Finite Fields

Emin Karayel

March 17, 2025

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

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]

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

[1] Thanks to Katharina Kreuzer for discovering that formalization.

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 $\bigwedge i.\ i \in A \Longrightarrow g\ i \leq f\ i$
  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 $\bigwedge i.\ i \in A \Longrightarrow f\ i = 0$
⟨*proof*⟩

lemma *sum'-eq-iff*:
  fixes $f :: 'a \Rightarrow nat$
  assumes *finite* $\{i \in A.\ f\ i \neq 0\}$

assumes $\bigwedge i.\ i \in A \Longrightarrow f\ i \geq g\ i$
assumes $sum'\ f\ A \leq sum'\ g\ A$
shows $\forall\, i \in A.\ f\ i = g\ i$
⟨*proof*⟩

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
⟨*proof*⟩

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

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 $f \in carrier\ (K[X]) - \{\mathbf{0}_{K[X]}\}$
assumes $g \in carrier\ (K[X]) - \{\mathbf{0}_{K[X]}\}$
assumes $degree\ f \neq degree\ g$
shows $degree\ (f \oplus_{K[X]} g) = max\ (degree\ f)\ (degree\ g)$
⟨*proof*⟩

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

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

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

lemma (**in** *domain*) *pow-non-zero*:
$x \in$ *carrier* $R \Longrightarrow x \neq \mathbf{0} \Longrightarrow x\ [\widehat{}]\ (n :: nat) \neq \mathbf{0}$
⟨*proof*⟩

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

lemma (**in** *ring*) *degree-var*:
degree $(X_R) = 1$
⟨*proof*⟩

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

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

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

lemma (**in** *domain*) *finprod-non-zero*:
assumes *finite* A
assumes $f \in A \to$ *carrier* $R - \{\mathbf{0}\}$
shows $(\bigotimes i \in A.\ f\ i) \in$ *carrier* $R - \{\mathbf{0}\}$
⟨*proof*⟩

lemma (**in** *domain*) *degree-prod*:
assumes *finite* A
assumes *subring* K R
assumes $f \in A \to$ *carrier* $(K[X]) - \{\mathbf{0}_{K[X]}\}$
shows *degree* $(\bigotimes_{K[X]} i \in A.\ f\ i) = (\sum i \in A.\ degree\ (f\ i))$
⟨*proof*⟩

lemma (**in** *ring*) *coeff-add*:
assumes *subring* K R

assumes $f \in carrier\ (K[X])$ $g \in carrier\ (K[X])$
shows $coeff\ (f \oplus_{K[X]} g)\ i = coeff\ f\ i \oplus_R coeff\ g\ i$
⟨*proof*⟩

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

This is a version of geometric sums for commutative rings:

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

lemma (**in** *domain*) *rupture-eq-0-iff*:
assumes *subfield K R p* $\in carrier\ (K[X])$ $q \in carrier\ (K[X])$
shows $rupture\text{-}surj\ K\ p\ q = \mathbf{0}_{Rupt\ K\ p} \longleftrightarrow p\ pdivides\ q$
(**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 $h \in ring\text{-}iso\ R\ S$ *domain R domain S*
shows $map\ h \in ring\text{-}iso\ (poly\text{-}ring\ R)\ (poly\text{-}ring\ S)$
⟨*proof*⟩

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

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

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

lemma *divides-hom*:
assumes $h \in$ *ring-iso R S*
assumes *domain R domain S*
assumes $x \in$ *carrier R* $y \in$ *carrier R*
shows x *divides*$_R$ $y \longleftrightarrow (h\ x)$ *divides*$_S$ $(h\ y)$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *properfactor-hom*:
assumes $h \in$ *ring-iso R S*
assumes *domain R domain S*
assumes $x \in$ *carrier R* $b \in$ *carrier R*
shows *properfactor R b x* $\longleftrightarrow$ *properfactor S* $(h\ b)$ $(h\ x)$
⟨*proof*⟩

lemma *Units-hom*:
assumes $h \in$ *ring-iso R S*
assumes *domain R domain S*
assumes $x \in$ *carrier R*
shows $x \in$ *Units R* $\longleftrightarrow$ $h\ x \in$ *Units S*
⟨*proof*⟩

lemma *irreducible-hom*:
assumes $h \in$ *ring-iso R S*
assumes *domain R domain S*
assumes $x \in$ *carrier R*
shows *irreducible R x* = *irreducible S* $(h\ x)$
⟨*proof*⟩

lemma *pirreducible-hom*:
assumes $h \in$ *ring-iso R S*
assumes *domain R domain S*
assumes $f \in$ *carrier* (*poly-ring R*)
shows *pirreducible*$_R$ (*carrier R*) f =
pirreducible$_S$ (*carrier S*) (*map h f*)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *ring-hom-cong*:
assumes $\bigwedge x.\ x \in$ *carrier R* $\Longrightarrow f'\ x = f\ x$
assumes *ring R*
assumes $f \in$ *ring-hom R S*
shows $f' \in$ *ring-hom R S*
⟨*proof*⟩

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 $(\lambda x.\ (J <+>_R x)) \in$ *ring-hom* $(R\ Quot\ I)\ (R\ Quot\ J)$
⟨*proof*⟩

lemma (**in** *ring*) *quot-carr*:
assumes *ideal I R*
assumes $y \in$ *carrier* $(R\ Quot\ I)$
shows $y \subseteq$ *carrier R*
⟨*proof*⟩

lemma (**in** *ring*) *set-add-zero*:
assumes $A \subseteq$ *carrier R*
shows $\{\mathbf{0}\} <+>_R A = A$
⟨*proof*⟩

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

lemma (**in** *domain*) *rupture-surj-as-eval*:
assumes *subring K R*
assumes $p \in$ *carrier* $(K[X])$ $q \in$ *carrier* $(K[X])$
shows *rupture-surj K p q* =
ring.eval (*Rupt K p*) (*map* ((*rupture-surj K p*) ∘ *poly-of-const*) *q*)
(*rupture-surj K p X*)
⟨*proof*⟩

2.4 Divisibility

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

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

sublocale *field* < *mult-of*: *comm-group mult-of R*
rewrites *mult* (*mult-of R*) = *mult R*
and *one* (*mult-of R*) = *one R*
⟨*proof*⟩

lemma (**in** *domain*) *div-neg*:
assumes $a \in$ *carrier R* $b \in$ *carrier R*
assumes *a divides b*

shows *a divides* $(\ominus\ b)$
⟨*proof*⟩

lemma (**in** *domain*) *div-sum*:
assumes $a \in$ *carrier* R $b \in$ *carrier* R $c \in$ *carrier* R
assumes *a divides b*
assumes *a divides c*
shows *a divides* $(b \oplus c)$
⟨*proof*⟩

lemma (**in** *domain*) *div-sum-iff*:
assumes $a \in$ *carrier* R $b \in$ *carrier* R $c \in$ *carrier* R
assumes *a divides b*
shows *a divides* $(b \oplus c) \longleftrightarrow$ *a divides c*
⟨*proof*⟩

lemma (**in** *comm-monoid*) *irreducible-prod-unit*:
assumes $f \in$ *carrier* G $x \in$ *Units* G
shows *irreducible* $G\ f =$ *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*. (∃ *as*. *f* = *fmset G as* ∧ *wfactors G as x* ∧ *set as* ⊆ *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 $G\ d\ g =$ *Max* $\{(n::nat).\ (d\ [\hat{}]_G\ n)\ divides_G\ g\}$

definition *canonical-irreducibles* **where**
canonical-irreducibles $G\ A = ($
$A \subseteq \{a.\ a \in$ *carrier* $G \wedge$ *irreducible* $G\ a\} \wedge$
$(\forall x\ y.\ x \in A \longrightarrow y \in A \longrightarrow x \sim_G y \longrightarrow x = y) \wedge$

$(\forall\, x \in \mathit{carrier}\ G.\ \mathit{irreducible}\ G\ x \longrightarrow (\exists\, y \in A.\ x \sim_G y)))$

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 $a \in \mathit{carrier}\ G$
    and $b \in \mathit{carrier}\ G$
    and $\mathit{set}\ \mathit{as} \subseteq \mathit{carrier}\ G$
    and $\mathit{set}\ \mathit{bs} \subseteq \mathit{carrier}\ G$
  shows $a \sim b \longleftrightarrow (\mathit{fmset}\ G\ \mathit{as} = \mathit{fmset}\ G\ \mathit{bs})$
⟨*proof*⟩

lemma *factor-mset-aux-1*:
  assumes $a \in \mathit{carrier}\ G\ \mathit{set}\ \mathit{as} \subseteq \mathit{carrier}\ G\ \mathit{wfactors}\ G\ \mathit{as}\ a$
  shows $\mathit{factor\text{-}mset}\ G\ a = \mathit{fmset}\ G\ \mathit{as}$
⟨*proof*⟩

lemma *factor-mset-aux*:
  assumes $a \in \mathit{carrier}\ G$
  shows $\exists\, \mathit{as}.\ \mathit{factor\text{-}mset}\ G\ a = \mathit{fmset}\ G\ \mathit{as} \wedge \mathit{wfactors}\ G\ \mathit{as}\ a \wedge$
    $\mathit{set}\ \mathit{as} \subseteq \mathit{carrier}\ G$
⟨*proof*⟩

lemma *factor-mset-set*:
  assumes $a \in \mathit{carrier}\ G$
  assumes $x \in\# \mathit{factor\text{-}mset}\ G\ a$
  obtains y **where**
    $y \in \mathit{carrier}\ G$
    $\mathit{irreducible}\ G\ y$
    $\mathit{assocs}\ G\ y = x$
⟨*proof*⟩

lemma *factor-mset-mult*:
  assumes $a \in \mathit{carrier}\ G\ b \in \mathit{carrier}\ G$
  shows $\mathit{factor\text{-}mset}\ G\ (a \otimes b) = \mathit{factor\text{-}mset}\ G\ a + \mathit{factor\text{-}mset}\ G\ b$
⟨*proof*⟩

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

lemma *factor-mset-irred*:

assumes $x \in$ *carrier* G *irreducible* G x
shows *factor-mset* G x = *image-mset* (*assocs* G) $\{\#x\#\}$
⟨*proof*⟩

lemma *factor-mset-divides*:
assumes $a \in$ *carrier* G $b \in$ *carrier* G
shows a *divides* $b \longleftrightarrow$ *factor-mset* G $a \subseteq\#$ *factor-mset* G b
⟨*proof*⟩

lemma *factor-mset-sim*:
assumes $a \in$ *carrier* G $b \in$ *carrier* G
shows $a \sim b \longleftrightarrow$ *factor-mset* G a = *factor-mset* G b
⟨*proof*⟩

lemma *factor-mset-prod*:
assumes *finite* A
assumes f ‘ $A \subseteq$ *carrier* G
shows *factor-mset* G $(\bigotimes a \in A.\ f\ a)$ =
$(\sum a \in A.$ *factor-mset* G $(f\ a))$
⟨*proof*⟩

lemma *factor-mset-pow*:
assumes $a \in$ *carrier* G
shows *factor-mset* G $(a\ [\hat{}]\ n)$ = *repeat-mset* n (*factor-mset* G a)
⟨*proof*⟩

lemma *image-mset-sum*:
assumes *finite* F
shows
image-mset h $(\sum x \in F.\ f\ x)$ = $(\sum x \in F.$ *image-mset* h $(f\ x))$
⟨*proof*⟩

lemma *decomp-mset*:
$(\sum x \in$*set-mset* $R.$ *replicate-mset* (*count* R x) $x)$ = R
⟨*proof*⟩

lemma *factor-mset-count*:
assumes $a \in$ *carrier* G $d \in$ *carrier* G *irreducible* G d
shows *count* (*factor-mset* G a) (*assocs* G d) = *multiplicity* G d a
⟨*proof*⟩

lemma *multiplicity-ge-iff*:
assumes $d \in$ *carrier* G *irreducible* G d $a \in$ *carrier* G
shows *multiplicity* G d $a \geq k \longleftrightarrow$ d $[\hat{}]$ k *divides* a
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *multiplicity-gt-0-iff*:
assumes $d \in$ *carrier* G *irreducible* G d $a \in$ *carrier* G

shows *multiplicity G d a > 0 ⟷ d divides a*
⟨*proof*⟩

lemma *factor-mset-count-2*:
assumes *a ∈ carrier G*
assumes ⋀*z*. *z ∈ carrier G ⟹ irreducible G z ⟹ y ≠ assocs G z*
shows *count* (*factor-mset G a*) *y* = *0*
⟨*proof*⟩

lemma *factor-mset-choose*:
assumes *a ∈ carrier G set-mset R ⊆ carrier G*
assumes *image-mset* (*assocs G*) *R* = *factor-mset G a*
shows *a* ∼ (⨂ *x*∈*set-mset R*. *x* [^] *count R x*) (**is** *a* ∼ *?rhs*)
⟨*proof*⟩

lemma *divides-iff-mult-mono*:
assumes *a ∈ carrier G b ∈ carrier G*
assumes *canonical-irreducibles G R*
assumes ⋀*d*. *d ∈ R ⟹ multiplicity G d a ≤ multiplicity G d b*
shows *a divides b*
⟨*proof*⟩

lemma *count-image-mset-inj*:
assumes *inj-on f R x ∈ R set-mset A ⊆ R*
shows *count* (*image-mset f A*) (*f x*) = *count A x*
⟨*proof*⟩

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

lemma *split-factors*:
assumes *canonical-irreducibles G R*
assumes *a ∈ carrier G*
shows
finite {*d*. *d ∈ R* ∧ *multiplicity G d a > 0*}
a ∼ (⨂ *d*∈{*d*. *d ∈ R* ∧ *multiplicity G d a > 0*}.
d [^] *multiplicity G d a*) (**is** *a* ∼ *?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 = *Idl*$_{\mathbb{Z}}$ {*int p*} +>$_{\mathbb{Z}}$ (*int k*)

context
  fixes *n* :: *nat*
  assumes *n-gt-0*: *n* > *0*

begin

private abbreviation I **where** $I \equiv Idl_{\mathcal{Z}}\ \{int\ n\}$

private lemma *ideal-I*: *ideal* I $\mathcal{Z}$
⟨*proof*⟩

lemma *int-cosetI*:
assumes $u\ mod\ (int\ n) = v\ mod\ (int\ n)$
shows $Idl_{\mathcal{Z}}\ \{int\ n\} +>_{\mathcal{Z}} u = Idl_{\mathcal{Z}}\ \{int\ n\} +>_{\mathcal{Z}} v$
⟨*proof*⟩

lemma *zfact-iso-inj*:
inj-on (*zfact-iso n*) {..<*n*}
⟨*proof*⟩

lemma *zfact-iso-ran*:
zfact-iso n ‘ {..<*n*} = *carrier* (*ZFact* (*int n*))
⟨*proof*⟩

lemma *zfact-iso-bij*:
bij-betw (*zfact-iso n*) {..<*n*} (*carrier* (*ZFact* (*int n*)))
⟨*proof*⟩

lemma *card-zfact-carr*: *card* (*carrier* (*ZFact* (*int n*))) = *n*
⟨*proof*⟩

lemma *fin-zfact*: *finite* (*carrier* (*ZFact* (*int n*)))
⟨*proof*⟩

end

lemma *zfact-prime-is-finite-field*:
assumes *Factorial-Ring.prime p*
shows *finite-field* (*ZFact* (*int p*))
⟨*proof*⟩

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

lemma (**in** *ring*) *add-pow-consistent*:
fixes *i* :: *int*
assumes *subring K R*
assumes $k \in K$
shows *add-pow R i k* = *add-pow* (*R* (| *carrier* := *K* |)) *i k*
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma (**in** *ring*) *int-embed-consistent*:

assumes *subring K R*
shows *int-embed R i* = *int-embed* (*R* (| *carrier* := *K* |)) *i*
⟨*proof*⟩

lemma (**in** *ring*) *int-embed-closed*:
int-embed R k ∈ *carrier R*
⟨*proof*⟩

lemma (**in** *ring*) *int-embed-range*:
assumes *subring K R*
shows *int-embed R k* ∈ *K*
⟨*proof*⟩

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

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

lemma (**in** *ring*) *int-embed-add*:
int-embed R (*x*+*y*) = *int-embed R x* $\oplus_R$ *int-embed R y*
⟨*proof*⟩

lemma (**in** *ring*) *int-embed-inv*:
int-embed R (−*x*) = $\ominus_R$ *int-embed R x* (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma (**in** *ring*) *int-embed-diff*:
int-embed R (*x*−*y*) = *int-embed R x* $\ominus_R$ *int-embed R y*
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma (**in** *ring*) *int-embed-mult-aux*:
int-embed R (*x*∗*int y*) = *int-embed R x* ⊗ *int-embed R y*
⟨*proof*⟩

lemma (**in** *ring*) *int-embed-mult*:
int-embed R (*x*∗*y*) = *int-embed R x* $\otimes_R$ *int-embed R y*
⟨*proof*⟩

lemma (**in** *ring*) *int-embed-ring-hom*:
ring-hom-ring int-ring R (*int-embed R*)
⟨*proof*⟩

abbreviation *char-subring* **where**
char-subring R ≡ *int-embed R* ‘ *UNIV*

definition *char* **where**
char R = *card* (*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 *int-embed R* (*int x*) = **0**
shows *char R* ≤ *x char R* > *0*
⟨*proof*⟩

lemma (**in** *ring*) *embed-char-eq-0*:
int-embed R (*int* (*char R*)) = **0**
⟨*proof*⟩

lemma (**in** *ring*) *embed-char-eq-0-iff*:
fixes *n* :: *int*
shows *int-embed R n* = **0** ⟷ *char R dvd n*
⟨*proof*⟩

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

lemma (**in** *domain*) *characteristic-is-prime*:
assumes *char R* > *0*
shows *prime* (*char R*)
⟨*proof*⟩

lemma (**in** *ring*) *char-ring-is-subring*:
subring (*char-subring R*) *R*
⟨*proof*⟩

lemma (**in** *cring*) *char-ring-is-subcring*:
subcring (*char-subring R*) *R*
⟨*proof*⟩

lemma (**in** *domain*) *char-ring-is-subdomain*:
subdomain (*char-subring R*) *R*
⟨*proof*⟩

lemma *image-set-eqI*:
assumes ⋀*x*. *x* ∈ *A* ⟹ *f x* ∈ *B*

assumes $\bigwedge x.\ x \in B \Longrightarrow g\ x \in A \wedge f\ (g\ x) = x$
shows $f\ `\ A = B$
⟨*proof*⟩

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 $(x \oplus y)\ [\widehat{}]\ n =$
$(\bigoplus k \in \{..n\}.\ int\text{-}embed\ R\ (n\ choose\ k) \otimes x\ [\widehat{}]\ k \otimes y\ [\widehat{}]\ (n-k))$
⟨*proof*⟩

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

theorem (**in** *domain*) *freshmans-dream*:
assumes $char\ R > 0$
assumes [*simp*]: $x \in carrier\ R\ y \in carrier\ R$
shows $(x \oplus y)\ [\widehat{}]\ (char\ R) = x\ [\widehat{}]\ char\ R \oplus y\ [\widehat{}]\ char\ R$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

The following theorem is somtimes 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 $n \equiv char\ R \hat{}\ m$
shows $(x \oplus y)\ [\widehat{}]\ n = x\ [\widehat{}]\ n \oplus y\ [\widehat{}]\ n$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

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 $m = char\ R \hat{}\ k$
shows *ring-hom-cring R R* $(\lambda x.\ x\ [\widehat{}]\ m)$
⟨*proof*⟩

lemma (**in** *domain*) *char-ring-is-subfield*:
assumes $char\ R > 0$

shows *subfield* (*char-subring R*) *R*
⟨*proof*⟩

lemma *card-lists-length-eq'*:
fixes *A* :: *'a set*
shows *card* {*xs*. *set xs* ⊆ *A* ∧ *length xs* = *n*} = *card A* ^ *n*
⟨*proof*⟩

lemma (**in** *ring*) *card-span*:
assumes *subfield K R*
assumes *independent K w*
assumes *set w* ⊆ *carrier R*
shows *card* (*Span K w*) = *card K*^(*length w*)
⟨*proof*⟩

lemma (**in** *ring*) *finite-carr-imp-char-ge-0*:
assumes *finite* (*carrier R*)
shows *char R* > *0*
⟨*proof*⟩

lemma (**in** *ring*) *char-consistent*:
assumes *subring H R*
shows *char* (*R* (| *carrier* := *H* |)) = *char R*
⟨*proof*⟩

lemma (**in** *ring-hom-ring*) *char-consistent*:
assumes *inj-on h* (*carrier R*)
shows *char R* = *char S*
⟨*proof*⟩

definition *char-iso* :: - ⇒ *int set* ⇒ *'a*
where *char-iso R x* = *the-elem* (*int-embed R* ‘ *x*)

The function *char-iso R* denotes the isomorphism between *ZFact* (*int* (*char R*)) and the characteristic subring.

lemma (**in** *ring*) *char-iso*: *char-iso R* ∈
ring-iso (*ZFact* (*char R*)) (*R*(|*carrier* := *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*:
∃ *n*. *order R* = *char R* ^ *n* ∧ *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 (λ*i*. *int-embed R i* ⊗$_R$ *ring.coeff R x i*) (*rev* [*1*..<*length x*]))

context *domain*
begin

lemma *coeff-range*:
assumes *subring K R*
assumes *f* ∈ *carrier* (*K*[*X*])
shows *coeff f i* ∈ *K*
⟨*proof*⟩

lemma *pderiv-carr*:
assumes *subring K R*
assumes *f* ∈ *carrier* (*K*[*X*])
shows *pderiv f* ∈ *carrier* (*K*[*X*])
⟨*proof*⟩

lemma *pderiv-coeff*:
assumes *subring K R*
assumes *f* ∈ *carrier* (*K*[*X*])
shows *coeff* (*pderiv f*) *k* = *int-embed R* (*Suc k*) ⊗ *coeff f* (*Suc k*)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *pderiv-const*:
assumes *degree x* = *0*
shows *pderiv x* = **0**$_{K[X]}$
⟨*proof*⟩

lemma *pderiv-var*:
shows *pderiv X* = **1**$_{K[X]}$
⟨*proof*⟩

lemma *pderiv-zero*:
shows *pderiv* **0**$_{K[X]}$ = **0**$_{K[X]}$
⟨*proof*⟩

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

(**is** *?lhs = ?rhs*)
⟨*proof*⟩

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

lemma *coeff-mult*:
assumes *subring K R*
assumes $f \in carrier\ (K[X])$ $g \in carrier\ (K[X])$
shows $coeff\ (f \otimes_{K[X]}\ g)\ i =$
$(\bigoplus\ k \in \{..i\}.\ (coeff\ f)\ k \otimes (coeff\ g)\ (i - k))$
⟨*proof*⟩

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

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

lemma *int-embed-consistent-with-poly-of-const*:
assumes *subring K R*
shows $int\text{-}embed\ (K[X])\ m = poly\text{-}of\text{-}const\ (int\text{-}embed\ R\ m)$
⟨*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 $\bigwedge x.\ x \in A \Longrightarrow f\ x \in carrier\ (mult\text{-}of\ R)$
shows *finprod R f A* = *finprod* (*mult-of R*) *f A*
⟨*proof*⟩

lemma (**in** *ring*) *finite-poly*:
assumes *subring K R*
assumes *finite K*
shows
finite $\{f.\ f \in carrier\ (K[X]) \wedge degree\ f = n\}$ (**is** *finite ?A*)
finite $\{f.\ f \in carrier\ (K[X]) \wedge degree\ f \leq n\}$ (**is** *finite ?B*)
⟨*proof*⟩

definition *pmult* :: - ⇒ *'a list* ⇒ *'a list* ⇒ *nat* (‹*pmult*₁›)
where $pmult_R\ d\ p = multiplicity\ (mult\text{-}of\ (poly\text{-}ring\ R))\ d\ p$

definition *monic-poly* :: - ⇒ *'a list* ⇒ *bool*
where *monic-poly R f* =
$(f \neq [] \wedge lead\text{-}coeff\ f = \mathbf{1}_R \wedge f \in carrier\ (poly\text{-}ring\ R))$

definition *monic-irreducible-poly* **where**
monic-irreducible-poly R f =
$(monic\text{-}poly\ R\ f \wedge pirreducible_R\ (carrier\ R)\ f)$

abbreviation *m-i-p* ≡ *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* ≡ *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 intepretation 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 $a \in$ *carrier* (*mult-of P*)
assumes $b \in$ *carrier* (*mult-of P*)
assumes $c \in$ *carrier* (*mult-of P*)
shows $a \otimes_P c$ *pdivides* $b \otimes_P c \longleftrightarrow a$ *pdivides* b
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

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

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

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

lemma *monic-poly-carr*:
assumes *monic-poly R f*
shows $f \in$ *carrier P*
⟨*proof*⟩

lemma *monic-poly-add-distinct*:
assumes *monic-poly R f*
assumes $g \in$ *carrier P degree* $g <$ *degree* f
shows *monic-poly R* $(f \oplus_P g)$
⟨*proof*⟩

lemma *monic-poly-one*: *monic-poly R* $\mathbf{1}_P$
⟨*proof*⟩

lemma *monic-poly-var*: *monic-poly R X*
⟨*proof*⟩

lemma *monic-poly-carr-2*:
assumes *monic-poly R f*
shows $f \in$ *carrier* (*mult-of P*)

⟨*proof*⟩

lemma *monic-poly-mult*:
assumes *monic-poly R f*
assumes *monic-poly R g*
shows *monic-poly R* (*f* ⊗$_P$ *g*)
⟨*proof*⟩

lemma *monic-poly-pow*:
assumes *monic-poly R f*
shows *monic-poly R* (*f* [^]$_P$ (*n*::*nat*))
⟨*proof*⟩

lemma *monic-poly-prod*:
assumes *finite A*
assumes ⋀*x*. *x* ∈ *A* ⟹ *monic-poly R* (*f x*)
shows *monic-poly R* (*finprod P f A*)
⟨*proof*⟩

lemma *monic-poly-not-assoc*:
assumes *monic-poly R f*
assumes *monic-poly R g*
assumes *f* ∼$_{(mult\text{-}of\ P)}$ *g*
shows *f* = *g*
⟨*proof*⟩

lemma *monic-poly-span*:
assumes *x* ∈ *carrier* (*mult-of P*) *irreducible* (*mult-of P*) *x*
shows ∃ *y*. *monic-irreducible-poly R y* ∧ *x* ∼$_{(mult\text{-}of\ P)}$ *y*
⟨*proof*⟩

lemma *monic-polys-are-canonical-irreducibles*:
canonical-irreducibles (*mult-of P*) {*d*. *monic-irreducible-poly R d*}
(**is** *canonical-irreducibles* (*mult-of P*) *?S*)
⟨*proof*⟩

lemma
assumes *monic-poly R a*
shows *factor-monic-poly*:
a = (⨂$_P$*d*∈{*d*. *monic-irreducible-poly R d* ∧ *pmult d a* > *0*}.
d [^]$_P$ *pmult d a*) (**is** *?lhs* = *?rhs*)
and *factor-monic-poly-fin*:
finite {*d*. *monic-irreducible-poly R d* ∧ *pmult d a* > *0*}
⟨*proof*⟩

lemma *degree-monic-poly′*:
assumes *monic-poly R f*
shows
sum′ (λ*d*. *pmult d f* ∗ *degree d*) {*d*. *monic-irreducible-poly R d*} =

degree f
⟨*proof*⟩

lemma *monic-poly-min-degree*:
assumes *monic-irreducible-poly R f*
shows *degree f ≥ 1*
⟨*proof*⟩

lemma *degree-one-monic-poly*:
monic-irreducible-poly R f ∧ degree f = 1 ⟷
*(∃ x ∈ carrier R. f = [**1**, ⊖x])*
⟨*proof*⟩

lemma *multiplicity-ge-iff*:
assumes *monic-irreducible-poly R d*
assumes *f ∈ carrier P − {**0**$_P$}*
shows *pmult d f ≥ k ⟷ d* $\hat{}_P$ *k pdivides f*
⟨*proof*⟩

lemma *multiplicity-ge-1-iff-pdivides*:
assumes *monic-irreducible-poly R d f ∈ carrier P − {**0**$_P$}*
shows *pmult d f ≥ 1 ⟷ d pdivides f*
⟨*proof*⟩

lemma *divides-monic-poly*:
assumes *monic-poly R f monic-poly R g*
assumes *⋀d. monic-irreducible-poly R d*
⟹ pmult d f ≤ pmult d g
shows *f pdivides g*
⟨*proof*⟩

end

lemma *monic-poly-hom*:
assumes *monic-poly R f*
assumes *h ∈ ring-iso R S domain R domain S*
shows *monic-poly S (map h f)*
⟨*proof*⟩

lemma *monic-irreducible-poly-hom*:
assumes *monic-irreducible-poly R f*
assumes *h ∈ ring-iso R S domain R domain S*
shows *monic-irreducible-poly S (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 $f \in$ *carrier* $(K[X])$ *degree* $f > 0$
shows *embed-inj*: *inj-on* (*rupture-surj K f* ∘ *poly-of-const*) *K*
and *rupture-order*: *order* (*Rupt K f*) = *card K^degree f*
and *rupture-char*: *char* (*Rupt K f*) = *char R*
⟨*proof*⟩

definition *gauss-poly* **where**
gauss-poly K n = X_K [^]$_{poly\text{-}ring\ K}$ (*n*::*nat*) ⊖$_{poly\text{-}ring\ K}$ 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* :: *nat*
assumes *l* > *0*
shows (*X* [^]$_P$ *l* ⊖$_P$ **1**$_P$) *pdivides* (*X* [^]$_P$ *m* ⊖$_P$ **1**$_P$) ⟷ *l dvd m*
(**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma *gauss-poly-factor*:
assumes *n* > *0*
shows *gauss-poly R n* = (*X* [^]$_P$ (*n*−*1*) ⊖$_P$ **1**$_P$) ⊗$_P$ *X* (**is** *- = ?rhs*)
⟨*proof*⟩

lemma *var-neq-zero*: *X* ≠ **0**$_P$
⟨*proof*⟩

lemma *var-pow-eq-one-iff*: *X* [^]$_P$ *k* = **1**$_P$ ⟷ *k* = (*0*::*nat*)
⟨*proof*⟩

lemma *gauss-poly-carr*: *gauss-poly R n* ∈ *carrier P*
⟨*proof*⟩

lemma *gauss-poly-degree*:
assumes $n > 1$
shows *degree* (*gauss-poly R n*) = n
⟨*proof*⟩

lemma *gauss-poly-not-zero*:
assumes $n > 1$
shows *gauss-poly R n* ≠ $\mathbf{0}_P$
⟨*proof*⟩

lemma *gauss-poly-monic*:
assumes $n > 1$
shows *monic-poly R* (*gauss-poly R n*)
⟨*proof*⟩

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

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 :: *int*
fixes $l\ m$ :: *nat*
assumes $l > 0\ a > 1$
shows $(a\ \hat{}\ l - 1)$ *dvd* $(a\ \hat{}\ m - 1) \longleftrightarrow l$ *dvd* m
(**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma *gauss-poly-div-gauss-poly-iff*:
assumes $m > 0\ n > 0\ a > 1$
shows *gauss-poly R* $(a\hat{}n)$ $pdivides_R$ *gauss-poly R* $(a\hat{}m)$
⟷ n *dvd* m (**is** *?lhs=?rhs*)
⟨*proof*⟩

end

context *finite-field*
begin

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

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

shows f *pdivides*$_R$ *gauss-poly* R (*order* $R\hat{}n$) $\longleftrightarrow$ *degree* f *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
pmult$_R$ f (*gauss-poly* R (*order* $R\hat{}n$)) = *of-bool* (*degree* f *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 *order* $R\hat{}n = (\sum d \mid d\ dvd\ n.\ d *$
card $\{f.\ monic\text{-}irreducible\text{-}poly\ R\ f \wedge degree\ f = d\}$)
(**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 $n * card\ \{f.\ monic\text{-}irreducible\text{-}poly\ R\ f \wedge degree\ f = n\} =$
$(\sum d \mid d\ dvd\ n.\ moebius\text{-}mu\ d * (order\ R \hat{}\ (n\ div\ d)))$
(**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*:
$card\ \{f.\ monic\text{-}irreducible\text{-}poly\ R\ f \wedge degree\ f = 1\} = order\ R$
⟨*proof*⟩

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

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

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

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

theorem *existence*:
assumes $n > 0$
assumes *Factorial-Ring.prime p*
shows $\exists (F::\ int\ set\ list\ set\ ring).\ finite\text{-}field\ F \wedge order\ F = p \hat{}\ n$
⟨*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 $p \equiv char\ R$
  assumes $f \in carrier\ (poly\text{-}ring\ (ZFact\ p))$
  assumes $pirreducible_{(ZFact\ p)}\ (carrier\ (ZFact\ p))\ f$
  assumes $order\ R = p \hat{} degree\ f$
  assumes $x \in carrier\ R$
  assumes $eval\ (map\ (char\text{-}iso\ R)\ f)\ x = \mathbf{0}$
  shows $ring\text{-}hom\text{-}ring\ (Rupt_{(ZFact\ p)}\ (carrier\ (ZFact\ p))\ f)\ R$
    $(\lambda g.\ the\text{-}elem\ ((\lambda g'.\ eval\ (map\ (char\text{-}iso\ R)\ g')\ x)\ `\ g))$
⟨*proof*⟩

lemma (**in** *domain*) *pdivides-consistent*:
  assumes $subfield\ K\ R\ f \in carrier\ (K[X])\ g \in carrier\ (K[X])$
  shows $f\ pdivides\ g \longleftrightarrow f\ pdivides_{R\ (\!|\ carrier := K\ |\!)}\ g$
⟨*proof*⟩

lemma (**in** *finite-field*) *find-root*:
  assumes $subfield\ K\ R$
  assumes $monic\text{-}irreducible\text{-}poly\ (R\ (\!|\ carrier := K\ |\!))\ f$
  assumes $order\ R = card\ K \hat{} degree\ f$
  obtains x **where** $eval\ f\ x = \mathbf{0}\ x \in carrier\ R$
⟨*proof*⟩

lemma (**in** *finite-field*) *find-iso-from-zfact*:
  defines $p \equiv int\ (char\ R)$
  assumes $monic\text{-}irreducible\text{-}poly\ (ZFact\ p)\ f$
  assumes $order\ R = char\ R \hat{} degree\ f$
  shows $\exists \varphi.\ \varphi \in ring\text{-}iso\ (Rupt_{(ZFact\ p)}\ (carrier\ (ZFact\ p))\ f)\ R$
⟨*proof*⟩

theorem *uniqueness*:
  assumes $finite\text{-}field\ F_1$
  assumes $finite\text{-}field\ F_2$
  assumes $order\ F_1 = order\ F_2$
  shows $F_1 \simeq F_2$
⟨*proof*⟩

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* :: - ⇒ *'a list* ⇒ *'a list* ⇒ *bool* (‹*pcoprime*1›)
where $pcoprime_R\ p\ q =$
$(\forall r \in carrier\ (poly\text{-}ring\ R).\ r\ pdivides_R\ p \wedge r\ pdivides_R\ q \longrightarrow degree\ r = 0)$

lemma *pcoprimeI*:
assumes $\bigwedge r.\ r \in carrier\ (poly\text{-}ring\ R) \Longrightarrow r\ pdivides_R\ p \Longrightarrow r\ pdivides_R\ q \Longrightarrow degree\ r = 0$
shows $pcoprime_R\ p\ q$
⟨*proof*⟩

context *field*
begin

interpretation *r*:*polynomial-ring R* (*carrier R*)
⟨*proof*⟩

lemma *pcoprime-one*: $pcoprime_R\ p\ \mathbf{1}_{poly\text{-}ring\ R}$
⟨*proof*⟩

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

lemma *pcoprime-sym*:
shows *pcoprime x y* = *pcoprime y x*
⟨*proof*⟩

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

lemma *pcoprime-left-assoc-cong*:

assumes *x1* ∈ *carrier* (*poly-ring R*) *x2* ∈ *carrier* (*poly-ring R*)
assumes *x1* $\sim_{\textit{poly-ring } R}$ *x2*
assumes *y* ∈ *carrier* (*poly-ring R*)
shows *pcoprime x1 y* = *pcoprime x2 y*
⟨*proof*⟩

lemma *pcoprime-right-assoc-cong*:
assumes *x1* ∈ *carrier* (*poly-ring R*) *x2* ∈ *carrier* (*poly-ring R*)
assumes *x1* $\sim_{\textit{poly-ring } R}$ *x2*
assumes *y* ∈ *carrier* (*poly-ring R*)
shows *pcoprime y x1* = *pcoprime y x2*
⟨*proof*⟩

lemma *pcoprime-step*:
assumes *f* ∈ *carrier* (*poly-ring R*)
assumes *g* ∈ *carrier* (*poly-ring R*)
shows *pcoprime f g* ⟷ *pcoprime g* (*f pmod g*)
⟨*proof*⟩

lemma *pcoprime-zero-iff*:
assumes *f* ∈ *carrier* (*poly-ring R*)
shows *pcoprime f* [] ⟷ *length f* = *1*
⟨*proof*⟩

end

context *finite-field*
begin

interpretation *r*:*polynomial-ring R* (*carrier R*)
⟨*proof*⟩

lemma *exists-irreducible-proper-factor*:
assumes *monic-poly R f degree f* > *0* ¬*monic-irreducible-poly R f*
shows ∃ *g*. *monic-irreducible-poly R g* ∧ *g pdivides*$_R$ *f* ∧ *degree g* < *degree f*
⟨*proof*⟩

theorem *rabin-irreducibility-condition*:
assumes *monic-poly R f degree f* > *0*
defines *N* ≡ {*degree f div p* | *p* . *Factorial-Ring.prime p* ∧ *p dvd degree f*}
shows *monic-irreducible-poly R f* ⟷
(*f pdivides gauss-poly R* (*order R*^*degree f*) ∧ (∀ *n* ∈ *N*. *pcoprime* (*gauss-poly R* (*order R*^*n*)) *f*))
(**is** *?L* ⟷ *?R1* ∧ *?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 $f \in carrier\ (poly\text{-}ring\ R)$ $degree\ f > 0$
defines $N \equiv \{degree\ f\ div\ p \mid p\ .\ Factorial\text{-}Ring.prime\ p \wedge p\ dvd\ degree\ f\}$
shows $pirreducible\ (carrier\ R)\ f \longleftrightarrow$
$(f\ pdivides\ gauss\text{-}poly\ R\ (order\ R\hat{}degree\ f) \wedge (\forall n \in N.\ pcoprime\ (gauss\text{-}poly\ R\ (order\ R\hat{}n))\ f))$
(**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 $(\oplus_{residue\text{-}ring\ (10::'c)}{}^{100})$ 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 $(\ominus_{residue\text{-}ring\ n}\ x)\ y$ (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 $'a\ idx\text{-}ring =$
$idx\text{-}pred :: 'a \Rightarrow bool$
$idx\text{-}uminus :: 'a \Rightarrow 'a$

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* (‹0_{C^1}›)
notation *idx-one* (‹1_{C^1}›)
notation *idx-plus* (**infixl** ‹$+_{C^1}$› *65*)
notation *idx-mult* (**infixl** ‹$*_{C^1}$› *70*)
notation *idx-uminus* (‹$-_{C^1}$ -› [*81*] *80*)
notation *idx-udivide* (‹- $^{-1}{}_{C^1}$› [*81*] *80*)
notation *idx-pow* (**infixr** ‹ $\hat{}_{C^1}$› *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* $*_{C\,A}$ *y*),
one = $1_{C\,A}$,
zero = $0_{C\,A}$,
add = (λ *x y*. *x* $+_{C\,A}$ *y*) ⦈

definition $ring_C$ **where**
$ring_C$ *A* = (*ring* (*ring-of A*) ∧ (∀ *x*. *idx-pred A x* ⟶ $-_{C\,A}$ *x* = $\ominus_{ring\text{-}of\ A}$ *x*) ∧
(∀ *x*. *x* ∈ *Units* (*ring-of A*) ⟶ *x* $^{-1}{}_{C\,A}$ = $inv_{ring\text{-}of\ A}$ *x*))

lemma *ring-cD-aux*:
x $\hat{}_{C\,A}$ *n* = *x* $[\hat{}]_{ring\text{-}of\ A}$ *n*
⟨*proof*⟩

lemma *ring-cD*:
assumes $ring_C$ *A*
shows
$0_{C\,A}$ = $\mathbf{0}_{ring\text{-}of\ A}$
$1_{C\,A}$ = $\mathbf{1}_{ring\text{-}of\ A}$
⋀*x y*. *x* $*_{C\,A}$ *y* = *x* $\otimes_{ring\text{-}of\ A}$ *y*

$\bigwedge x\ y.\ x +_{C\ A} y = x \oplus_{ring\text{-}of\ A} y$
$\bigwedge x.\ x \in carrier\ (ring\text{-}of\ A) \Longrightarrow\ -_{C\ A}\ x = \ominus_{ring\text{-}of\ A}\ x$
$\bigwedge x.\ x \in Units\ (ring\text{-}of\ A) \Longrightarrow\ x\ ^{-1}{}_{C\ A} = inv_{ring\text{-}of\ A}\ x$
$\bigwedge x.\ x\ \widehat{}_{C\ A}\ n = x\ [\widehat{}]_{ring\text{-}of\ A}\ n$
⟨*proof*⟩

lemma *ring-cI*:
assumes *ring* (*ring-of A*)
assumes $\bigwedge x.\ x \in carrier\ (ring\text{-}of\ A) \Longrightarrow\ -_{C\ A}\ x = \ominus_{ring\text{-}of\ A}\ x$
assumes $\bigwedge x.\ x \in Units\ (ring\text{-}of\ A) \Longrightarrow\ x^{-1}{}_{C\ A} = inv_{ring\text{-}of\ A}\ x$
shows $ring_C\ A$
⟨*proof*⟩

definition $cring_C$ **where** $cring_C\ A = (ring_C\ A \wedge cring\ (ring\text{-}of\ A))$

lemma *cring-cI*:
assumes *cring* (*ring-of A*)
assumes $\bigwedge x.\ x \in carrier\ (ring\text{-}of\ A) \Longrightarrow\ -_{C\ A}\ x = \ominus_{ring\text{-}of\ A}\ x$
assumes $\bigwedge x.\ x \in Units\ (ring\text{-}of\ A) \Longrightarrow x^{-1}{}_{C\ A} = inv_{ring\text{-}of\ A}\ x$
shows $cring_C\ A$
⟨*proof*⟩

lemma *cring-c-imp-ring*: $cring_C\ A \Longrightarrow ring_C\ A$
⟨*proof*⟩

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

definition $domain_C$ **where** $domain_C\ A = (cring_C\ A \wedge domain\ (ring\text{-}of\ A))$

lemma *domain-cI*:
assumes *domain* (*ring-of A*)
assumes $\bigwedge x.\ x \in carrier\ (ring\text{-}of\ A) \Longrightarrow\ -_{C\ A}\ x = \ominus_{ring\text{-}of\ A}\ x$
assumes $\bigwedge x.\ x \in Units\ (ring\text{-}of\ A) \Longrightarrow x^{-1}{}_{C\ A} = inv_{ring\text{-}of\ A}\ x$
shows $domain_C\ A$
⟨*proof*⟩

lemma *domain-c-imp-ring*: $domain_C\ A \Longrightarrow ring_C\ A$
⟨*proof*⟩

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

definition $field_C$ **where** $field_C\ A = (domain_C\ A \wedge field\ (ring\text{-}of\ A))$

lemma *field-cI*:
assumes *field* (*ring-of A*)
assumes $\bigwedge x.\ x \in carrier\ (ring\text{-}of\ A) \Longrightarrow\ -_{C\ A}\ x = \ominus_{ring\text{-}of\ A}\ x$
assumes $\bigwedge x.\ x \in Units\ (ring\text{-}of\ A) \Longrightarrow\ x^{-1}{}_{C\ A} = inv_{ring\text{-}of\ A}\ x$

shows $field_C\ A$
⟨*proof*⟩

lemma *field-c-imp-ring*: $field_C\ A \Longrightarrow ring_C\ A$
⟨*proof*⟩

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

definition $enum_C$ **where** $enum_C\ A = ($
finite (*carrier* (*ring-of A*)) ∧
idx-size A = *order* (*ring-of A*) ∧
bij-betw (*idx-enum A*) {..<*order* (*ring-of A*)} (*carrier* (*ring-of A*)) ∧
(∀ *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 ⋀*x*. *x* < *order* (*ring-of A*) ⟹ *idx-enum-inv A* (*idx-enum A x*) = *x*
shows $enum_C\ A$
⟨*proof*⟩

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 ⋀*x*. *x* < *order* (*ring-of R*) ⟹ *idx-enum-inv R* (*idx-enum R x*) = *x*
and ⋀*x*. *x* ∈ *carrier* (*ring-of R*) ⟹ *idx-enum R* (*idx-enum-inv R x*) = *x*
⟨*proof*⟩

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 = (λ*x*. (*x* = [] ∨ *hd x* ≠ $0_{C\,E}$) ∧ *list-all* (*idx-pred E*) *x*),
idx-uminus = (λ*x*. *map* (*idx-uminus E*) *x*),
idx-plus = *o-poly-add E*,
idx-udivide = (λ*x*. [*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* = $X_{ring\text{-}of\ R}$
⟨*proof*⟩

fun *poly-eval* :: ('*a*,'*b*) *idx-ring-scheme* ⇒ '*a list* ⇒ '*a* ⇒ '*a*
where *poly-eval R fs x* = *fold* (λ*a 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* ∈ *carrier* (*ring-of* (*poly R*)) **and** *xc*:*x* ∈ *carrier* (*ring-of R*)
shows *poly-eval R fs x* = *ring.eval* (*ring-of R*) *fs x*
⟨*proof*⟩

lemma *poly-domain*:
assumes $domain_C$ *A*
shows $domain_C$ (*poly A*)
⟨*proof*⟩

function $long\text{-}division_C$ :: ('*a*,'*b*) *idx-ring-scheme* ⇒ '*a list* ⇒ '*a list* ⇒ '*a list* × '*a list*
where $long\text{-}division_C$ *F f g* = (
if (*length g* = *0* ∨ *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 (λ*x*. $x\ +_{C\,poly\ F}\ -_{C\,poly\ F}\ h$) ($long\text{-}division_C$ *F f*'' *g*)))
⟨*proof*⟩

lemma *pmod-termination-helper*:
g ≠ [] ⟹ ¬*length f* < *length g* ⟹ *min x* (*length f* − *1*) < *length f*
⟨*proof*⟩

termination ⟨*proof*⟩

declare $long\text{-}division_C.simps$[*simp del*]

lemma *long-division-c-length*:
assumes *length g* > *0*
shows *length* (*snd* ($long\text{-}division_C$ *R f g*)) < *length g*
⟨*proof*⟩

context *field*
begin

interpretation *r*:*polynomial-ring R* (*carrier R*)
⟨*proof*⟩

lemma *poly-length-from-coeff*:
assumes *p* ∈ *carrier* (*poly-ring R*)
assumes ⋀*i*. *i* ≥ *k* ⟹ *coeff p i* = **0**
shows *length p* ≤ *k*
⟨*proof*⟩

lemma *poly-add-cancel-len*:
assumes $f \in$ *carrier* (*poly-ring* R) $-$ $\{\mathbf{0}_{\textit{poly-ring } R}\}$
assumes $g \in$ *carrier* (*poly-ring* R) $-$ $\{\mathbf{0}_{\textit{poly-ring } R}\}$
assumes *hd* $f = \ominus$ *hd* g *degree* $f =$ *degree* g
shows *length* $(f \oplus_{\textit{poly-ring } R} g) <$ *length* f
⟨*proof*⟩

lemma *pmod-mult-left*:
assumes $f \in$ *carrier* (*poly-ring* R)
assumes $g \in$ *carrier* (*poly-ring* R)
assumes $h \in$ *carrier* (*poly-ring* R)
shows $(f \otimes_{\textit{poly-ring } R} g)$ *pmod* $h = ((f \textit{ pmod } h) \otimes_{\textit{poly-ring } R} g)$ *pmod* h (**is** *?L* = *?R*)
⟨*proof*⟩

lemma *pmod-mult-right*:
assumes $f \in$ *carrier* (*poly-ring* R)
assumes $g \in$ *carrier* (*poly-ring* R)
assumes $h \in$ *carrier* (*poly-ring* R)
shows $(f \otimes_{\textit{poly-ring } R} g)$ *pmod* $h = (f \otimes_{\textit{poly-ring } R} (g \textit{ pmod } h))$ *pmod* h (**is** *?L* = *?R*)
⟨*proof*⟩

lemma *pmod-mult-both*:
assumes $f \in$ *carrier* (*poly-ring* R)
assumes $g \in$ *carrier* (*poly-ring* R)
assumes $h \in$ *carrier* (*poly-ring* R)
shows $(f \otimes_{\textit{poly-ring } R} g)$ *pmod* $h = ((f \textit{ pmod } h) \otimes_{\textit{poly-ring } R} (g \textit{ pmod } h))$ *pmod* h
(**is** *?L* = *?R*)
⟨*proof*⟩

lemma *field-Unit-minus-closed*:
assumes $x \in$ *Units* R
shows $\ominus x \in$ *Units* R
⟨*proof*⟩

end

lemma *long-division-c*:
assumes $\textit{field}_C$ R
assumes $f \in$ *carrier* (*poly-ring* (*ring-of* R))
assumes $g \in$ *carrier* (*poly-ring* (*ring-of* R))
shows $\textit{long-division}_C$ R f g = (*ring.pdiv* (*ring-of* R) f g, *ring.pmod* (*ring-of* R) f g)
⟨*proof*⟩

definition $\textit{pdiv}_C$ :: ('*a*,'*b*) *idx-ring-scheme* ⇒ '*a list* ⇒ '*a list* ⇒ '*a*

list **where**
  $pdiv_C\ R\ f\ g = fst\ (long\text{-}division_C\ R\ f\ g)$

lemma *pdiv-c*:
  assumes $field_C\ R$
  assumes $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
  assumes $g \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
  shows $pdiv_C\ R\ f\ g = ring.pdiv\ (ring\text{-}of\ R)\ f\ g$
  ⟨*proof*⟩

definition $pmod_C :: ('a,'b)\ idx\text{-}ring\text{-}scheme \Rightarrow 'a\ list \Rightarrow 'a\ list \Rightarrow 'a$ *list* **where**
  $pmod_C\ R\ f\ g = snd\ (long\text{-}division_C\ R\ f\ g)$

lemma *pmod-c*:
  assumes $field_C\ R$
  assumes $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
  assumes $g \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
  shows $pmod_C\ R\ f\ g = ring.pmod\ (ring\text{-}of\ R)\ f\ g$
  ⟨*proof*⟩

function *ext-euclidean* ::
  $('a,'b)\ idx\text{-}ring\text{-}scheme \Rightarrow 'a\ list \Rightarrow 'a\ list \Rightarrow ('a\ list \times 'a\ list) \times 'a$ *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 $(p,q) = long\text{-}division_C\ F\ f\ g$;
          $((u,v),r) = ext\text{-}euclidean\ F\ g\ q$
       in $((v, u +_{C\ poly\ F} (-_{C\ poly\ F} (p *_{C\ poly\ F} v))), r)))$
  ⟨*proof*⟩

termination
  ⟨*proof*⟩

lemma (**in** *domain*) *pdivides-self*:
  assumes $x \in carrier\ (poly\text{-}ring\ R)$
  shows *x pdivides x*
⟨*proof*⟩

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

lemma *ext-euclidean*:
  assumes $field_C\ R$
  defines $P \equiv poly\text{-}ring\ (ring\text{-}of\ R)$
  assumes $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
  assumes $g \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$

defines $r \equiv$ *ext-euclidean R f g*
shows *snd r* $= f \otimes_P$ (*fst* (*fst r*)) $\oplus_P g \otimes_P$ (*snd* (*fst r*)) (**is** *?T1*)
and *snd r pdivides*$_{\text{ring-of } R}$ *f* (**is** *?T2*) *snd r pdivides*$_{\text{ring-of } R}$ *g* (**is** *?T3*)
and {*snd r*, *fst* (*fst r*), *snd* (*fst r*)} $\subseteq$ *carrier P* (**is** *?T4*)
and *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* :: *nat* ⇒ *nat idx-ring-enum*
where *mod-ring n* = (|
idx-pred = (λ*x*. *x* < *n*),
idx-uminus = (λ*x*. (*n*−*x*) *mod n*),
idx-plus = (λ*x y*. (*x*+*y*) *mod n*),
idx-udivide = (λ*x*. *nat* (*fst* (*bezout-coefficients* (*int x*) (*int n*)) *mod* (*int n*))),
idx-mult = (λ*x y*. (*x*∗*y*) *mod n*),
idx-zero = *0*,
idx-one = *1*,
idx-size = *n*,
idx-enum = *id*,
idx-enum-inv = *id*
|)

lemma *zfact-iso-0*:
assumes *n* > *0*
shows *zfact-iso n 0* = $\mathbf{0}_{ZFact\ (int\ n)}$
⟨*proof*⟩

lemma *zfact-prime-is-field*:
assumes *Factorial-Ring.prime* (*p* :: *nat*)
shows *field* (*ZFact* (*int p*))
⟨*proof*⟩

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

lemma *zfact-iso-inv-0*:
assumes *n-ge-0*: *n* > *0*
shows *zfact-iso-inv n* $\mathbf{0}_{ZFact\ (int\ n)}$ = *0*
⟨*proof*⟩

lemma *zfact-coset*:
assumes *n-ge-0*: $n > 0$
assumes $x \in$ *carrier* (*ZFact* (*int* n))
defines $I \equiv Idl_{\mathcal{Z}}$ {*int* n}
shows $x = I +>_{\mathcal{Z}}$ (*int* (*zfact-iso-inv* n x))
⟨*proof*⟩

lemma *zfact-iso-inv-bij*:
assumes $n > 0$
shows *bij-betw* (*zfact-iso-inv* n) (*carrier* (*ZFact* (*int* n))) (*carrier* (*ring-of* (*mod-ring* n)))
⟨*proof*⟩

lemma *zfact-iso-inv-is-ring-iso*:
fixes n :: *nat*
assumes *n-ge-1*: $n > 1$
shows *zfact-iso-inv* $n \in$ *ring-iso* (*ZFact* (*int* n)) (*ring-of* (*mod-ring* n)) (**is** *?f* ∈ -)
⟨*proof*⟩

lemma *mod-ring-finite*:
finite (*carrier* (*ring-of* (*mod-ring* n)))
⟨*proof*⟩

lemma *mod-ring-carr*:
$x \in$ *carrier* (*ring-of* (*mod-ring* n)) $\longleftrightarrow$ $x < n$
⟨*proof*⟩

lemma *mod-ring-is-cring*:
assumes *n-ge-1*: $n > 1$
shows *cring* (*ring-of* (*mod-ring* n))
⟨*proof*⟩

lemma *zfact-iso-is-ring-iso*:
assumes *n-ge-1*: $n > 1$
shows *zfact-iso* $n \in$ *ring-iso* (*ring-of* (*mod-ring* n)) (*ZFact* (*int* n))
⟨*proof*⟩

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

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

lemma *mod-ring-is-ring-c*:
assumes $n > 1$
shows $cring_C$ (*mod-ring* n)
⟨*proof*⟩

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

lemma *mod-ring-is-enum-c*:
shows *enum*$_C$ (*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 *pcoprime*$_C$ :: ('*a*, '*b*) *idx-ring-scheme* ⇒ '*a list* ⇒ '*a list* ⇒ *bool*
where *pcoprime*$_C$ *R f g* = (*length* (*snd* (*ext-euclidean R f g*)) = *1*)

declare *pcoprime*$_C$.*simps*[*simp del*]

lemma *pcoprime-c*:
assumes *field*$_C$ *R*
assumes *f* ∈ *carrier* (*poly-ring* (*ring-of R*))
assumes *g* ∈ *carrier* (*poly-ring* (*ring-of R*))
shows *pcoprime*$_C$ *R f g* ⟷ *pcoprime*$_{ring\text{-}of\ R}$ *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-pow*$_C$ :: ('*a*,'*b*) *idx-ring-scheme* ⇒ '*a list* ⇒ *nat* ⇒ '*a list* ⇒ '*a list*
where *pmod-pow*$_C$ *F f n g* = (
let r = (*if n* ≥ *2 then pmod-pow*$_C$ *F f* (*n div 2*) *g* ^$_{C\,poly\ F}$ *2 else* *1*$_{C\,poly\ F}$)
in pmod$_C$ *F* (*r* *$_{C\,poly\ F}$ (*f* ^$_{C\,poly\ F}$ (*n mod 2*))) *g*)

declare *pmod-pow*$_C$.*simps*[*simp del*]

lemma *pmod-pow-c*:
assumes *field*$_C$ *R*

assumes $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
assumes $g \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
shows $pmod\text{-}pow_C\ R\ f\ n\ g = ring.pmod\ (ring\text{-}of\ R)\ (f\ [\hat{}]_{poly\text{-}ring\ (ring\text{-}of\ R)}\ n)\ g$
⟨*proof*⟩

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

definition *pcoprime-with-gauss-poly* :: ('*a*,'*b*) *idx-ring-scheme* ⇒ '*a list* ⇒ *nat* ⇒ *bool*
where *pcoprime-with-gauss-poly F p n* =
$(pcoprime_C\ F\ p\ (pmod\text{-}pow_C\ F\ X_{C\,F}\ n\ p\ +_{C\,poly\ F}\ (-_{C\,poly\ F}\ pmod_C\ F\ X_{C\,F}\ p)))$

definition *divides-gauss-poly* :: ('*a*,'*b*) *idx-ring-scheme* ⇒ '*a list* ⇒ *nat* ⇒ *bool*
where *divides-gauss-poly F p n* =
$(pmod\text{-}pow_C\ F\ X_{C\,F}\ n\ p\ +_{C\,poly\ F}\ (-_{C\,poly\ F}\ pmod_C\ F\ X_{C\,F}\ p) = [])$

lemma *mod-gauss-poly*:
assumes $field_C\ R$
assumes $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
shows $pmod\text{-}pow_C\ R\ X_{C\,R}\ n\ f\ +_{C\,poly\ R}\ (-_{C\,poly\ R}\ pmod_C\ R\ X_{C\,R}\ f) =$
$ring.pmod\ (ring\text{-}of\ R)\ (gauss\text{-}poly\ (ring\text{-}of\ R)\ n)\ f\ (\mathbf{is}\ ?L = ?R)$
⟨*proof*⟩

lemma *pcoprime-with-gauss-poly*:
assumes $field_C\ R$
assumes $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
shows $pcoprime\text{-}with\text{-}gauss\text{-}poly\ R\ f\ n \longleftrightarrow pcoprime_{ring\text{-}of\ R}\ (gauss\text{-}poly\ (ring\text{-}of\ R)\ n)\ f$
$(\mathbf{is}\ ?L = ?R)$
⟨*proof*⟩

lemma *divides-gauss-poly*:
assumes $field_C\ R$
assumes $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$
shows $divides\text{-}gauss\text{-}poly\ R\ f\ n \longleftrightarrow f\ pdivides_{ring\text{-}of\ R}\ (gauss\text{-}poly\ (ring\text{-}of\ R)\ n)$
$(\mathbf{is}\ ?L = ?R)$
⟨*proof*⟩

fun *rabin-test-powers* :: ('*a*, '*b*) *idx-ring-enum-scheme* ⇒ *nat* ⇒ *nat list*
where *rabin-test-powers F n* =

map (λ*p*. *idx-size F*^(*n div p*)) (*filter* (λ*p*. *prime p* ∧ *p 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* ⇒ '*a list* ⇒ *bool*
where *rabin-test F f* = (
if degree f = *0 then*
False
else (*if* ¬*divides-gauss-poly F f* (*idx-size F*^*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*: *field*$_C$ *R*
assumes *enum-R*: *enum*$_C$ *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*)^(*n div p*) | *p* . *Factorial-Ring.prime p* ∧ *p dvd n*}
(**is** *?L* = *?R*)
⟨*proof*⟩

lemma *rabin-test*:
assumes *monic-poly* (*ring-of R*) *f*
shows *rabin-test R f* ⟷ *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 $x < b\hat{}k$
shows *nth-digit x k b* = *0*
⟨*proof*⟩

lemma *nth-digit-bounded′*:
assumes $b > 0$
shows *nth-digit v x b* < *b*
⟨*proof*⟩

lemma *digit-gen-sum-repr′*:
assumes $n < b\hat{}c$
shows $n = (\sum k{<}c.\ \textit{nth-digit}\ n\ k\ b * b\ \hat{}\ k)$
⟨*proof*⟩

lemma
assumes $\bigwedge x.\ x \in A \Longrightarrow f\ (g\ x) = x$
shows $\bigwedge y.\ y \in g\ `\ A \Longrightarrow g\ (f\ y) = y$
⟨*proof*⟩

lemma *nth-digit-bij*:
bij-betw (λ*v*. (λ*x*∈{..<*n*}. *nth-digit v x b*)) {..<*b*^*n*} ({..<*n*} →$_E$ {..<*b*})
(**is** *bij-betw ?f ?A ?B*)
⟨*proof*⟩

lemma *nth-digit-sum*:
assumes $\bigwedge i.\ i < l \Longrightarrow f\ i < b$
shows $\bigwedge k.\ k < l \Longrightarrow \textit{nth-digit}\ (\sum i{<}\ l.\ f\ i * b\hat{}i)\ k\ b = f\ k$
and $(\sum i{<}l.\ f\ i * b\hat{}i) < b\hat{}l$
⟨*proof*⟩

lemma *bij-betw-reindex*:
assumes *bij-betw f I J*
shows *bij-betw* (λ*x*. λ*i*∈*I*. *x* (*f i*)) (*J* →$_E$ *S*) (*I* →$_E$ *S*)
⟨*proof*⟩

lemma *lift-bij-betw*:
assumes *bij-betw f S T*
shows *bij-betw* (λ*x*. λ*i*∈*I*. *f* (*x i*)) (*I* →$_E$ *S*) (*I* →$_E$ *T*)
⟨*proof*⟩

lemma *lists-bij*:
bij-betw (λ*x*. *map x* [*0*..<*d*]) ({..<*d*} →$_E$ *S*) {*x*. *set x* ⊆ *S* ∧ *length x* = *d*}
⟨*proof*⟩

lemma *bij-betw-prod*: *bij-betw* ($\lambda x.$ (x *mod* s, x *div* s)) {..<$s * t$} ({..<(s::*nat*)} × {..<t})
⟨*proof*⟩

end

14 Additional results about PMFs

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

lemma *powr-mono-rev*:
fixes x :: *real*
assumes $a \leq b$ **and** $x > 0$ $x \leq 1$
shows x *powr* $b \leq x$ *powr* a
⟨*proof*⟩

lemma *integral-bind-pmf*:
fixes f :: - ⇒ *real*
assumes *bounded* (f ‘ *set-pmf* (*bind-pmf* p q))
shows ($\int x.$ f x ∂*bind-pmf* p q) = ($\int x.$ $\int y.$ f y ∂q x ∂p) (**is** *?L* = *?R*)
⟨*proof*⟩

lemma *measure-bind-pmf*:
measure (*bind-pmf* m f) s = ($\int x.$ *measure* (f x) s ∂m) (**is** *?L* = *?R*)
⟨*proof*⟩

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

lemma *replicate-drop-while-cancel*:

assumes $k = length\ (takeWhile\ ((=)\ x)\ y)$
shows $replicate\ k\ x\ @\ dropWhile\ ((=)\ x)\ y = y$ (**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*: $list\text{-}all\ p\ xs \Longrightarrow list\text{-}all\ p\ (dropWhile\ q\ xs)$
⟨*proof*⟩

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

definition *poly-enum-inv* :: ('a,'b) *idx-ring-enum-scheme* ⇒ *nat* ⇒ 'a *list* ⇒ *nat*
where $poly\text{-}enum\text{-}inv\ R\ l\ f =$
$(let\ f' = replicate\ (l - length\ f)\ 0_{C\,R}\ @\ f\ in$
$(\sum i{<}l.\ idx\text{-}enum\text{-}inv\ R\ (f'\ !\ (l - 1 - i)) * idx\text{-}size\ R\ \hat{}i\ ))$

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

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

definition *poly-mod-ring* :: ('a,'b) *idx-ring-enum-scheme* ⇒ 'a *list* => 'a *list idx-ring-enum*
where $poly\text{-}mod\text{-}ring\ R\ f = (\!|$
$idx\text{-}pred = (\lambda xs.\ idx\text{-}pred\ (poly\ R)\ xs \wedge length\ xs \leq degree\ f),$
$idx\text{-}uminus = idx\text{-}uminus\ (poly\ R),$
$idx\text{-}plus = (\lambda x\ y.\ pmod_C\ R\ (x +_{C\,poly\ R}\ y)\ f),$
$idx\text{-}udivide = (\lambda x.\ let\ ((u,v),r) = ext\text{-}euclidean\ R\ x\ f\ in\ pmod_C\ R\ (r^{-1}{}_{C\,poly\ R} *_{C\,poly\ R}\ u)\ f),$
$idx\text{-}mult = (\lambda x\ y.\ pmod_C\ R\ (x *_{C\,poly\ R}\ y)\ f),$
$idx\text{-}zero = 0_{C\,poly\ R},$
$idx\text{-}one = 1_{C\,poly\ R},$
$idx\text{-}size = idx\text{-}size\ R\ \hat{}\ degree\ f,$
$idx\text{-}enum = poly\text{-}enum\ R\ (degree\ f),$
$idx\text{-}enum\text{-}inv = poly\text{-}enum\text{-}inv\ R\ (degree\ f)\ |\!)$

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

definition *poly-mod-ring-iso-inv* :: ('a,'b) *idx-ring-enum-scheme* ⇒ 'a
list ⇒ 'a *list set* ⇒ '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) *idx-ring-enum-scheme*
assumes *field-R*: $field_C\ R$
assumes *f-carr*: *f* ∈ *carrier* (*poly-ring* (*ring-of R*))
assumes *deg-f*: *degree f* > *0*
begin

private abbreviation *P* **where** *P* ≡ *poly-ring* (*ring-of R*)
private abbreviation *I* **where** *I* ≡ $PIdl_{poly\text{-}ring\ (ring\text{-}of\ R)}\ f$

interpretation *field ring-of R*
⟨*proof*⟩

interpretation *d*: *domain P*
⟨*proof*⟩

interpretation *i*: *ideal I P*
⟨*proof*⟩

interpretation *s*: *ring-hom-ring P P Quot I* $(+>_P)$ *I*
⟨*proof*⟩

interpretation *cr*: *cring P Quot I*
⟨*proof*⟩

lemma *ring-c*: $ring_C\ R$
⟨*proof*⟩

lemma *d-poly*: $domain_C$ (*poly R*) ⟨*proof*⟩

lemma *ideal-mod*:
assumes *y* ∈ *carrier P*
shows *I* $+>_P$ (*pmod y f*) = *I* $+>_P$ *y*
⟨*proof*⟩

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

carrier (*ring-of* (*poly-mod-ring R f*)) = {*xs*. *xs* ∈ *carrier P* ∧ *degree xs* < *degree f*}
(**is** *?L* = *?R*)
⟨*proof*⟩

lemma *poly-mod-ring-carr*:
assumes *y* ∈ *carrier P*
shows *pmod y f* ∈ *carrier* (*ring-of* (*poly-mod-ring R f*))
⟨*proof*⟩

lemma *poly-mod-ring-iso-ran*:
poly-mod-ring-iso R f ‘ *carrier* (*ring-of* (*poly-mod-ring R f*)) = *carrier* (*P Quot I*)
⟨*proof*⟩

lemma *poly-mod-ring-iso-inj*:
inj-on (*poly-mod-ring-iso R f*) (*carrier* (*ring-of* (*poly-mod-ring R f*)))
⟨*proof*⟩

lemma *poly-mod-iso-ring-bij*:
bij-betw (*poly-mod-ring-iso R f*) (*carrier* (*ring-of* (*poly-mod-ring R f*))) (*carrier* (*P Quot I*))
⟨*proof*⟩

lemma *poly-mod-iso-ring-bij-2*:
bij-betw (*poly-mod-ring-iso-inv R f*) (*carrier* (*P Quot I*)) (*carrier* (*ring-of* (*poly-mod-ring R f*)))
⟨*proof*⟩

lemma *poly-mod-ring-iso-inv-1*:
assumes *x* ∈ *carrier* (*P Quot I*)
shows *poly-mod-ring-iso R f* (*poly-mod-ring-iso-inv R f x*) = *x*
⟨*proof*⟩

lemma *poly-mod-ring-iso-inv-2*:
assumes *x* ∈ *carrier* (*ring-of* (*poly-mod-ring R f*))
shows *poly-mod-ring-iso-inv R f* (*poly-mod-ring-iso R f x*) = *x*
⟨*proof*⟩

lemma *poly-mod-ring-add*:
assumes *x* ∈ *carrier P*
assumes *y* ∈ *carrier P*
shows $x \oplus_{\textit{ring-of}\ (\textit{poly-mod-ring}\ R\ f)}\ y = \textit{pmod}\ (x \oplus_P y)\ f$ (**is** *?L* = *?R*)
⟨*proof*⟩

lemma *poly-mod-ring-zero*: $\mathbf{0}_{\textit{ring-of}\ (\textit{poly-mod-ring}\ R\ f)} = \mathbf{0}_P$
⟨*proof*⟩

lemma *poly-mod-ring-one*: $\mathbf{1}_{\textit{ring-of }(\textit{poly-mod-ring } R\ f)} = \mathbf{1}_P$
⟨*proof*⟩

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

lemma *poly-mod-ring-iso-inv*:
poly-mod-ring-iso-inv R f ∈ *ring-iso* (*P Quot I*) (*ring-of* (*poly-mod-ring R f*))
(**is** *?f* ∈ *ring-iso ?S ?T*)
⟨*proof*⟩

lemma *cring-poly-mod-ring-1*:
shows *ring-of* (*poly-mod-ring R f*)(|*zero* := *poly-mod-ring-iso-inv R f* $\mathbf{0}_{P\ \textit{Quot}\ I}$|) =
ring-of (*poly-mod-ring R f*)
and *cring* (*ring-of* (*poly-mod-ring R f*))
⟨*proof*⟩

interpretation *cr-p*: *cring* (*ring-of* (*poly-mod-ring R f*))
⟨*proof*⟩

lemma *cring-c-poly-mod-ring*: $\textit{cring}_C$ (*poly-mod-ring R f*)
⟨*proof*⟩

end

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

lemma *enum-c-poly-mod-ring*:
assumes $\textit{enum}_C\ R\ \textit{ring}_C\ R$
shows $\textit{enum}_C$ (*poly-mod-ring R f*)
⟨*proof*⟩

end

16 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
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* ⇒ *nat* ⇒ *nat* ⇒ '*a list*
where *enum-monic-poly A d i* = $1_{C\,A}$#[*idx-enum A* (*nth-digit i j* (*idx-size A*)). *j* ← *rev* [*0*..<*d*]]

lemma *enum-monic-poly*:
assumes *field*$_C$ *R enum*$_C$ *R*
shows *bij-betw* (*enum-monic-poly R d*) {..<*order* (*ring-of R*)^*d*}
{*f*. *monic-poly* (*ring-of R*) *f* ∧ *degree f* = *d*}
⟨*proof*⟩

abbreviation *tick-spmf* :: ('*a* × *nat*) *spmf* ⇒ ('*a* × *nat*) *spmf*
where *tick-spmf* ≡ *map-spmf* (λ(*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* ⇒ *nat* ⇒ (*nat list* × *nat*) *spmf*
where
sample-irreducible-poly p n =
do {
k ← *spmf-of-set* {..<*p*^*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* ((λ*k*. ¬*rabin-test* (*mod-ring p*) (*f k*))) (λ*x*. *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* {..<*p*^*n*};
let poly = *enum-monic-poly* (*mod-ring p*) *n k*;
if rabin-test (*mod-ring p*) *poly then return-spmf* (*poly*,*c*) *else x*
} =
do {
poly ← *spmf-of-set S*;
if monic-irreducible-poly (*ring-of* (*mod-ring p*)) *poly*
then return-spmf (*poly*,*c*)

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*:
($\int^+ x$. *cost x ∂sample-irreducible-poly p n*) ≤ *2∗real n* (**is** *?L ≤ ?R*)
⟨*proof*⟩ **lemma** *weight-sample-irreducible-poly*:
weight-spmf (*sample-irreducible-poly p n*) = *1* (**is** *?L = ?R*)
⟨*proof*⟩

lemma *sample-irreducible-poly-result*:
map-spmf fst (*sample-irreducible-poly p n*) =
spmf-of-set {*f*. *monic-irreducible-poly* (*ring-of* (*mod-ring p*)) *f* ∧ *degree f = n*} (**is** *?L = ?R*)
⟨*proof*⟩

lemma *find-irreducible-poly-result*:
defines *res* ≡ *find-irreducible-poly p n*
shows *monic-irreducible-poly* (*ring-of* (*mod-ring p*)) *res degree res = n*
⟨*proof*⟩

lemma *monic-irred-poly-set-nonempty-finite*:
{*f*. *monic-irreducible-poly* (*ring-of* (*mod-ring p*)) *f* ∧ *degree f = n*} ≠ {} (**is** *?R1*)
finite {*f*. *monic-irreducible-poly* (*ring-of* (*mod-ring p*)) *f* ∧ *degree f = n*} (**is** *?R2*)
⟨*proof*⟩

end

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

definition *split-power* :: *nat* ⇒ *nat* × *nat*
where *split-power n* = (
let e = last (*filter* (λ*x*. *is-nth-power-nat x n*) (*1#*[*2*..<*floorlog 2 n*]))
in (*nth-root-nat e n*, *e*))

lemma *split-power-result*:
assumes $(x,e) = \textit{split-power}\ n$
shows $n = x \hat{} e \bigwedge k.\ n > 1 \Longrightarrow k > e \Longrightarrow \neg \textit{is-nth-power}\ k\ n$
⟨*proof*⟩

definition *not-perfect-power* :: *nat* ⇒ *bool*
where $\textit{not-perfect-power}\ n = (n > 1 \wedge (\forall x\ k.\ n = x \hat{} k \longrightarrow k = 1))$

lemma *is-nth-power-from-multiplicities*:
assumes $n > (0::nat)$
assumes $\bigwedge p.\ \textit{Factorial-Ring.prime}\ p \Longrightarrow k\ \textit{dvd}\ (\textit{multiplicity}\ p\ n)$
shows *is-nth-power k n*
⟨*proof*⟩

lemma *power-inj-aux*:
assumes *not-perfect-power a not-perfect-power b*
assumes *n > 0 m > n*
assumes $a \hat{} n = b \hat{} m$
shows *False*
⟨*proof*⟩

Generalization of *prime-power-inj′*

lemma *power-inj*:
assumes *not-perfect-power a not-perfect-power b*
assumes *n > 0 m > 0*
assumes $a \hat{} n = b \hat{} m$
shows $a = b \wedge n = m$
⟨*proof*⟩

lemma *split-power-base-not-perfect*:
assumes *n > 1*
shows *not-perfect-power* (*fst* (*split-power n*))
⟨*proof*⟩

lemma *prime-not-perfect*:
assumes *Factorial-Ring.prime p*
shows *not-perfect-power p*
⟨*proof*⟩

lemma *split-power-prime*:
assumes *Factorial-Ring.prime p n > 0*
shows *split-power* $(p \hat{} n) = (p,n)$
⟨*proof*⟩

definition $\textit{is-prime-power}\ n = (\exists p\ k.\ \textit{Factorial-Ring.prime}\ p \wedge k > 0 \wedge n = p \hat{} k)$

lemma *is-prime-powerI*:
assumes *prime p k* > *0*
shows *is-prime-power* (*p* ^ *k*)
⟨*proof*⟩

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

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

lemma *GF-in-GF-R*:
assumes *is-prime-power n*
shows *GF n* ∈ *set-spmf* (GF_R *n*)
⟨*proof*⟩

lemma *galois-field-random-1*:
assumes *is-prime-power n*
shows ⋀ω. ω ∈ *set-spmf* (GF_R *n*) ⟹ $enum_C$ ω ∧ $field_C$ ω ∧ *order* (*ring-of* ω) = *n*
and *lossless-spmf* (GF_R *n*)
⟨*proof*⟩

lemma *galois-field*:
assumes *is-prime-power n*
shows $enum_C$ (*GF n*) $field_C$ (*GF n*) *order* (*ring-of* (*GF n*)) = *n*
⟨*proof*⟩

lemma *lossless-imp-spmf-of-pmf*:
assumes *lossless-spmf M*
shows *spmf-of-pmf* (*map-pmf the M*) = *M*
⟨*proof*⟩

lemma *galois-field-random-2*:
assumes *is-prime-power n*
shows *map-spmf* (λω. $enum_C$ ω ∧ $field_C$ ω ∧ *order* (*ring-of* ω) = *n*) (GF_R *n*) = *return-spmf True*
(**is** *?L* = -)
⟨*proof*⟩

lemma *bind-galois-field-cong*:
assumes *is-prime-power n*
assumes $\bigwedge\omega$. *enum*$_C$ $\omega \Longrightarrow$ *field*$_C$ $\omega \Longrightarrow$ *order* (*ring-of* ω) $= n \Longrightarrow$ $f\ \omega = g\ \omega$
shows *bind-spmf* (*GF*$_R$ n) f = *bind-spmf* (*GF*$_R$ n) g
⟨*proof*⟩

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.