$ 
  shows  $\text{Idl}_{\mathcal{Z}} \{ \text{int } n \} +>_{\mathcal{Z}} u = \text{Idl}_{\mathcal{Z}} \{ \text{int } n \} +>_{\mathcal{Z}} v$ 
   $\langle \text{proof} \rangle$ 

lemma  $\text{zfact-iso-inj}$ :
   $\text{inj-on } (\text{zfact-iso } n) \{ ..<n \}$ 
   $\langle \text{proof} \rangle$ 

lemma  $\text{zfact-iso-ran}$ :
   $\text{zfact-iso } n \text{ ' } \{ ..<n \} = \text{carrier } (\text{ZFact } (\text{int } n))$ 
   $\langle \text{proof} \rangle$ 

lemma  $\text{zfact-iso-bij}$ :
   $\text{bij-betw } (\text{zfact-iso } n) \{ ..<n \} (\text{carrier } (\text{ZFact } (\text{int } n)))$ 
   $\langle \text{proof} \rangle$ 

lemma  $\text{card-zfact-carr}$ :  $\text{card } (\text{carrier } (\text{ZFact } (\text{int } n))) = n$ 
   $\langle \text{proof} \rangle$ 

lemma  $\text{fin-zfact}$ :  $\text{finite } (\text{carrier } (\text{ZFact } (\text{int } n)))$ 
   $\langle \text{proof} \rangle$ 

end

lemma  $\text{zfact-prime-is-finite-field}$ :
  assumes  $\text{Factorial-Ring.prime } p$ 
  shows  $\text{finite-field } (\text{ZFact } (\text{int } p))$ 
   $\langle \text{proof} \rangle$ 

definition  $\text{int-embed} :: - \Rightarrow \text{int} \Rightarrow -$  where
   $\text{int-embed } R \ k = \text{add-pow } R \ k \ \mathbf{1}_R$ 

lemma (in ring)  $\text{add-pow-consistent}$ :
  fixes  $i :: \text{int}$ 
  assumes  $\text{subring } K \ R$ 
  assumes  $k \in K$ 
  shows  $\text{add-pow } R \ i \ k = \text{add-pow } (R \ \langle \text{carrier } := K \ \rangle) \ i \ k$ 
  (is  $?lhs = ?rhs$ )
   $\langle \text{proof} \rangle$ 

```

lemma (in ring) *int-embed-consistent*:
assumes *subring K R*
shows $\text{int-embed } R \ i = \text{int-embed } (R \ (\! \text{carrier} := K \)) \ i$
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-closed*:
 $\text{int-embed } R \ k \in \text{carrier } R$
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-range*:
assumes *subring K R*
shows $\text{int-embed } R \ k \in K$
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-zero*:
 $\text{int-embed } R \ 0 = \mathbf{0}_R$
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-one*:
 $\text{int-embed } R \ 1 = \mathbf{1}_R$
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-add*:
 $\text{int-embed } R \ (x+y) = \text{int-embed } R \ x \oplus_R \text{int-embed } R \ y$
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-inv*:
 $\text{int-embed } R \ (-x) = \ominus_R \text{int-embed } R \ x$ (is ?lhs = ?rhs)
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-diff*:
 $\text{int-embed } R \ (x-y) = \text{int-embed } R \ x \ominus_R \text{int-embed } R \ y$
(is ?lhs = ?rhs)
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-mult-aux*:
 $\text{int-embed } R \ (x*\text{int } y) = \text{int-embed } R \ x \otimes \text{int-embed } R \ y$
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-mult*:
 $\text{int-embed } R \ (x*y) = \text{int-embed } R \ x \otimes_R \text{int-embed } R \ y$
 $\langle \text{proof} \rangle$

lemma (in ring) *int-embed-ring-hom*:
 $\text{ring-hom-ring int-ring } R \ (\text{int-embed } R)$
 $\langle \text{proof} \rangle$

abbreviation *char-subring where*

$\text{char-subring } R \equiv \text{int-embed } R \text{ ' UNIV}$

definition *char where*

$\text{char } R = \text{card } (\text{char-subring } R)$

This is a non-standard definition for the characteristic of a ring. Commonly [4, Definition 1.43] it is defined to be the smallest natural number n such that n -times repeated addition of any number is zero. If no such number exists then it is defined to be 0. In the case of rings with unit elements — not that the locale *Ring.ring* requires unit elements — the above definition can be simplified to the number of times the unit elements needs to be repeatedly added to reach 0.

The following three lemmas imply that the definition of the characteristic here coincides with the latter definition.

lemma (*in ring*) *char-bound*:

assumes $x > 0$

assumes $\text{int-embed } R \text{ (int } x) = \mathbf{0}$

shows $\text{char } R \leq x \text{ char } R > 0$

<proof>

lemma (*in ring*) *embed-char-eq-0*:

$\text{int-embed } R \text{ (int (char } R)) = \mathbf{0}$

<proof>

lemma (*in ring*) *embed-char-eq-0-iff*:

fixes $n :: \text{int}$

shows $\text{int-embed } R \ n = \mathbf{0} \longleftrightarrow \text{char } R \ \text{dvd } n$

<proof>

This result can be found in [4, Theorem 1.44].

lemma (*in domain*) *characteristic-is-prime*:

assumes $\text{char } R > 0$

shows $\text{prime } (\text{char } R)$

<proof>

lemma (*in ring*) *char-ring-is-subring*:

$\text{subring } (\text{char-subring } R) \ R$

<proof>

lemma (*in cring*) *char-ring-is-subcring*:

$\text{subcring } (\text{char-subring } R) \ R$

<proof>

lemma (*in domain*) *char-ring-is-subdomain*:

$\text{subdomain } (\text{char-subring } R) \ R$

<proof>

lemma *image-set-eqI*:
assumes $\bigwedge x. x \in A \implies f x \in B$
assumes $\bigwedge x. x \in B \implies g x \in A \wedge f (g x) = x$
shows $f ` A = B$
 $\langle proof \rangle$

This is the binomial expansion theorem for commutative rings.

lemma (**in** *cring*) *binomial-expansion*:
fixes $n :: nat$
assumes [*simp*]: $x \in carrier R \ y \in carrier R$
shows $(x \oplus y) [\wedge] n =$
 $(\bigoplus k \in \{..n\}. int-embed R (n \ choose \ k) \otimes x [\wedge] k \otimes y [\wedge] (n-k))$
 $\langle proof \rangle$

lemma *bin-prime-factor*:
assumes *prime* p
assumes $k > 0 \ k < p$
shows $p \ dvd \ (p \ choose \ k)$
 $\langle proof \rangle$

theorem (**in** *domain*) *freshmans-dream*:
assumes $char R > 0$
assumes [*simp*]: $x \in carrier R \ y \in carrier R$
shows $(x \oplus y) [\wedge] (char R) = x [\wedge] char R \oplus y [\wedge] char R$
 $(is \ ?lhs = ?rhs)$
 $\langle proof \rangle$

The following theorem is sometimes called Freshman's dream for obvious reasons, it can be found in Lidl and Niederreiter [4, Theorem 1.46].

lemma (**in** *domain*) *freshmans-dream-ext*:
fixes m
assumes $char R > 0$
assumes [*simp*]: $x \in carrier R \ y \in carrier R$
defines $n \equiv char R \wedge m$
shows $(x \oplus y) [\wedge] n = x [\wedge] n \oplus y [\wedge] n$
 $(is \ ?lhs = ?rhs)$
 $\langle proof \rangle$

The following is a generalized version of the Frobenius homomorphism. The classic version of the theorem is the case where $k = 1$.

theorem (**in** *domain*) *frobenius-hom*:
assumes $char R > 0$
assumes $m = char R \wedge k$
shows *ring-hom-cring* $R R (\lambda x. x [\wedge] m)$
 $\langle proof \rangle$

lemma (in domain) *char-ring-is-subfield*:

assumes $\text{char } R > 0$

shows $\text{subfield } (\text{char-subring } R) R$

<proof>

lemma *card-lists-length-eq'*:

fixes $A :: 'a \text{ set}$

shows $\text{card } \{xs. \text{set } xs \subseteq A \wedge \text{length } xs = n\} = \text{card } A ^ n$

<proof>

lemma (in ring) *card-span*:

assumes $\text{subfield } K R$

assumes $\text{independent } K w$

assumes $\text{set } w \subseteq \text{carrier } R$

shows $\text{card } (\text{Span } K w) = \text{card } K ^{\text{length } w}$

<proof>

lemma (in ring) *finite-carr-imp-char-ge-0*:

assumes $\text{finite } (\text{carrier } R)$

shows $\text{char } R > 0$

<proof>

lemma (in ring) *char-consistent*:

assumes $\text{subring } H R$

shows $\text{char } (R \upharpoonright \text{carrier } := H) = \text{char } R$

<proof>

lemma (in ring-hom-ring) *char-consistent*:

assumes $\text{inj-on } h (\text{carrier } R)$

shows $\text{char } R = \text{char } S$

<proof>

definition *char-iso* :: $- \Rightarrow \text{int set} \Rightarrow 'a$

where $\text{char-iso } R x = \text{the-elem } (\text{int-embed } R ' x)$

The function *char-iso* R denotes the isomorphism between $ZFact$ $(\text{int } (\text{char } R))$ and the characteristic subring.

lemma (in ring) *char-iso*: $\text{char-iso } R \in$

$\text{ring-iso } (ZFact (\text{char } R)) (R \upharpoonright \text{carrier } := \text{char-subring } R)$

<proof>

The size of a finite field must be a prime power. This can be found in Ireland and Rosen [3, Proposition 7.1.3].

theorem (in finite-field) *finite-field-order*:

$\exists n. \text{order } R = \text{char } R ^ n \wedge n > 0$

<proof>

end

4 Formal Derivatives

theory *Formal-Polynomial-Derivatives*
imports *HOL-Algebra.Polynomial-Divisibility Ring-Characteristic*
begin

definition *pderiv* (*pderiv*) **where**
 $pderiv_R x = ring.normalize R ($
 $map (\lambda i. int-embed R i \otimes_R ring.coeff R x i) (rev [1..<length x]))$

context *domain*
begin

lemma *coeff-range*:
assumes *subring* $K R$
assumes $f \in carrier (K[X])$
shows $coeff f i \in K$
<proof>

lemma *pderiv-carr*:
assumes *subring* $K R$
assumes $f \in carrier (K[X])$
shows $pderiv f \in carrier (K[X])$
<proof>

lemma *pderiv-coeff*:
assumes *subring* $K R$
assumes $f \in carrier (K[X])$
shows $coeff (pderiv f) k = int-embed R (Suc k) \otimes coeff f (Suc k)$
(is ?lhs = ?rhs)
<proof>

lemma *pderiv-const*:
assumes $degree x = 0$
shows $pderiv x = \mathbf{0}_{K[X]}$
<proof>

lemma *pderiv-var*:
shows $pderiv X = \mathbf{1}_{K[X]}$
<proof>

lemma *pderiv-zero*:
shows $pderiv \mathbf{0}_{K[X]} = \mathbf{0}_{K[X]}$
<proof>

lemma *pderiv-add*:
assumes *subring* $K R$
assumes [*simp*]: $f \in carrier (K[X])$ $g \in carrier (K[X])$
shows $pderiv (f \oplus_{K[X]} g) = pderiv f \oplus_{K[X]} pderiv g$

(is ?lhs = ?rhs)
 <proof>

lemma pderiv-inv:
assumes subring $K R$
assumes [simp]: $f \in \text{carrier } (K[X])$
shows $pderiv (\ominus_{K[X]} f) = \ominus_{K[X]} pderiv f$ (is ?lhs = ?rhs)
 <proof>

lemma coeff-mult:
assumes subring $K R$
assumes $f \in \text{carrier } (K[X])$ $g \in \text{carrier } (K[X])$
shows $\text{coeff } (f \otimes_{K[X]} g) i =$
 $(\bigoplus_{k \in \{..i\}} (\text{coeff } f) k \otimes (\text{coeff } g) (i - k))$
 <proof>

lemma pderiv-mult:
assumes subring $K R$
assumes [simp]: $f \in \text{carrier } (K[X])$ $g \in \text{carrier } (K[X])$
shows $pderiv (f \otimes_{K[X]} g) =$
 $pderiv f \otimes_{K[X]} g \oplus_{K[X]} f \otimes_{K[X]} pderiv g$
 (is ?lhs = ?rhs)
 <proof>

lemma pderiv-pow:
assumes $n > (0 :: \text{nat})$
assumes subring $K R$
assumes [simp]: $f \in \text{carrier } (K[X])$
shows $pderiv (f [\wedge]_{K[X]} n) =$
 $\text{int-embed } (K[X]) n \otimes_{K[X]} f [\wedge]_{K[X]} (n-1) \otimes_{K[X]} pderiv f$
 (is ?lhs = ?rhs)
 <proof>

lemma pderiv-var-pow:
assumes $n > (0 :: \text{nat})$
assumes subring $K R$
shows $pderiv (X [\wedge]_{K[X]} n) =$
 $\text{int-embed } (K[X]) n \otimes_{K[X]} X [\wedge]_{K[X]} (n-1)$
 <proof>

lemma int-embed-consistent-with-poly-of-const:
assumes subring $K R$
shows $\text{int-embed } (K[X]) m = \text{poly-of-const } (\text{int-embed } R m)$
 <proof>

end

end

5 Factorization into Monic Polynomials

theory *Monic-Polynomial-Factorization*

imports

Finite-Fields-Factorization-Ext

Formal-Polynomial-Derivatives

begin

hide-const *Factorial-Ring.multiplicity*

hide-const *Factorial-Ring.irreducible*

lemma (in *domain*) *finprod-mult-of*:

assumes *finite A*

assumes $\bigwedge x. x \in A \implies f x \in \text{carrier } (\text{mult-of } R)$

shows $\text{finprod } R f A = \text{finprod } (\text{mult-of } R) f A$

<proof>

lemma (in *ring*) *finite-poly*:

assumes *subring K R*

assumes *finite K*

shows

finite $\{f. f \in \text{carrier } (K[X]) \wedge \text{degree } f = n\}$ (**is finite** ?A)

finite $\{f. f \in \text{carrier } (K[X]) \wedge \text{degree } f \leq n\}$ (**is finite** ?B)

<proof>

definition *pmult* :: $- \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow \text{nat}$ (*pmult*)

where $\text{pmult}_R d p = \text{multiplicity } (\text{mult-of } (\text{poly-ring } R)) d p$

definition *monic-poly* :: $- \Rightarrow 'a \text{ list} \Rightarrow \text{bool}$

where *monic-poly* *R f* =

$(f \neq [] \wedge \text{lead-coeff } f = \mathbf{1}_R \wedge f \in \text{carrier } (\text{poly-ring } R))$

definition *monic-irreducible-poly* where

monic-irreducible-poly *R f* =

$(\text{monic-poly } R f \wedge \text{pirreducible}_R (\text{carrier } R) f)$

abbreviation *m-i-p* \equiv *monic-irreducible-poly*

locale *polynomial-ring* = *field* +

fixes *K*

assumes *polynomial-ring-assms*: *subfield K R*

begin

lemma *K-subring*: *subring K R*

<proof>

abbreviation *P* where $P \equiv K[X]$

This locale is used to specialize the following lemmas for a fixed coefficient ring. It can be introduced in a context as an interpretation to be able to use the following specialized lemmas. Because it is not (and should not) introduced as a sublocale it has no lasting effect for the field locale itself.

lemmas

```

    poly-mult-lead-coeff = poly-mult-lead-coeff[OF K-subring]
  and degree-add-distinct = degree-add-distinct[OF K-subring]
  and coeff-add = coeff-add[OF K-subring]
  and var-closed = var-closed[OF K-subring]
  and degree-prod = degree-prod[OF - K-subring]
  and degree-pow = degree-pow[OF K-subring]
  and pirreducible-degree = pirreducible-degree[OF polynomial-ring-assms]
  and degree-one-imp-pirreducible =
    degree-one-imp-pirreducible[OF polynomial-ring-assms]
  and var-pow-closed = var-pow-closed[OF K-subring]
  and var-pow-carr = var-pow-carr[OF K-subring]
  and univ-poly-a-inv-degree = univ-poly-a-inv-degree[OF K-subring]
  and var-pow-degree = var-pow-degree[OF K-subring]
  and pdivides-zero = pdivides-zero[OF K-subring]
  and pdivides-imp-degree-le = pdivides-imp-degree-le[OF K-subring]
  and var-carr = var-carr[OF K-subring]
  and rupture-eq-0-iff = rupture-eq-0-iff[OF polynomial-ring-assms]
  and rupture-is-field-iff-pirreducible =
    rupture-is-field-iff-pirreducible[OF polynomial-ring-assms]
  and rupture-surj-hom = rupture-surj-hom[OF K-subring]
  and canonical-embedding-ring-hom =
    canonical-embedding-ring-hom[OF K-subring]
  and rupture-surj-norm-is-hom = rupture-surj-norm-is-hom[OF K-subring]
  and rupture-surj-as-eval = rupture-surj-as-eval[OF K-subring]
  and eval-cring-hom = eval-cring-hom[OF K-subring]
  and coeff-range = coeff-range[OF K-subring]
  and finite-poly = finite-poly[OF K-subring]
  and int-embed-consistent-with-poly-of-const =
    int-embed-consistent-with-poly-of-const[OF K-subring]
  and pderiv-var-pow = pderiv-var-pow[OF - K-subring]
  and pderiv-add = pderiv-add[OF K-subring]
  and pderiv-inv = pderiv-inv[OF K-subring]
  and pderiv-mult = pderiv-mult[OF K-subring]
  and pderiv-pow = pderiv-pow[OF - K-subring]
  and pderiv-carr = pderiv-carr[OF K-subring]

```

sublocale *p:principal-domain poly-ring R*

⟨*proof*⟩

end

context *field*

begin

interpretation *polynomial-ring R carrier R*
⟨*proof*⟩

lemma *pdivides-mult-r:*

assumes $a \in \text{carrier } (\text{mult-of } P)$
assumes $b \in \text{carrier } (\text{mult-of } P)$
assumes $c \in \text{carrier } (\text{mult-of } P)$
shows $a \otimes_P c \text{ pdivides } b \otimes_P c \longleftrightarrow a \text{ pdivides } b$
(**is** $?lhs \longleftrightarrow ?rhs$)
⟨*proof*⟩

lemma *lead-coeff-carr:*

assumes $x \in \text{carrier } (\text{mult-of } P)$
shows $\text{lead-coeff } x \in \text{carrier } R - \{0\}$
⟨*proof*⟩

lemma *lead-coeff-poly-of-const:*

assumes $r \neq 0$
shows $\text{lead-coeff } (\text{poly-of-const } r) = r$
⟨*proof*⟩

lemma *lead-coeff-mult:*

assumes $f \in \text{carrier } (\text{mult-of } P)$
assumes $g \in \text{carrier } (\text{mult-of } P)$
shows $\text{lead-coeff } (f \otimes_P g) = \text{lead-coeff } f \otimes \text{lead-coeff } g$
⟨*proof*⟩

lemma *monic-poly-carr:*

assumes *monic-poly R f*
shows $f \in \text{carrier } P$
⟨*proof*⟩

lemma *monic-poly-add-distinct:*

assumes *monic-poly R f*
assumes $g \in \text{carrier } P \text{ degree } g < \text{degree } f$
shows *monic-poly R (f ⊕_P g)*
⟨*proof*⟩

lemma *monic-poly-one: monic-poly R 1_P*

⟨*proof*⟩

lemma *monic-poly-var: monic-poly R X*

⟨*proof*⟩

lemma *monic-poly-carr-2:*

assumes *monic-poly R f*
shows $f \in \text{carrier } (\text{mult-of } P)$

$\langle \text{proof} \rangle$

lemma *monic-poly-mult*:
 assumes *monic-poly* R f
 assumes *monic-poly* R g
 shows *monic-poly* R $(f \otimes_P g)$
 $\langle \text{proof} \rangle$

lemma *monic-poly-pow*:
 assumes *monic-poly* R f
 shows *monic-poly* R $(f [\wedge]_P (n::\text{nat}))$
 $\langle \text{proof} \rangle$

lemma *monic-poly-prod*:
 assumes *finite* A
 assumes $\bigwedge x. x \in A \implies \text{monic-poly } R (f x)$
 shows *monic-poly* R $(\text{finprod } P f A)$
 $\langle \text{proof} \rangle$

lemma *monic-poly-not-assoc*:
 assumes *monic-poly* R f
 assumes *monic-poly* R g
 assumes $f \sim_{(\text{mult-of } P)} g$
 shows $f = g$
 $\langle \text{proof} \rangle$

lemma *monic-poly-span*:
 assumes $x \in \text{carrier } (\text{mult-of } P) \text{ irreducible } (\text{mult-of } P) x$
 shows $\exists y. \text{monic-irreducible-poly } R y \wedge x \sim_{(\text{mult-of } P)} y$
 $\langle \text{proof} \rangle$

lemma *monic-polys-are-canonical-irreducibles*:
 canonical-irreducibles $(\text{mult-of } P) \{d. \text{monic-irreducible-poly } R d\}$
 (is *canonical-irreducibles* $(\text{mult-of } P) ?S$
 $\langle \text{proof} \rangle$

lemma
 assumes *monic-poly* R a
 shows *factor-monic-poly*:
 $a = (\bigotimes_{p \in d} p) \{d. \text{monic-irreducible-poly } R d \wedge \text{pmult } d a > 0\}.$
 $d [\wedge]_P \text{pmult } d a$ **(is** *?lhs = ?rhs*)
 and *factor-monic-poly-fin*:
 finite $\{d. \text{monic-irreducible-poly } R d \wedge \text{pmult } d a > 0\}$
 $\langle \text{proof} \rangle$

lemma *degree-monic-poly'*:
 assumes *monic-poly* R f
 shows
 $\text{sum}' (\lambda d. \text{pmult } d f * \text{degree } d) \{d. \text{monic-irreducible-poly } R d\} =$

degree f
⟨proof⟩

lemma *monic-poly-min-degree*:
assumes *monic-irreducible-poly* R f
shows $\text{degree } f \geq 1$
⟨proof⟩

lemma *degree-one-monic-poly*:
 $\text{monic-irreducible-poly } R \ f \wedge \text{degree } f = 1 \longleftrightarrow$
 $(\exists x \in \text{carrier } R. f = [\mathbf{1}, \ominus x])$
⟨proof⟩

lemma *multiplicity-ge-iff*:
assumes *monic-irreducible-poly* R d
assumes $f \in \text{carrier } P - \{\mathbf{0}_P\}$
shows $\text{pmult } d \ f \geq k \longleftrightarrow d \ [\bigwedge]_P \ k \ \text{pdivides } f$
⟨proof⟩

lemma *multiplicity-ge-1-iff-pdivides*:
assumes *monic-irreducible-poly* R d $f \in \text{carrier } P - \{\mathbf{0}_P\}$
shows $\text{pmult } d \ f \geq 1 \longleftrightarrow d \ \text{pdivides } f$
⟨proof⟩

lemma *divides-monic-poly*:
assumes *monic-poly* R f *monic-poly* R g
assumes $\bigwedge d. \text{monic-irreducible-poly } R \ d$
 $\implies \text{pmult } d \ f \leq \text{pmult } d \ g$
shows $f \ \text{pdivides } g$
⟨proof⟩

end

lemma *monic-poly-hom*:
assumes *monic-poly* R f
assumes $h \in \text{ring-iso } R \ S \ \text{domain } R \ \text{domain } S$
shows *monic-poly* S $(\text{map } h \ f)$
⟨proof⟩

lemma *monic-irreducible-poly-hom*:
assumes *monic-irreducible-poly* R f
assumes $h \in \text{ring-iso } R \ S \ \text{domain } R \ \text{domain } S$
shows *monic-irreducible-poly* S $(\text{map } h \ f)$
⟨proof⟩

end

6 Counting Irreducible Polynomials

6.1 The polynomial $X^n - X$

theory *Card-Irreducible-Polynomials-Aux*

imports

HOL-Algebra.Multiplicative-Group

Formal-Polynomial-Derivatives

Monic-Polynomial-Factorization

begin

lemma (*in domain*)

assumes *subfield* $K R$

assumes $f \in \text{carrier } (K[X])$ *degree* $f > 0$

shows *embed-inj*: *inj-on* (*rupture-surj* $K f \circ \text{poly-of-const}$) K

and *rupture-order*: *order* (*Rupt* $K f$) = *card* $K^{\wedge \text{degree } f}$

and *rupture-char*: *char* (*Rupt* $K f$) = *char* R

<proof>

definition *gauss-poly* **where**

gauss-poly $K n = X_K [\bigwedge_{\text{poly-ring } K} (n::\text{nat}) \ominus_{\text{poly-ring } K} X_K$

context *field*

begin

interpretation *polynomial-ring* R *carrier* R

<proof>

The following lemma can be found in Ireland and Rosen [3, §7.1, Lemma 2].

lemma *gauss-poly-div-gauss-poly-iff-1*:

fixes $l m :: \text{nat}$

assumes $l > 0$

shows $(X [\bigwedge_P l \ominus_P \mathbf{1}_P]) \text{ pdivides } (X [\bigwedge_P m \ominus_P \mathbf{1}_P]) \iff l \text{ dvd } m$

(*is ?lhs* \iff *?rhs*)

<proof>

lemma *gauss-poly-factor*:

assumes $n > 0$

shows *gauss-poly* $R n = (X [\bigwedge_P (n-1) \ominus_P \mathbf{1}_P] \otimes_P X$ (*is* $- =$ *?rhs*)

<proof>

lemma *var-neq-zero*: $X \neq \mathbf{0}_P$

<proof>

lemma *var-pow-eq-one-iff*: $X [\bigwedge_P k = \mathbf{1}_P \iff k = (0::\text{nat})$

<proof>

lemma *gauss-poly-carr*: *gauss-poly* $R n \in \text{carrier } P$

<proof>

lemma *gauss-poly-degree*:
assumes $n > 1$
shows $\text{degree } (\text{gauss-poly } R \ n) = n$
 $\langle \text{proof} \rangle$

lemma *gauss-poly-not-zero*:
assumes $n > 1$
shows $\text{gauss-poly } R \ n \neq \mathbf{0}_P$
 $\langle \text{proof} \rangle$

lemma *gauss-poly-monic*:
assumes $n > 1$
shows $\text{monic-poly } R \ (\text{gauss-poly } R \ n)$
 $\langle \text{proof} \rangle$

lemma *geom-nat*:
fixes $q :: \text{nat}$
fixes $x :: - :: \{\text{comm-ring}, \text{monoid-mult}\}$
shows $(x-1) * (\sum i \in \{..<q\}. x^i) = x^q - 1$
 $\langle \text{proof} \rangle$

The following lemma can be found in Ireland and Rosen [3, §7.1, Lemma 3].

lemma *gauss-poly-div-gauss-poly-iff-2*:
fixes $a :: \text{int}$
fixes $l \ m :: \text{nat}$
assumes $l > 0 \ a > 1$
shows $(a^l - 1) \text{ dvd } (a^m - 1) \iff l \text{ dvd } m$
(is ?lhs \iff ?rhs)
 $\langle \text{proof} \rangle$

lemma *gauss-poly-div-gauss-poly-iff*:
assumes $m > 0 \ n > 0 \ a > 1$
shows $\text{gauss-poly } R \ (a^n) \text{ pdivides}_R \ \text{gauss-poly } R \ (a^m)$
 $\iff n \text{ dvd } m$ **(is ?lhs=?rhs)**
 $\langle \text{proof} \rangle$

end

context *finite-field*
begin

interpretation *polynomial-ring* R *carrier* R
 $\langle \text{proof} \rangle$

lemma *div-gauss-poly-iff*:
assumes $n > 0$
assumes *monic-irreducible-poly* $R \ f$

shows $f \text{ pdivides}_R \text{ gauss-poly } R \text{ (order } R \hat{n}) \longleftrightarrow \text{degree } f \text{ dvd } n$
 ⟨proof⟩

lemma *gauss-poly-splitted*:
splitted (gauss-poly R (order R))
 ⟨proof⟩

The following lemma, for the case when R is a simple prime field, can be found in Ireland and Rosen [3, §7.1, Theorem 2]. Here the result is verified even for arbitrary finite fields.

lemma *multiplicity-of-factor-of-gauss-poly*:
assumes $n > 0$
assumes *monic-irreducible-poly R f*
shows
 $\text{pmult}_R f \text{ (gauss-poly } R \text{ (order } R \hat{n})) = \text{of-bool (degree } f \text{ dvd } n)$
 ⟨proof⟩

The following lemma, for the case when R is a simple prime field, can be found in Ireland and Rosen [3, §7.1, Corollary 1]. Here the result is verified even for arbitrary finite fields.

lemma *card-irred-aux*:
assumes $n > 0$
shows $\text{order } R \hat{n} = (\sum d \mid d \text{ dvd } n. d * \text{card } \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = d\})$
 (is ?lhs = ?rhs)
 ⟨proof⟩

end

end

6.2 Gauss Formula

theory *Card-Irreducible-Polynomials*
imports
Dirichlet-Series.Moebius-Mu
Card-Irreducible-Polynomials-Aux
begin

hide-const *Polynomial.order*

The following theorem is a slightly generalized form of the formula discovered by Gauss for the number of monic irreducible polynomials over a finite field. He originally verified the result for the case when R is a simple prime field. The version of the formula here for the case where R may be an arbitrary finite field can be found in Chebolu and Mináč [1].

theorem (in *finite-field*) *card-irred*:

assumes $n > 0$
shows $n * \text{card} \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = n\} =$
 $(\sum d \mid d \text{ dvd } n. \text{moebius-mu } d * (\text{order } R \wedge (n \text{ div } d)))$
(is ?lhs = ?rhs)
 <proof>

In the following an explicit analytic lower bound for the cardinality of monic irreducible polynomials is shown, with which existence follows. This part deviates from the classic approach, where existence is verified using a divisibility argument. The reason for the deviation is that an analytic bound can also be used to estimate the runtime of a randomized algorithm selecting an irreducible polynomial, by randomly sampling monic polynomials.

lemma (in finite-field) card-irred-1:
 $\text{card} \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = 1\} = \text{order } R$
 <proof>

lemma (in finite-field) card-irred-2:
 $\text{real} (\text{card} \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = 2\}) =$
 $(\text{real} (\text{order } R)^2 - \text{order } R) / 2$
 <proof>

lemma (in finite-field) card-irred-gt-2:
assumes $n > 2$
shows $\text{real} (\text{order } R)^n / (2 * \text{real } n) \leq$
 $\text{card} \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = n\}$
(is ?lhs \leq ?rhs)
 <proof>

lemma (in finite-field) card-irred-gt-0:
assumes $d > 0$
shows $\text{real} (\text{order } R)^d / (2 * \text{real } d) \leq \text{real} (\text{card} \{f. \text{monic-irreducible-poly}$
 $R f \wedge \text{degree } f = d\})$
(is ?L \leq ?R)
 <proof>

lemma (in finite-field) exist-irred:
assumes $n > 0$
obtains f **where** $\text{monic-irreducible-poly } R f \wedge \text{degree } f = n$
 <proof>

theorem existence:
assumes $n > 0$
assumes *Factorial-Ring.prime* p
shows $\exists (F:: \text{int set list set ring}). \text{finite-field } F \wedge \text{order } F = p^n$
 <proof>

end

7 Isomorphism between Finite Fields

theory *Finite-Fields-Isomorphic*

imports

Card-Irreducible-Polynomials

begin

lemma (in *finite-field*) *eval-on-root-is-iso*:

defines $p \equiv \text{char } R$

assumes $f \in \text{carrier } (\text{poly-ring } (\text{ZFact } p))$

assumes $\text{pirreducible}_{(\text{ZFact } p)} (\text{carrier } (\text{ZFact } p)) f$

assumes $\text{order } R = p^{\text{degree } f}$

assumes $x \in \text{carrier } R$

assumes $\text{eval } (\text{map } (\text{char-iso } R) f) x = \mathbf{0}$

shows $\text{ring-hom-ring } (\text{Rupt}_{(\text{ZFact } p)} (\text{carrier } (\text{ZFact } p)) f) R$

$(\lambda g. \text{the-elem } ((\lambda g'. \text{eval } (\text{map } (\text{char-iso } R) g') x) 'g))$

$\langle \text{proof} \rangle$

lemma (in *domain*) *pdivides-consistent*:

assumes $\text{subfield } K R f \in \text{carrier } (K[X]) g \in \text{carrier } (K[X])$

shows $f \text{ pdivides } g \iff f \text{ pdivides }_R (\text{carrier } := K) g$

$\langle \text{proof} \rangle$

lemma (in *finite-field*) *find-root*:

assumes $\text{subfield } K R$

assumes $\text{monic-irreducible-poly } (R (\text{carrier } := K)) f$

assumes $\text{order } R = \text{card } K^{\text{degree } f}$

obtains x where $\text{eval } f x = \mathbf{0} x \in \text{carrier } R$

$\langle \text{proof} \rangle$

lemma (in *finite-field*) *find-iso-from-zfact*:

defines $p \equiv \text{int } (\text{char } R)$

assumes $\text{monic-irreducible-poly } (\text{ZFact } p) f$

assumes $\text{order } R = \text{char } R^{\text{degree } f}$

shows $\exists \varphi. \varphi \in \text{ring-iso } (\text{Rupt}_{(\text{ZFact } p)} (\text{carrier } (\text{ZFact } p)) f) R$

$\langle \text{proof} \rangle$

theorem *uniqueness*:

assumes $\text{finite-field } F_1$

assumes $\text{finite-field } F_2$

assumes $\text{order } F_1 = \text{order } F_2$

shows $F_1 \simeq F_2$

$\langle \text{proof} \rangle$

end

8 Rabin's test for irreducible polynomials

theory *Rabin-Irreducibility-Test*
imports *Card-Irreducible-Polynomials-Aux*
begin

This section introduces an effective test for irreducibility of polynomials (in finite fields) based on Rabin [5].

definition *pcoprime* :: $- \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow \text{bool}$ ($\langle \text{pcoprime} \rangle$)
where *pcoprime*_R *p q* =
 $(\forall r \in \text{carrier } (\text{poly-ring } R). r \text{ pdivides}_R p \wedge r \text{ pdivides}_R q \longrightarrow \text{degree } r = 0)$

lemma *pcoprimeI*:
assumes $\bigwedge r. r \in \text{carrier } (\text{poly-ring } R) \Longrightarrow r \text{ pdivides}_R p \Longrightarrow r \text{ pdivides}_R q \Longrightarrow \text{degree } r = 0$
shows *pcoprime*_R *p q*
 $\langle \text{proof} \rangle$

context *field*
begin

interpretation *r:polynomial-ring R (carrier R)*
 $\langle \text{proof} \rangle$

lemma *pcoprime-one*: *pcoprime*_R *p 1*_{poly-ring R}
 $\langle \text{proof} \rangle$

lemma *pcoprime-left-factor*:
assumes $x \in \text{carrier } (\text{poly-ring } R)$
assumes $y \in \text{carrier } (\text{poly-ring } R)$
assumes $z \in \text{carrier } (\text{poly-ring } R)$
assumes *pcoprime*_R $(x \otimes_{\text{poly-ring } R} y) z$
shows *pcoprime*_R $x z$
 $\langle \text{proof} \rangle$

lemma *pcoprime-sym*:
shows *pcoprime* $x y = \text{pcoprime } y x$
 $\langle \text{proof} \rangle$

lemma *pcoprime-left-assoc-cong-aux*:
assumes $x1 \in \text{carrier } (\text{poly-ring } R) x2 \in \text{carrier } (\text{poly-ring } R)$
assumes $x2 \sim_{\text{poly-ring } R} x1$
assumes $y \in \text{carrier } (\text{poly-ring } R)$
assumes *pcoprime* $x1 y$
shows *pcoprime* $x2 y$
 $\langle \text{proof} \rangle$

lemma *pcoprime-left-assoc-cong*:

assumes $x1 \in \text{carrier } (\text{poly-ring } R) \ x2 \in \text{carrier } (\text{poly-ring } R)$
assumes $x1 \sim_{\text{poly-ring } R} x2$
assumes $y \in \text{carrier } (\text{poly-ring } R)$
shows $\text{pcoprime } x1 \ y = \text{pcoprime } x2 \ y$
 <proof>

lemma *pcoprime-right-assoc-cong*:
assumes $x1 \in \text{carrier } (\text{poly-ring } R) \ x2 \in \text{carrier } (\text{poly-ring } R)$
assumes $x1 \sim_{\text{poly-ring } R} x2$
assumes $y \in \text{carrier } (\text{poly-ring } R)$
shows $\text{pcoprime } y \ x1 = \text{pcoprime } y \ x2$
 <proof>

lemma *pcoprime-step*:
assumes $f \in \text{carrier } (\text{poly-ring } R)$
assumes $g \in \text{carrier } (\text{poly-ring } R)$
shows $\text{pcoprime } f \ g \longleftrightarrow \text{pcoprime } g \ (f \text{ pmod } g)$
 <proof>

lemma *pcoprime-zero-iff*:
assumes $f \in \text{carrier } (\text{poly-ring } R)$
shows $\text{pcoprime } f \ [] \longleftrightarrow \text{length } f = 1$
 <proof>

end

context *finite-field*
begin

interpretation *r:polynomial-ring R (carrier R)*
 <proof>

lemma *exists-irreducible-proper-factor*:
assumes $\text{monic-poly } R \ f \ \text{degree } f > 0 \ \neg \text{monic-irreducible-poly } R \ f$
shows $\exists g. \text{monic-irreducible-poly } R \ g \wedge g \text{ pdivides}_R f \wedge \text{degree } g < \text{degree } f$
 <proof>

theorem *rabin-irreducibility-condition*:
assumes $\text{monic-poly } R \ f \ \text{degree } f > 0$
defines $N \equiv \{\text{degree } f \ \text{div } p \mid p . \text{Factorial-Ring.prime } p \wedge p \ \text{dvd } \text{degree } f\}$
shows $\text{monic-irreducible-poly } R \ f \longleftrightarrow$
 $(f \text{ pdivides } \text{gauss-poly } R \ (\text{order } R \widehat{\text{degree } f}) \wedge (\forall n \in N. \text{pcoprime } (\text{gauss-poly } R \ (\text{order } R \widehat{n}) \ f))$
 $(\text{is } ?L \longleftrightarrow ?R1 \wedge ?R2)$
 <proof>

A more general variant of the previous theorem for non-monic

polynomials. The result is from Lemma 1 [5].

theorem *rabin-irreducibility-condition-2:*

assumes $f \in \text{carrier } (\text{poly-ring } R) \text{ degree } f > 0$

defines $N \equiv \{\text{degree } f \text{ div } p \mid p . \text{Factorial-Ring.prime } p \wedge p \text{ dvd degree } f\}$

shows $\text{pirreducible } (\text{carrier } R) f \longleftrightarrow$

$(f \text{ pdivides gauss-poly } R (\text{order } R \hat{\text{degree}} f) \wedge (\forall n \in N. \text{pcoprime } (\text{gauss-poly } R (\text{order } R \hat{n})) f))$

(is $?L \longleftrightarrow ?R1 \wedge ?R2)$

<proof>

end

end

9 Executable Structures

theory *Finite-Fields-Indexed-Algebra-Code*

imports *HOL-Algebra.Ring HOL-Algebra.Coset*

begin

In the following, we introduce records for executable operations for algebraic structures, which can be used for code-generation and evaluation. These are then shown to be equivalent to the (not-necessarily constructive) definitions using *HOL-Algebra*. A more direct approach, i.e., instantiating the structures in the framework with effective operations fails. For example the structure records represent the domain of the algebraic structure as a set, which implies the evaluation of $(\oplus_{\text{residue-ring } (10::'c)}^{100})$ requires the construction of $\{0..(10::'a)^{100} - 1\}$. This is technically constructive but very impractical. Moreover, the additive/multiplicative inverse is defined non-constructively using the description operator **THE** in *HOL-Algebra*.

The above could be avoided, if it were possible to introduce code equations conditionally, e.g., for example for $(\ominus_{\text{residue-ring } n} x) y$ (if $x y$ are in the carrier of the structure, but this does not seem to be possible).

Note that, the algebraic structures defined in *HOL-Computational_Algebra* are type-based, which prevents using them in some algorithmic settings. For example, choosing an irreducible polynomial dynamically and performing operations in the factoring ring with respect to it is not possible in the type-based approach.

record *'a idx-ring =*
idx-pred :: 'a \Rightarrow bool
idx-uminus :: 'a \Rightarrow 'a

```

idx-plus :: 'a ⇒ 'a ⇒ 'a
idx-udivide :: 'a ⇒ 'a
idx-mult :: 'a ⇒ 'a ⇒ 'a
idx-zero :: 'a
idx-one :: 'a

```

```

record 'a idx-ring-enum = 'a idx-ring +
  idx-size :: nat
  idx-enum :: nat ⇒ 'a
  idx-enum-inv :: 'a ⇒ nat

```

```

fun idx-pow :: ('a,'b) idx-ring-scheme ⇒ 'a ⇒ nat ⇒ 'a where
  idx-pow E x 0 = idx-one E |
  idx-pow E x (Suc n) = idx-mult E (idx-pow E x n) x

```

open-bundle *index-algebra-syntax*

begin

```

notation idx-zero (⟨0C1⟩)
notation idx-one (⟨1C1⟩)
notation idx-plus (infixl ⟨+C1⟩ 65)
notation idx-mult (infixl ⟨*C1⟩ 70)
notation idx-uminus (⟨-C1 -> [81] 80)
notation idx-udivide (⟨·-1C1⟩ [81] 80)
notation idx-pow (infixr ⟨^C1⟩ 75)

```

end

definition *ring-of* :: ('a,'b) idx-ring-scheme ⇒ 'a ring

```

where ring-of A = ⟨
  carrier = {x. idx-pred A x},
  mult = (λ x y. x *CA y),
  one = 1CA,
  zero = 0CA,
  add = (λ x y. x +CA y) ⟩

```

definition *ring_C* **where**

```

ringC A = (ring (ring-of A) ∧ (∀ x. idx-pred A x ⟶ -CA x =
  ⊖ring-of A x) ∧
  (∀ x. x ∈ Units (ring-of A) ⟶ x-1CA = invring-of A x))

```

lemma *ring-cD-aux*:

```

x ^CA n = x [ ^ring-of A n
⟨proof⟩

```

lemma *ring-cD*:

assumes *ring_C* A

shows

```

0CA = 0ring-of A
1CA = 1ring-of A
∧ x y. x *CA y = x ⊗ring-of A y

```

$\bigwedge x y. x +_{CA} y = x \oplus_{ring-of A} y$
 $\bigwedge x. x \in carrier (ring-of A) \implies -_{CA} x = \ominus_{ring-of A} x$
 $\bigwedge x. x \in Units (ring-of A) \implies x^{-1}_{CA} = inv_{ring-of A} x$
 $\bigwedge x. x \wedge_{CA} n = x \lceil_{ring-of A} n$
 <proof>

lemma ring-cI:

assumes ring (ring-of A)
assumes $\bigwedge x. x \in carrier (ring-of A) \implies -_{CA} x = \ominus_{ring-of A} x$
assumes $\bigwedge x. x \in Units (ring-of A) \implies x^{-1}_{CA} = inv_{ring-of A} x$
shows ring_C A
 <proof>

definition cring_C where cring_C A = (ring_C A \wedge cring (ring-of A))

lemma cring-cI:

assumes cring (ring-of A)
assumes $\bigwedge x. x \in carrier (ring-of A) \implies -_{CA} x = \ominus_{ring-of A} x$
assumes $\bigwedge x. x \in Units (ring-of A) \implies x^{-1}_{CA} = inv_{ring-of A} x$
shows cring_C A
 <proof>

lemma cring-c-imp-ring: cring_C A \implies ring_C A

<proof>

lemmas cring-cD = ring-cD[OF cring-c-imp-ring]

definition domain_C where domain_C A = (cring_C A \wedge domain (ring-of A))

lemma domain-cI:

assumes domain (ring-of A)
assumes $\bigwedge x. x \in carrier (ring-of A) \implies -_{CA} x = \ominus_{ring-of A} x$
assumes $\bigwedge x. x \in Units (ring-of A) \implies x^{-1}_{CA} = inv_{ring-of A} x$
shows domain_C A
 <proof>

lemma domain-c-imp-ring: domain_C A \implies ring_C A

<proof>

lemmas domain-cD = ring-cD[OF domain-c-imp-ring]

definition field_C where field_C A = (domain_C A \wedge field (ring-of A))

lemma field-cI:

assumes field (ring-of A)
assumes $\bigwedge x. x \in carrier (ring-of A) \implies -_{CA} x = \ominus_{ring-of A} x$
assumes $\bigwedge x. x \in Units (ring-of A) \implies x^{-1}_{CA} = inv_{ring-of A} x$

shows $field_C A$
 $\langle proof \rangle$

lemma $field-c-imp-ring$: $field_C A \implies ring_C A$
 $\langle proof \rangle$

lemmas $field-cD = ring-cD[OF field-c-imp-ring]$

definition $enum_C$ **where** $enum_C A = ($
 $finite (carrier (ring-of A)) \wedge$
 $idx-size A = order (ring-of A) \wedge$
 $bij-betw (idx-enum A) \{..<order (ring-of A)\} (carrier (ring-of A)) \wedge$
 $(\forall x < order (ring-of A). idx-enum-inv A (idx-enum A x) = x)$
 $)$

lemma $enum-cI$:
 assumes $finite (carrier (ring-of A))$
 assumes $idx-size A = order (ring-of A)$
 assumes $bij-betw (idx-enum A) \{..<order (ring-of A)\} (carrier (ring-of A))$
 assumes $\bigwedge x. x < order (ring-of A) \implies idx-enum-inv A (idx-enum A x) = x$
 shows $enum_C A$
 $\langle proof \rangle$

lemma $enum-cD$:
 assumes $enum_C R$
 shows $finite (carrier (ring-of R))$
 and $idx-size R = order (ring-of R)$
 and $bij-betw (idx-enum R) \{..<order (ring-of R)\} (carrier (ring-of R))$
 and $bij-betw (idx-enum-inv R) (carrier (ring-of R)) \{..<order (ring-of R)\}$
 and $\bigwedge x. x < order (ring-of R) \implies idx-enum-inv R (idx-enum R x) = x$
 and $\bigwedge x. x \in carrier (ring-of R) \implies idx-enum R (idx-enum-inv R x) = x$
 $\langle proof \rangle$

end

10 Executable Polynomial Rings

theory $Finite-Fields-Poly-Ring-Code$
 imports
 $Finite-Fields-Indexed-Algebra-Code$
 $HOL-Algebra.Polynomials$
 $Finite-Fields.Card-Irreducible-Polynomials-Aux$
begin

fun *o-normalize* :: ('a,'b) *idx-ring-scheme* ⇒ 'a list ⇒ 'a list
where
o-normalize E [] = []
| *o-normalize* E p = (if lead-coeff p ≠ 0_{C E} then p else *o-normalize* E (tl p))

fun *o-poly-add* :: ('a,'b) *idx-ring-scheme* ⇒ 'a list ⇒ 'a list ⇒ 'a list
where
o-poly-add E p1 p2 = (
if length p1 ≥ length p2
then *o-normalize* E (map2 (*idx-plus* E) p1 ((replicate (length p1
– length p2) 0_{C E}) @ p2))
else *o-poly-add* E p2 p1)

fun *o-poly-mult* :: ('a,'b) *idx-ring-scheme* ⇒ 'a list ⇒ 'a list ⇒ 'a list
where
o-poly-mult E [] p2 = []
| *o-poly-mult* E p1 p2 =
o-poly-add E ((map (*idx-mult* E (hd p1)) p2) @
(replicate (degree p1) 0_{C E})) (*o-poly-mult* E (tl p1) p2)

definition *poly* :: ('a,'b) *idx-ring-scheme* ⇒ 'a list *idx-ring*
where *poly* E = (
idx-pred = (λx. (x = [] ∨ hd x ≠ 0_{C E}) ∧ list-all (*idx-pred* E) x),
idx-uminus = (λx. map (*idx-uminus* E) x),
idx-plus = *o-poly-add* E,
idx-udivide = (λx. [*idx-udivide* E (hd x)]),
idx-mult = *o-poly-mult* E,
idx-zero = [],
idx-one = [*idx-one* E])

definition *poly-var* :: ('a,'b) *idx-ring-scheme* ⇒ 'a list (⟨X_{C1}⟩)
where *poly-var* E = [*idx-one* E, *idx-zero* E]

lemma *poly-var*: *poly-var* R = X_{ring-of} R
⟨proof⟩

fun *poly-eval* :: ('a,'b) *idx-ring-scheme* ⇒ 'a list ⇒ 'a ⇒ 'a
where *poly-eval* R fs x = fold (λa b. b *_{C R} x +_{C R} a) fs 0_{C R}

lemma *ring-of-poly*:
assumes *ring_C* A
shows *ring-of* (*poly* A) = *poly-ring* (*ring-of* A)
⟨proof⟩

lemma *poly-eval*:
assumes *ring_C* R

assumes $fsc:fs \in carrier (ring-of (poly R))$ **and** $xc:x \in carrier (ring-of R)$
shows $poly-eval R fs x = ring.eval (ring-of R) fs x$
 $\langle proof \rangle$

lemma *poly-domain*:
assumes $domain_C A$
shows $domain_C (poly A)$
 $\langle proof \rangle$

function $long-division_C :: ('a,'b) idx-ring-scheme \Rightarrow 'a list \Rightarrow 'a list \Rightarrow 'a list \times 'a list$
where $long-division_C F f g = ($
 if $(length g = 0 \vee length f < length g)$
 then $([], f)$
 else (
 let $k = length f - length g;$
 $\alpha = -_C F (hd f *_C F (hd g)^{-1}_C F);$
 $h = [\alpha] *_C poly F X_C F \widehat{C} poly F k;$
 $f' = f +_C poly F (h *_C poly F g);$
 $f'' = take (length f - 1) f'$
 in $apfst (\lambda x. x +_C poly F -_C poly F h) (long-division_C F f'' g))$
 $\langle proof \rangle$

lemma *pmod-termination-helper*:
 $g \neq [] \implies \neg length f < length g \implies \min x (length f - 1) < length f$
 $\langle proof \rangle$

termination $\langle proof \rangle$

declare $long-division_C.simps[simp del]$

lemma *long-division-c-length*:
assumes $length g > 0$
shows $length (snd (long-division_C R f g)) < length g$
 $\langle proof \rangle$

context *field*
begin

interpretation $r:polynomial-ring R (carrier R)$
 $\langle proof \rangle$

lemma *poly-length-from-coeff*:
assumes $p \in carrier (poly-ring R)$
assumes $\bigwedge i. i \geq k \implies coeff p i = 0$
shows $length p \leq k$
 $\langle proof \rangle$

lemma *poly-add-cancel-len*:

assumes $f \in \text{carrier } (\text{poly-ring } R) - \{\mathbf{0}_{\text{poly-ring } R}\}$
assumes $g \in \text{carrier } (\text{poly-ring } R) - \{\mathbf{0}_{\text{poly-ring } R}\}$
assumes $\text{hd } f = \ominus \text{hd } g \text{ degree } f = \text{degree } g$
shows $\text{length } (f \oplus_{\text{poly-ring } R} g) < \text{length } f$

<proof>

lemma *pmod-mult-left*:

assumes $f \in \text{carrier } (\text{poly-ring } R)$
assumes $g \in \text{carrier } (\text{poly-ring } R)$
assumes $h \in \text{carrier } (\text{poly-ring } R)$
shows $(f \otimes_{\text{poly-ring } R} g) \text{ pmod } h = ((f \text{ pmod } h) \otimes_{\text{poly-ring } R} g) \text{ pmod } h$
(is ?L = ?R)

<proof>

lemma *pmod-mult-right*:

assumes $f \in \text{carrier } (\text{poly-ring } R)$
assumes $g \in \text{carrier } (\text{poly-ring } R)$
assumes $h \in \text{carrier } (\text{poly-ring } R)$
shows $(f \otimes_{\text{poly-ring } R} g) \text{ pmod } h = (f \otimes_{\text{poly-ring } R} (g \text{ pmod } h)) \text{ pmod } h$
(is ?L = ?R)

<proof>

lemma *pmod-mult-both*:

assumes $f \in \text{carrier } (\text{poly-ring } R)$
assumes $g \in \text{carrier } (\text{poly-ring } R)$
assumes $h \in \text{carrier } (\text{poly-ring } R)$
shows $(f \otimes_{\text{poly-ring } R} g) \text{ pmod } h = ((f \text{ pmod } h) \otimes_{\text{poly-ring } R} (g \text{ pmod } h)) \text{ pmod } h$
(is ?L = ?R)

<proof>

lemma *field-Unit-minus-closed*:

assumes $x \in \text{Units } R$
shows $\ominus x \in \text{Units } R$

<proof>

end

lemma *long-division-c*:

assumes $\text{field}_C R$
assumes $f \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$
assumes $g \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$
shows $\text{long-division}_C R f g = (\text{ring.pdiv } (\text{ring-of } R) f g, \text{ring.pmod } (\text{ring-of } R) f g)$

<proof>

definition $\text{pdiv}_C :: ('a, 'b) \text{ idx-ring-scheme} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow 'a$

list where

$pdiv_C R f g = fst (long-division_C R f g)$

lemma *pdiv-c*:

assumes $field_C R$

assumes $f \in carrier (poly-ring (ring-of R))$

assumes $g \in carrier (poly-ring (ring-of R))$

shows $pdiv_C R f g = ring.pdiv (ring-of R) f g$

$\langle proof \rangle$

definition $pmod_C :: ('a,'b) idx-ring-scheme \Rightarrow 'a list \Rightarrow 'a list \Rightarrow 'a$

list where

$pmod_C R f g = snd (long-division_C R f g)$

lemma *pmod-c*:

assumes $field_C R$

assumes $f \in carrier (poly-ring (ring-of R))$

assumes $g \in carrier (poly-ring (ring-of R))$

shows $pmod_C R f g = ring.pmod (ring-of R) f g$

$\langle proof \rangle$

function *ext-euclidean* ::

$('a,'b) idx-ring-scheme \Rightarrow 'a list \Rightarrow 'a list \Rightarrow ('a list \times 'a list) \times 'a$

list

where $ext-euclidean F f g = ($

$if f = [] \vee g = [] then$

$((1_C poly F, 1_C poly F), f +_C poly F g)$

$else ($

$let (p,q) = long-division_C F f g;$

$((u,v),r) = ext-euclidean F g q$

$in ((v,u +_C poly F (-_C poly F (p *_C poly F v))),r))$

$\langle proof \rangle$

termination

$\langle proof \rangle$

lemma (*in domain*) *pdivides-self*:

assumes $x \in carrier (poly-ring R)$

shows $x pdivides x$

$\langle proof \rangle$

declare $ext-euclidean.simps[simp del]$

lemma *ext-euclidean*:

assumes $field_C R$

defines $P \equiv poly-ring (ring-of R)$

assumes $f \in carrier (poly-ring (ring-of R))$

assumes $g \in carrier (poly-ring (ring-of R))$

defines $r \equiv \text{ext-euclidean } R \ f \ g$
shows $\text{snd } r = f \otimes_P (\text{fst } (\text{fst } r)) \oplus_P g \otimes_P (\text{snd } (\text{fst } r))$ (**is** ?T1)
and $\text{snd } r \text{ pdivides}_{\text{ring-of } R} f$ (**is** ?T2) $\text{snd } r \text{ pdivides}_{\text{ring-of } R} g$ (**is** ?T3)
and $\{\text{snd } r, \text{fst } (\text{fst } r), \text{snd } (\text{fst } r)\} \subseteq \text{carrier } P$ (**is** ?T4)
and $\text{snd } r = [] \longrightarrow f = [] \wedge g = []$ (**is** ?T5)
 <proof>
end

11 Executable Factor Rings

theory *Finite-Fields-Mod-Ring-Code*
imports *Finite-Fields-Indexed-Algebra-Code Ring-Characteristic*
begin

definition *mod-ring* :: $\text{nat} \Rightarrow \text{nat } \text{idx-ring-enum}$
where *mod-ring* $n = ($
 $\text{idx-pred} = (\lambda x. x < n),$
 $\text{idx-uminus} = (\lambda x. (n-x) \text{ mod } n),$
 $\text{idx-plus} = (\lambda x \ y. (x+y) \text{ mod } n),$
 $\text{idx-udivide} = (\lambda x. \text{nat } (\text{fst } (\text{bezout-coefficients } (\text{int } x) (\text{int } n)) \text{ mod } (\text{int } n))),$
 $\text{idx-mult} = (\lambda x \ y. (x*y) \text{ mod } n),$
 $\text{idx-zero} = 0,$
 $\text{idx-one} = 1,$
 $\text{idx-size} = n,$
 $\text{idx-enum} = \text{id},$
 $\text{idx-enum-inv} = \text{id}$
 $)$

lemma *zfact-iso-0*:
assumes $n > 0$
shows $\text{zfact-iso } n \ 0 = \mathbf{0}_{Z\text{Fact } (\text{int } n)}$
 <proof>

lemma *zfact-prime-is-field*:
assumes *Factorial-Ring.prime* ($p :: \text{nat}$)
shows $\text{field } (Z\text{Fact } (\text{int } p))$
 <proof>

definition *zfact-iso-inv* :: $\text{nat} \Rightarrow \text{int set} \Rightarrow \text{nat}$ **where**
 $\text{zfact-iso-inv } p = \text{the-inv-into } \{..<p\} (\text{zfact-iso } p)$

lemma *zfact-iso-inv-0*:
assumes $n \text{ ge } 0: n > 0$
shows $\text{zfact-iso-inv } n \ \mathbf{0}_{Z\text{Fact } (\text{int } n)} = 0$
 <proof>

lemma *zfact-coset*:
assumes $n \geq 0$: $n > 0$
assumes $x \in \text{carrier } (ZFact \text{ (int } n))$
defines $I \equiv \text{Idl}_{\mathbb{Z}} \{ \text{int } n \}$
shows $x = I +_{\mathbb{Z}} (\text{int } (zfact\text{-iso}\text{-inv } n \ x))$
 $\langle \text{proof} \rangle$

lemma *zfact-iso-inv-bij*:
assumes $n > 0$
shows $\text{bij_betw } (zfact\text{-iso}\text{-inv } n) (\text{carrier } (ZFact \text{ (int } n))) (\text{carrier } (\text{ring-of } (\text{mod-ring } n)))$
 $\langle \text{proof} \rangle$

lemma *zfact-iso-inv-is-ring-iso*:
fixes $n :: \text{nat}$
assumes $n \geq 1$: $n > 1$
shows $zfact\text{-iso}\text{-inv } n \in \text{ring-iso } (ZFact \text{ (int } n)) (\text{ring-of } (\text{mod-ring } n))$ **(is ?f \in -)**
 $\langle \text{proof} \rangle$

lemma *mod-ring-finite*:
 $\text{finite } (\text{carrier } (\text{ring-of } (\text{mod-ring } n)))$
 $\langle \text{proof} \rangle$

lemma *mod-ring-carr*:
 $x \in \text{carrier } (\text{ring-of } (\text{mod-ring } n)) \iff x < n$
 $\langle \text{proof} \rangle$

lemma *mod-ring-is-crng*:
assumes $n \geq 1$: $n > 1$
shows $\text{crng } (\text{ring-of } (\text{mod-ring } n))$
 $\langle \text{proof} \rangle$

lemma *zfact-iso-is-ring-iso*:
assumes $n \geq 1$: $n > 1$
shows $zfact\text{-iso } n \in \text{ring-iso } (\text{ring-of } (\text{mod-ring } n)) (ZFact \text{ (int } n))$
 $\langle \text{proof} \rangle$

If p is a prime than *mod-ring* p is a field:

lemma *mod-ring-is-field*:
assumes *Factorial-Ring.prime* p
shows $\text{field } (\text{ring-of } (\text{mod-ring } p))$
 $\langle \text{proof} \rangle$

lemma *mod-ring-is-ring-c*:
assumes $n > 1$
shows $\text{crng}_C (\text{mod-ring } n)$
 $\langle \text{proof} \rangle$

```

lemma mod-ring-is-field-c:
  assumes Factorial-Ring.prime p
  shows fieldC (mod-ring p)
  ⟨proof⟩

```

```

lemma mod-ring-is-enum-c:
  shows enumC (mod-ring n)
  ⟨proof⟩

```

end

12 Executable Code for Rabin's Irreducibility Test

```

theory Rabin-Irreducibility-Test-Code
imports
  Finite-Fields-Poly-Ring-Code
  Finite-Fields-Mod-Ring-Code
  Rabin-Irreducibility-Test
begin

```

```

fun pcoprimeC :: ('a, 'b) idx-ring-scheme ⇒ 'a list ⇒ 'a list ⇒ bool
  where pcoprimeC R f g = (length (snd (ext-euclidean R f g)) = 1)

```

```

declare pcoprimeC.simps[simp del]

```

```

lemma pcoprime-c:
  assumes fieldC R
  assumes f ∈ carrier (poly-ring (ring-of R))
  assumes g ∈ carrier (poly-ring (ring-of R))
  shows pcoprimeC R f g ⇔ pcoprimering-of R f g (is ?L = ?R)
  ⟨proof⟩

```

The following is a fast version of *pmod* for polynomials (to a high power) that need to be reduced, this is used for the higher order term of the Gauss polynomial.

```

fun pmod-powC :: ('a, 'b) idx-ring-scheme ⇒ 'a list ⇒ nat ⇒ 'a list
  ⇒ 'a list
  where pmod-powC F f n g = (
    let r = (if n ≥ 2 then pmod-powC F f (n div 2) g  $\widehat{C}_{poly F 2}$  else
       $1_{C poly F}$ )
    in pmodC F (r *C poly F (f  $\widehat{C}_{poly F}$  (n mod 2))) g)

```

```

declare pmod-powC.simps[simp del]

```

```

lemma pmod-pow-c:
  assumes fieldC R

```

assumes $f \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$
assumes $g \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$
shows $\text{pmod-pow}_C R f n g = \text{ring.pmod } (\text{ring-of } R) (f [\wedge]_{\text{poly-ring } (\text{ring-of } R)} n) g$
 <proof>

The following function checks whether a given polynomial is co-prime with the Gauss polynomial $X^n - X$.

definition $\text{pcoprime-with-gauss-poly} :: ('a, 'b) \text{idx-ring-scheme} \Rightarrow 'a \text{ list} \Rightarrow \text{nat} \Rightarrow \text{bool}$
where $\text{pcoprime-with-gauss-poly } F p n =$
 $(\text{pcoprime}_C F p (\text{pmod-pow}_C F X_C F n p +_C \text{poly } F (-_C \text{poly } F \text{pmod}_C F X_C F p)))$

definition $\text{divides-gauss-poly} :: ('a, 'b) \text{idx-ring-scheme} \Rightarrow 'a \text{ list} \Rightarrow \text{nat} \Rightarrow \text{bool}$
where $\text{divides-gauss-poly } F p n =$
 $(\text{pmod-pow}_C F X_C F n p +_C \text{poly } F (-_C \text{poly } F \text{pmod}_C F X_C F p) = \square)$

lemma mod-gauss-poly :
assumes $\text{field}_C R$
assumes $f \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$
shows $\text{pmod-pow}_C R X_C R n f +_C \text{poly } R (-_C \text{poly } R \text{pmod}_C R X_C R f) =$
 $\text{ring.pmod } (\text{ring-of } R) (\text{gauss-poly } (\text{ring-of } R) n) f \text{ (is ?L = ?R)}$
 <proof>

lemma $\text{pcoprime-with-gauss-poly}$:
assumes $\text{field}_C R$
assumes $f \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$
shows $\text{pcoprime-with-gauss-poly } R f n \longleftrightarrow \text{pcoprime}_{\text{ring-of } R} (\text{gauss-poly } (\text{ring-of } R) n) f$
 (is ?L = ?R)
 <proof>

lemma $\text{divides-gauss-poly}$:
assumes $\text{field}_C R$
assumes $f \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$
shows $\text{divides-gauss-poly } R f n \longleftrightarrow f \text{pdivides}_{\text{ring-of } R} (\text{gauss-poly } (\text{ring-of } R) n)$
 (is ?L = ?R)
 <proof>

fun $\text{rabin-test-powers} :: ('a, 'b) \text{idx-ring-enum-scheme} \Rightarrow \text{nat} \Rightarrow \text{nat list}$
where $\text{rabin-test-powers } F n =$

```

    map ( $\lambda p. \text{idx-size } F \wedge (n \text{ div } p)$ ) (filter ( $\lambda p. \text{prime } p \wedge p \text{ dvd } n$ )
[2.. $(n+1)$ ])

```

Given a monic polynomial with coefficients over a finite field returns true, if it is irreducible

```

fun rabin-test :: ('a, 'b) idx-ring-enum-scheme  $\Rightarrow$  'a list  $\Rightarrow$  bool
  where rabin-test F f = (
    if degree f = 0 then
      False
    else (if  $\neg$ divides-gauss-poly F f (idx-size F  $\wedge$  degree f) then
      False
    else (list-all (pcoprime-with-gauss-poly F f) (rabin-test-powers F
(degree f))))))

```

```

declare rabin-test.simps[simp del]

```

```

context
  fixes R
  assumes field-R: fieldC R
  assumes enum-R: enumC R
begin

```

```

interpretation finite-field (ring-of R)
  <proof>

```

```

lemma rabin-test-powers:
  assumes n > 0
  shows set (rabin-test-powers R n) =
    {order (ring-of R)  $\wedge$  (n div p) | p . Factorial-Ring.prime p  $\wedge$  p dvd
n}
  (is ?L = ?R)
  <proof>

```

```

lemma rabin-test:
  assumes monic-poly (ring-of R) f
  shows rabin-test R f  $\longleftrightarrow$  monic-irreducible-poly (ring-of R) f (is
?L = ?R)
  <proof>

```

```

end

```

```

end

```

13 Additional results about Bijections and Digit Representations

```

theory Finite-Fields-More-Bijections
  imports HOL-Library.FuncSet Digit-Expansions.Bits-Digits

```

begin

lemma *nth-digit-0*:

assumes $x < b^k$

shows $\text{nth-digit } x \ k \ b = 0$

$\langle \text{proof} \rangle$

lemma *nth-digit-bounded'*:

assumes $b > 0$

shows $\text{nth-digit } v \ x \ b < b$

$\langle \text{proof} \rangle$

lemma *digit-gen-sum-repr'*:

assumes $n < b^c$

shows $n = (\sum_{k < c} \text{nth-digit } n \ k \ b * b^k)$

$\langle \text{proof} \rangle$

lemma

assumes $\bigwedge x. x \in A \implies f (g x) = x$

shows $\bigwedge y. y \in g^{-1} A \implies g (f y) = y$

$\langle \text{proof} \rangle$

lemma *nth-digit-bij*:

bij-betw $(\lambda v. (\lambda x \in \{..<n\}. \text{nth-digit } v \ x \ b)) \{..<b^n\} (\{..<n\} \rightarrow_E \{..<b\})$

(**is** *bij-betw* ?f ?A ?B)

$\langle \text{proof} \rangle$

lemma *nth-digit-sum*:

assumes $\bigwedge i. i < l \implies f i < b$

shows $\bigwedge k. k < l \implies \text{nth-digit } (\sum_{i < l} f i * b^i) \ k \ b = f k$

and $(\sum_{i < l} f i * b^i) < b^l$

$\langle \text{proof} \rangle$

lemma *bij-betw-reindex*:

assumes *bij-betw* $f \ I \ J$

shows *bij-betw* $(\lambda x. \lambda i \in I. x (f i)) (J \rightarrow_E S) (I \rightarrow_E S)$

$\langle \text{proof} \rangle$

lemma *lift-bij-betw*:

assumes *bij-betw* $f \ S \ T$

shows *bij-betw* $(\lambda x. \lambda i \in I. f (x i)) (I \rightarrow_E S) (I \rightarrow_E T)$

$\langle \text{proof} \rangle$

lemma *lists-bij*:

bij-betw $(\lambda x. \text{map } x \ [\ 0..<d \]) (\{..<d\} \rightarrow_E S) \{x. \text{set } x \subseteq S \wedge \text{length } x = d\}$

$\langle \text{proof} \rangle$

lemma *bij-betw-prod*: *bij-betw* ($\lambda x. (x \text{ mod } s, x \text{ div } s)$) $\{..<s * t\}$
 $\{..<(s::\text{nat})\} \times \{..<t\}$
 $\langle \text{proof} \rangle$

end

14 Additional results about PMFs

theory *Finite-Fields-More-PMF*
imports *HOL-Probability.Probability-Mass-Function*
begin

lemma *powr-mono-rev*:
fixes $x :: \text{real}$
assumes $a \leq b$ **and** $x > 0$ $x \leq 1$
shows $x \text{ powr } b \leq x \text{ powr } a$
 $\langle \text{proof} \rangle$

lemma *integral-bind-pmf*:
fixes $f :: - \Rightarrow \text{real}$
assumes *bounded* (f ' *set-pmf* (*bind-pmf* p q))
shows $(\int x. f x \partial \text{bind-pmf } p \ q) = (\int x. \int y. f y \partial q \ x \ \partial p)$ (**is** $?L = ?R$)
 $\langle \text{proof} \rangle$

lemma *measure-bind-pmf*:
 $\text{measure } (\text{bind-pmf } m \ f) \ s = (\int x. \text{measure } (f \ x) \ s \ \partial m)$ (**is** $?L = ?R$)
 $\langle \text{proof} \rangle$

end

15 Executable Polynomial Factor Rings

theory *Finite-Fields-Poly-Factor-Ring-Code*
imports
Finite-Fields-Poly-Ring-Code
Rabin-Irreducibility-Test-Code
Finite-Fields-More-Bijections
begin

Enumeration of the polynomials with a given degree:

definition *poly-enum* $:: ('a, 'b) \text{ idx-ring-enum-scheme} \Rightarrow \text{nat} \Rightarrow \text{nat}$
 $\Rightarrow 'a \text{ list}$
where *poly-enum* $R \ l \ n =$
 $\text{dropWhile } ((=) \ 0_{CR}) \ (\text{map } (\lambda p. \text{idx-enum } R \ (\text{nth-digit } n \ (l-1-p))$
 $(\text{idx-size } R))) \ [0..<l])$

lemma *replicate-drop-while-cancel*:

assumes $k = \text{length } (\text{takeWhile } ((=) x) y)$
shows $\text{replicate } k x \text{ @ dropWhile } ((=) x) y = y \text{ (is ?L = ?R)}$
 ⟨proof⟩

lemma *arg-cong3*:
assumes $x = u \ y = v \ z = w$
shows $f x y z = f u v w$
 ⟨proof⟩

lemma *list-all-dropwhile*: $\text{list-all } p \ xs \implies \text{list-all } p \ (\text{dropWhile } q \ xs)$
 ⟨proof⟩

lemma *bij-betw-poly-enum*:
assumes $\text{enum}_C R \ \text{ring}_C R$
shows $\text{bij-betw } (\text{poly-enum } R \ l) \ \{..<\text{idx-size } R \ \wedge\}$
 $\{xs. xs \in \text{carrier } (\text{poly-ring } (\text{ring-of } R)) \wedge \text{length } xs \leq l\}$
 ⟨proof⟩

definition *poly-enum-inv* :: $('a, 'b) \ \text{idx-ring-enum-scheme} \Rightarrow \text{nat} \Rightarrow 'a \ \text{list} \Rightarrow \text{nat}$

where $\text{poly-enum-inv } R \ l \ f =$
 $(\text{let } f' = \text{replicate } (l - \text{length } f) \ 0_C R \ \text{@ } f \ \text{in}$
 $(\sum_{i < l} \ \text{idx-enum-inv } R \ (f' ! (l - 1 - i)) * \text{idx-size } R \ \hat{i} \))$

find-theorems $(\sum_{i < ?l} \ ?f \ i * \ ?x \ \hat{i}) < \ ?x \ \hat{?l}$

lemma *poly-enum-inv*:
assumes $\text{enum}_C R \ \text{ring}_C R$
assumes $x \in \{xs. xs \in \text{carrier } (\text{poly-ring } (\text{ring-of } R)) \wedge \text{length } xs \leq l\}$
shows $\text{the-inv-into } \{..<\text{idx-size } R \ \wedge\} \ (\text{poly-enum } R \ l) \ x = \text{poly-enum-inv } R \ l \ x$
 ⟨proof⟩

definition *poly-mod-ring* :: $('a, 'b) \ \text{idx-ring-enum-scheme} \Rightarrow 'a \ \text{list} \Rightarrow 'a \ \text{list} \ \text{idx-ring-enum}$

where $\text{poly-mod-ring } R \ f = \langle$
 $\text{idx-pred} = (\lambda xs. \ \text{idx-pred } (\text{poly } R) \ xs \wedge \text{length } xs \leq \text{degree } f),$
 $\text{idx-uminus} = \text{idx-uminus } (\text{poly } R),$
 $\text{idx-plus} = (\lambda x \ y. \ \text{pmod}_C \ R \ (x +_C \text{poly } R \ y) \ f),$
 $\text{idx-udivide} = (\lambda x. \ \text{let } ((u, v), r) = \text{ext-euclidean } R \ x \ f \ \text{in } \text{pmod}_C \ R$
 $(r^{-1}_C \ \text{poly } R \ *_C \ \text{poly } R \ u) \ f),$
 $\text{idx-mult} = (\lambda x \ y. \ \text{pmod}_C \ R \ (x *_C \ \text{poly } R \ y) \ f),$
 $\text{idx-zero} = 0_C \ \text{poly } R,$
 $\text{idx-one} = 1_C \ \text{poly } R,$
 $\text{idx-size} = \text{idx-size } R \ \wedge \ \text{degree } f,$
 $\text{idx-enum} = \text{poly-enum } R \ (\text{degree } f),$
 $\text{idx-enum-inv} = \text{poly-enum-inv } R \ (\text{degree } f) \ \rangle$

definition *poly-mod-ring-iso* :: ('a,'b) *idx-ring-enum-scheme* \Rightarrow 'a *list*
 \Rightarrow 'a *list* \Rightarrow 'a *list set*
where *poly-mod-ring-iso* $R f x = \text{PIdl}_{\text{poly-ring (ring-of } R)} f \text{ } +>_{\text{poly-ring (ring-of } R)} x$

definition *poly-mod-ring-iso-inv* :: ('a,'b) *idx-ring-enum-scheme* \Rightarrow 'a
list \Rightarrow 'a *list set* \Rightarrow 'a *list*
where *poly-mod-ring-iso-inv* $R f =$
the-inv-into (carrier (ring-of (poly-mod-ring R f))) (poly-mod-ring-iso
 $R f)$

context
fixes f
fixes $R :: ('a,'b) \text{ idx-ring-enum-scheme}$
assumes *field-R*: $\text{field}_C R$
assumes *f-carr*: $f \in \text{carrier (poly-ring (ring-of } R))$
assumes *deg-f*: $\text{degree } f > 0$
begin

private abbreviation P **where** $P \equiv \text{poly-ring (ring-of } R)$
private abbreviation I **where** $I \equiv \text{PIdl}_{\text{poly-ring (ring-of } R)} f$

interpretation *field ring-of R*
 $\langle \text{proof} \rangle$

interpretation d : *domain P*
 $\langle \text{proof} \rangle$

interpretation i : *ideal I P*
 $\langle \text{proof} \rangle$

interpretation s : *ring-hom-ring P P Quot I (+>P) I*
 $\langle \text{proof} \rangle$

interpretation cr : *cring P Quot I*
 $\langle \text{proof} \rangle$

lemma *ring-c*: $\text{ring}_C R$
 $\langle \text{proof} \rangle$

lemma *d-poly*: $\text{domain}_C (\text{poly } R) \langle \text{proof} \rangle$

lemma *ideal-mod*:
assumes $y \in \text{carrier } P$
shows $I \text{ } +>_P (\text{pmod } y f) = I \text{ } +>_P y$
 $\langle \text{proof} \rangle$

lemma *poly-mod-ring-carr-1*:

$\text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f)) = \{xs. xs \in \text{carrier } P \wedge \text{degree } xs < \text{degree } f\}$
(is ?L = ?R)
 <proof>

lemma poly-mod-ring-carr:
assumes $y \in \text{carrier } P$
shows $\text{pmod } y f \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$
 <proof>

lemma poly-mod-ring-iso-ran:
 $\text{poly-mod-ring-iso } R f \text{ ' carrier } (\text{ring-of } (\text{poly-mod-ring } R f)) = \text{carrier } (P \text{ Quot } I)$
 <proof>

lemma poly-mod-ring-iso-inj:
 $\text{inj-on } (\text{poly-mod-ring-iso } R f) (\text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f)))$
 <proof>

lemma poly-mod-iso-ring-bij:
 $\text{bij-betw } (\text{poly-mod-ring-iso } R f) (\text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))) (\text{carrier } (P \text{ Quot } I))$
 <proof>

lemma poly-mod-iso-ring-bij-2:
 $\text{bij-betw } (\text{poly-mod-ring-iso-inv } R f) (\text{carrier } (P \text{ Quot } I)) (\text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f)))$
 <proof>

lemma poly-mod-ring-iso-inv-1:
assumes $x \in \text{carrier } (P \text{ Quot } I)$
shows $\text{poly-mod-ring-iso } R f (\text{poly-mod-ring-iso-inv } R f x) = x$
 <proof>

lemma poly-mod-ring-iso-inv-2:
assumes $x \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$
shows $\text{poly-mod-ring-iso-inv } R f (\text{poly-mod-ring-iso } R f x) = x$
 <proof>

lemma poly-mod-ring-add:
assumes $x \in \text{carrier } P$
assumes $y \in \text{carrier } P$
shows $x \oplus_{\text{ring-of } (\text{poly-mod-ring } R f)} y = \text{pmod } (x \oplus_P y) f$ **(is ?L = ?R)**
 <proof>

lemma poly-mod-ring-zero: $\mathbf{0}_{\text{ring-of } (\text{poly-mod-ring } R f)} = \mathbf{0}_P$
 <proof>

lemma *poly-mod-ring-one*: $\mathbf{1}_{\text{ring-of } (\text{poly-mod-ring } R f)} = \mathbf{1}_P$
<proof>

lemma *poly-mod-ring-mult*:
 assumes $x \in \text{carrier } P$
 assumes $y \in \text{carrier } P$
 shows $x \otimes_{\text{ring-of } (\text{poly-mod-ring } R f)} y = \text{pmod } (x \otimes_P y) f$ (**is** $?L$
 $= ?R$)
<proof>

lemma *poly-mod-ring-iso-inv*:
 $\text{poly-mod-ring-iso-inv } R f \in \text{ring-iso } (P \text{ Quot } I) (\text{ring-of } (\text{poly-mod-ring } R f))$
 (**is** $?f \in \text{ring-iso } ?S ?T$)
<proof>

lemma *cring-poly-mod-ring-1*:
 shows $\text{ring-of } (\text{poly-mod-ring } R f) \setminus \{\text{zero} := \text{poly-mod-ring-iso-inv } R f \mathbf{0}_{P \text{ Quot } I}\} =$
 $\text{ring-of } (\text{poly-mod-ring } R f)$
 and $\text{cring } (\text{ring-of } (\text{poly-mod-ring } R f))$
<proof>

interpretation *cr-p*: $\text{cring } (\text{ring-of } (\text{poly-mod-ring } R f))$
<proof>

lemma *cring-c-poly-mod-ring*: $\text{cring}_C (\text{poly-mod-ring } R f)$
<proof>

end

lemma *field-c-poly-mod-ring*:
 assumes $\text{field-R: field}_C R$
 assumes $\text{monic-irreducible-poly } (\text{ring-of } R) f$
 shows $\text{field}_C (\text{poly-mod-ring } R f)$
<proof>

lemma *enum-c-poly-mod-ring*:
 assumes $\text{enum}_C R \text{ ring}_C R$
 shows $\text{enum}_C (\text{poly-mod-ring } R f)$
<proof>

end

16 Fast algorithms for Computations of Roots

```

theory Finite-Fields-Nth-Root-Code
imports
  HOL-Computational-Algebra.Nth-Powers
  HOL-Library.Code-Target-Nat
begin

```

This section adds code equations for *nth-root-nat* and *is-nth-power* with fast algorithms using binary search. (The existing implementations in `HOL-Computational-Algebra` perform linear searches, which are too slow. (An example for comparison is the evaluation of the term *nth-root-nat 2 2⁶⁴*).

The following is an implementation of binary search, returning the first index in an interval of the form $[l, u)$ where a predicate becomes true. It returns the upper bound u if the predicate is *False* on the entire domain.

```

function find-first :: nat  $\Rightarrow$  nat  $\Rightarrow$  (nat  $\Rightarrow$  bool)  $\Rightarrow$  nat
where
  find-first l u f = (
    if (l  $\geq$  u) then u else
    (let m = (l + u) div 2 in
      if f m then find-first l m f else find-first (m+1) u f)
  )
  <proof>

```

```

termination <proof>

```

```

lemma Min-subset-eq:
  assumes  $A \subseteq B$  finite B  $A \neq \{\}$   $\forall x \in B. \exists y \in A. y \leq x$ 
  shows  $\text{Min } A = \text{Min } B$ 
  <proof>

```

```

lemma find-first-eq:
  assumes mono f
  shows find-first l u f =  $\text{Min } (\{x. f\ x\} \cap \{l..<u\} \cup \{u\})$ 
  <proof>

```

```

lemma nth-root-nat-fast[code]: nth-root-nat e n = (if e = 0 then 0 else
find-first 0 (n+1) ( $\lambda x. x^e > n$ ) - 1)
  <proof>

```

```

lemma is-nth-power-nat-fast[code]: is-nth-power-nat e n  $\longleftrightarrow$  ((nth-root-nat
e n)e = n)
  <proof>

```

```

end

```

17 Algorithms for finding irreducible polynomials

```

theory Find-Irreducible-Poly
imports
  Finite-Fields-More-PMF
  Finite-Fields-Poly-Factor-Ring-Code
  Rabin-Irreducibility-Test-Code
  Probabilistic-While.While-SPMF
  Card-Irreducible-Polynomials
  Executable-Randomized-Algorithms.Randomized-Algorithm
  Finite-Fields-Nth-Root-Code
  HOL-Library.Log-Nat
begin

hide-const (open) Divisibility.prime
hide-const (open) Finite-Fields-Factorization-Ext.multiplicity
hide-const (open) Numeral-Type.mod-ring
hide-const (open) Polynomial.degree
hide-const (open) Polynomial.order

Enumeration of the monic polynomials in lexicographic order.
definition enum-monic-poly :: ('a,'b) idx-ring-enum-scheme  $\Rightarrow$  nat  $\Rightarrow$ 
nat  $\Rightarrow$  'a list
  where enum-monic-poly A d i = 1CA#[ idx-enum A (nth-digit i j
(idx-size A)). j  $\leftarrow$  rev [0.. $d$ ]]

lemma enum-monic-poly:
  assumes fieldC R enumC R
  shows bij-betw (enum-monic-poly R d) {.. $\text{order (ring-of R)}^d$ }
  {f. monic-poly (ring-of R) f  $\wedge$  degree f = d}
  <proof>

abbreviation tick-spmf :: ('a  $\times$  nat) spmf  $\Rightarrow$  ('a  $\times$  nat) spmf
  where tick-spmf  $\equiv$  map-spmf ( $\lambda(x,c). (x,c+1)$ )

Finds an irreducible polynomial in the finite field mod-ring p
with given degree n:
partial-function (spmf) sample-irreducible-poly :: nat  $\Rightarrow$  nat  $\Rightarrow$  (nat
list  $\times$  nat) spmf
where
  sample-irreducible-poly p n =
  do {
    k  $\leftarrow$  spmf-of-set {.. $p^{\wedge}n$ };
    let poly = enum-monic-poly (mod-ring p) n k;
    if rabin-test (mod-ring p) poly
    then return-spmf (poly,1)
    else tick-spmf (sample-irreducible-poly p n)
  }

```

The following is a deterministic version. It returns the lexicographically minimal monic irreducible polynomial. Note that contrary to the randomized algorithm, the run time of the deterministic algorithm may be exponential (w.r.t. to the size of the field and degree of the polynomial).

```
fun find-irreducible-poly :: nat ⇒ nat ⇒ nat list
  where find-irreducible-poly p n = (let f = enum-monic-poly (mod-ring p) n in
    f (while ((λk. ¬rabin-test (mod-ring p) (f k))) (λx. x + 1) 0))
```

```
definition cost :: ('a × nat) option ⇒ enat
  where cost x = (case x of None ⇒ ∞ | Some (-,r) ⇒ enat r)
```

```
lemma cost-tick: cost (map-option (λ(x, c). (x, Suc c)) c) = eSuc (cost c)
  ⟨proof⟩
```

```
context
  fixes n p :: nat
  assumes p-prime: Factorial-Ring.prime p
  assumes n-gt-0: n > 0
begin
```

```
private definition S where S = {f. monic-poly (ring-of (mod-ring p)) f ∧ degree f = n }
private definition T where T = {f. monic-irreducible-poly (ring-of (mod-ring p)) f ∧ degree f = n }
```

```
lemmas field-c = mod-ring-is-field-c[OF p-prime]
lemmas enum-c = mod-ring-is-enum-c[where n=p]
```

```
interpretation finite-field ring-of (mod-ring p)
  ⟨proof⟩ lemmas field-ops = field-cD[OF field-c]
```

```
private lemma S-fin: finite S
  ⟨proof⟩ lemma T-sub-S: T ⊆ S
  ⟨proof⟩ lemma T-card-gt-0: real (card T) > 0
  ⟨proof⟩ lemma S-card-gt-0: real (card S) > 0
  ⟨proof⟩ lemma S-ne: S ≠ {} ⟨proof⟩ lemma sample-irreducible-poly-step-aux:
    do {
      k ← spmf-of-set {..

```

```

    else x
  }
  (is ?L = ?R)
<proof> lemma sample-irreducible-poly-step:
  sample-irreducible-poly p n =
  do {
    poly ← spmf-of-set S;
    if monic-irreducible-poly (ring-of (mod-ring p)) poly
    then return-spmf (poly, 1)
    else tick-spmf (sample-irreducible-poly p n)
  }
<proof> lemma sample-irreducible-poly-aux-1:
ord-spmf (=) (map-spmf fst (sample-irreducible-poly p n)) (spmf-of-set
T)
<proof>

```

lemma *cost-sample-irreducible-poly*:
 $(\int^+ x. \text{cost } x \text{ } \partial \text{sample-irreducible-poly } p \text{ } n) \leq 2 * \text{real } n$ (is ?L ≤ ?R)
<proof> **lemma** *weight-sample-irreducible-poly*:
 $\text{weight-spmf (sample-irreducible-poly } p \text{ } n) = 1$ (is ?L = ?R)
<proof>

lemma *sample-irreducible-poly-result*:
 $\text{map-spmf fst (sample-irreducible-poly } p \text{ } n) =$
 $\text{spmf-of-set } \{f. \text{monic-irreducible-poly (ring-of (mod-ring } p)) f \wedge$
 $\text{degree } f = n\}$ (is ?L = ?R)
<proof>

lemma *find-irreducible-poly-result*:
defines $\text{res} \equiv \text{find-irreducible-poly } p \text{ } n$
shows $\text{monic-irreducible-poly (ring-of (mod-ring } p)) \text{res degree res}$
 $= n$
<proof>

lemma *monic-irred-poly-set-nonempty-finite*:
 $\{f. \text{monic-irreducible-poly (ring-of (mod-ring } p)) f \wedge \text{degree } f = n\}$
 $\neq \{\}$ (is ?R1)
 $\text{finite } \{f. \text{monic-irreducible-poly (ring-of (mod-ring } p)) f \wedge \text{degree } f$
 $= n\}$ (is ?R2)
<proof>

end

Returns m e such that $n = m^e$, where e is maximal.

definition *split-power* :: $\text{nat} \Rightarrow \text{nat} \times \text{nat}$
where $\text{split-power } n =$ (
 $\text{let } e = \text{last (filter } (\lambda x. \text{is-nth-power-nat } x \text{ } n) (1 \# [2..<\text{floorlog } 2$
 $n]))$
 $\text{in (nth-root-nat } e \text{ } n, e)$)

lemma *split-power-result*:

assumes $(x,e) = \text{split-power } n$

shows $n = x^e \wedge k. n > 1 \implies k > e \implies \neg \text{is-nth-power } k \ n$

<proof>

definition *not-perfect-power* :: $\text{nat} \Rightarrow \text{bool}$

where $\text{not-perfect-power } n = (n > 1 \wedge (\forall x \ k. n = x^k \longrightarrow k = 1))$

lemma *is-nth-power-from-multiplicities*:

assumes $n > (0::\text{nat})$

assumes $\bigwedge p. \text{Factorial-Ring.prime } p \implies k \ \text{dvd} \ (\text{multiplicity } p \ n)$

shows $\text{is-nth-power } k \ n$

<proof>

lemma *power-inj-aux*:

assumes $\text{not-perfect-power } a \ \text{not-perfect-power } b$

assumes $n > 0 \ m > n$

assumes $a^n = b^m$

shows *False*

<proof>

Generalization of *prime-power-inj'*

lemma *power-inj*:

assumes $\text{not-perfect-power } a \ \text{not-perfect-power } b$

assumes $n > 0 \ m > 0$

assumes $a^n = b^m$

shows $a = b \wedge n = m$

<proof>

lemma *split-power-base-not-perfect*:

assumes $n > 1$

shows $\text{not-perfect-power} \ (\text{fst} \ (\text{split-power } n))$

<proof>

lemma *prime-not-perfect*:

assumes $\text{Factorial-Ring.prime } p$

shows $\text{not-perfect-power } p$

<proof>

lemma *split-power-prime*:

assumes $\text{Factorial-Ring.prime } p \ n > 0$

shows $\text{split-power} \ (p^n) = (p,n)$

<proof>

definition *is-prime-power* :: $\text{nat} \Rightarrow \text{bool}$ **where**

$\text{is-prime-power } n = (\exists p \ k. \text{prime } p \wedge k > 0 \wedge n = p^k)$

lemma *is-prime-powerI*:
assumes *prime* p $k > 0$
shows *is-prime-power* $(p \wedge k)$
 \langle *proof* \rangle

definition *GF* **where**
 $GF\ n =$ (
 let $(p,k) =$ *split-power* n ;
 f = *find-irreducible-poly* $p\ k$
 in *poly-mod-ring* $(\text{mod-ring } p)\ f$)

definition GF_R **where**
 $GF_R\ n =$
 do {
 let $(p,k) =$ *split-power* n ;
 f \leftarrow *sample-irreducible-poly* $p\ k$;
 return-spmf $(\text{poly-mod-ring } (\text{mod-ring } p)\ (\text{fst } f))$
 }

lemma *GF-in-GF-R*:
assumes *is-prime-power* n
shows $GF\ n \in$ *set-spmf* $(GF_R\ n)$
 \langle *proof* \rangle

lemma *galois-field-random-1*:
assumes *is-prime-power* n
shows $\bigwedge \omega. \omega \in$ *set-spmf* $(GF_R\ n) \implies$ *enum* $_C\ \omega \wedge$ *field* $_C\ \omega \wedge$ *order*
(ring-of $\omega) = n$
 and *lossless-spmf* $(GF_R\ n)$
 \langle *proof* \rangle

lemma *galois-field*:
assumes *is-prime-power* n
shows *enum* $_C\ (GF\ n)$ *field* $_C\ (GF\ n)$ *order* $(\text{ring-of } (GF\ n)) = n$
 \langle *proof* \rangle

lemma *lossless-imp-spmf-of-pmf*:
assumes *lossless-spmf* M
shows *spmf-of-pmf* $(\text{map-pmf the } M) = M$
 \langle *proof* \rangle

lemma *galois-field-random-2*:
assumes *is-prime-power* n
shows *map-spmf* $(\lambda \omega. \text{enum}_C\ \omega \wedge \text{field}_C\ \omega \wedge \text{order } (\text{ring-of } \omega) = n)$ $(GF_R\ n) =$ *return-spmf* *True*
 (is ?L = -)
 \langle *proof* \rangle

lemma *bind-galois-field-cong*:
assumes *is-prime-power* n
assumes $\bigwedge \omega. \text{enum}_C \omega \implies \text{field}_C \omega \implies \text{order}(\text{ring-of } \omega) = n \implies$
 $f \omega = g \omega$
shows $\text{bind-spmf}(GF_R n) f = \text{bind-spmf}(GF_R n) g$
 $\langle \text{proof} \rangle$

end

References

- [1] S. K. Chebolu and J. Mináč. Counting irreducible polynomials over finite fields using the inclusion-exclusion principle. *Mathematics Magazine*, 84:369 – 371, 2010.
- [2] M. Eberl. Dirichlet series. *Archive of Formal Proofs*, Oct. 2017. https://isa-afp.org/entries/Dirichlet_Series.html, Formal proof development.
- [3] K. Ireland and M. Rosen. *A classical introduction to modern number theory*, volume 84 of *Graduate texts in mathematics*. Springer, 1982.
- [4] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, USA, 1986.
- [5] M. O. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computing*, 9(2):273–280, 1980.