

# 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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Preliminary Results</b>	<b>3</b>
2.1	Summation in the discrete topology . . . . .	3
2.2	Polynomials . . . . .	5
2.3	Ring Isomorphisms . . . . .	11
2.4	Divisibility . . . . .	21
2.5	Factorization . . . . .	23
<b>3</b>	<b>Characteristic of Rings</b>	<b>34</b>
<b>4</b>	<b>Formal Derivatives</b>	<b>55</b>
<b>5</b>	<b>Factorization into Monic Polynomials</b>	<b>63</b>
<b>6</b>	<b>Counting Irreducible Polynomials</b>	<b>77</b>
6.1	The polynomial $X^n - X$ . . . . .	77
6.2	Gauss Formula . . . . .	96

<b>7</b>	<b>Isomorphism between Finite Fields</b>	<b>101</b>
<b>8</b>	<b>Rabin’s test for irreducible polynomials</b>	<b>108</b>
<b>9</b>	<b>Executable Structures</b>	<b>116</b>
<b>10</b>	<b>Executable Polynomial Rings</b>	<b>120</b>
<b>11</b>	<b>Executable Factor Rings</b>	<b>136</b>
<b>12</b>	<b>Executable Code for Rabin’s Irreducibility Test</b>	<b>143</b>
<b>13</b>	<b>Additional results about Bijections and Digit Representations</b>	<b>151</b>
<b>14</b>	<b>Additional results about PMFs</b>	<b>156</b>
<b>15</b>	<b>Executable Polynomial Factor Rings</b>	<b>157</b>
<b>16</b>	<b>Fast algorithms for Computations of Roots</b>	<b>174</b>
<b>17</b>	<b>Algorithms for finding irreducible polynomials</b>	<b>177</b>

## 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].<sup>1</sup>

---

<sup>1</sup>Thanks 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** –  
**have**  $c:finite \{i \in A. g\ i \neq 0\}$   
**using**  $assms(2)$   
**by** (*intro finite-subset[OF - assms(1)] subsetI, force*)  
**let**  $?B = \{i \in A. f\ i \neq 0 \vee g\ i \neq 0\}$   
  
**have**  $b:?B = \{i \in A. f\ i \neq 0\} \cup \{i \in A. g\ i \neq 0\}$   
**by** (*auto simp add:set-eq-iff*)  
**have**  $a:finite\ ?B$   
**using**  $assms(1)\ c$  **by** (*subst b, simp*)

**have**  $?lhs = \text{sum}' (\lambda i. f i - g i) ?B$   
 by *(intro sum.mono-neutral-cong-right', simp-all)*  
**also have**  $\dots = \text{sum} (\lambda i. f i - g i) ?B$   
 by *(intro sum.eq-sum a)*  
**also have**  $\dots = \text{sum } f ?B - \text{sum } g ?B$   
 using *assms(2)* by *(subst sum-subtractf-nat, auto)*  
**also have**  $\dots = \text{sum}' f ?B - \text{sum}' g ?B$   
 by *(intro arg-cong2[where f=(-)] sum.eq-sum[symmetric] a)*  
**also have**  $\dots = ?rhs$   
 by *(intro arg-cong2[where f=(-)] sum.mono-neutral-cong-left')*  
     *simp-all*  
**finally show** *?thesis*  
 by *simp*  
**qed**

**lemma** *sum'-nat-eq-0-iff*:

**fixes**  $f :: 'a \Rightarrow \text{nat}$   
**assumes** *finite*  $\{i \in A. f i \neq 0\}$   
**assumes**  $\text{sum}' f A = 0$   
**shows**  $\bigwedge i. i \in A \implies f i = 0$

**proof** –

**let**  $?B = \{i \in A. f i \neq 0\}$

**have**  $\text{sum } f ?B = \text{sum}' f ?B$   
 by *(intro sum.eq-sum[symmetric] assms(1))*  
**also have**  $\dots = \text{sum}' f A$   
 by *(intro sum.non-neutral')*  
**also have**  $\dots = 0$  using *assms(2)* by *simp*  
**finally have**  $a:\text{sum } f ?B = 0$  by *simp*  
**have**  $\bigwedge i. i \in ?B \implies f i = 0$   
 using *sum-nonneg-0[OF assms(1) - a]* by *blast*  
**thus**  $\bigwedge i. i \in A \implies f i = 0$   
 by *blast*

**qed**

**lemma** *sum'-eq-iff*:

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

**proof** –

**have**  $\{i \in A. g i \neq 0\} \subseteq \{i \in A. f i \neq 0\}$   
 using *assms(2)* *order-less-le-trans*  
 by *(intro subsetI, auto)*  
**hence**  $a:\text{finite } \{i \in A. g i \neq 0\}$   
 by *(rule finite-subset, intro assms(1))*  
**have**  $\{i \in A. f i - g i \neq 0\} \subseteq \{i \in A. f i \neq 0\}$   
 by *(intro subsetI, simp-all)*

**hence**  $b$ : *finite*  $\{i \in A. f\ i - g\ i \neq 0\}$   
**by** (*rule finite-subset, intro assms(1)*)  
**have**  $\text{sum}' (\lambda i. f\ i - g\ i)\ A = \text{sum}'\ f\ A - \text{sum}'\ g\ A$   
**using** *assms(1,2) a* **by** (*subst sum'-subtractf-nat, auto*)  
**also have**  $\dots = 0$   
**using** *assms(3)* **by** *simp*  
**finally have**  $\text{sum}' (\lambda i. f\ i - g\ i)\ A = 0$  **by** *simp*  
**hence**  $\bigwedge i. i \in A \implies f\ i - g\ i = 0$   
**using** *sum'-nat-eq-0-iff[OF b]* **by** *simp*  
**thus** *?thesis*  
**using** *assms(2) diff-is-0-eq' diffs0-imp-equal* **by** *blast*  
**qed**

## 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** –

**have** *coeff (poly-of-const x) 0 = x* **for**  $x$   
**unfolding** *poly-of-const-def normalize-coeff[symmetric]*  
**by** *simp*  
**thus** *?thesis* **by** (*metis injI*)

**qed**

**lemma** (*in domain*) *embed-hom*:

**assumes** *subring K R*  
**shows** *ring-hom-ring (K[X]) (poly-ring R) id*

**proof** (*rule ring-hom-ringI*)

**show** *ring (K[X])*  
**using** *univ-poly-is-ring[OF assms(1)]* **by** *simp*  
**show** *ring (poly-ring R)*  
**using** *univ-poly-is-ring[OF carrier-is-subring]* **by** *simp*  
**have**  $K \subseteq \text{carrier}\ R$

**using** *subringE(1)[OF assms(1)]* **by** *simp*  
**thus**  $\bigwedge x. x \in \text{carrier}\ (K\ [X]) \implies \text{id}\ x \in \text{carrier}\ (\text{poly-ring}\ R)$   
**unfolding** *univ-poly-carrier[symmetric] polynomial-def* **by** *auto*

**show**  $\text{id}\ (x \otimes_K\ [X]\ y) = \text{id}\ x \otimes_{\text{poly-ring}\ R}\ \text{id}\ y$   
**if**  $x \in \text{carrier}\ (K\ [X])\ y \in \text{carrier}\ (K\ [X])$  **for**  $x\ y$   
**unfolding** *univ-poly-mult* **by** *simp*

**show**  $\text{id}\ (x \oplus_K\ [X]\ y) = \text{id}\ x \oplus_{\text{poly-ring}\ R}\ \text{id}\ y$   
**if**  $x \in \text{carrier}\ (K\ [X])\ y \in \text{carrier}\ (K\ [X])$  **for**  $x\ y$   
**unfolding** *univ-poly-add* **by** *simp*

**show**  $\text{id}\ \mathbf{1}_K\ [X] = \mathbf{1}_{\text{poly-ring}\ R}$   
**unfolding** *univ-poly-one* **by** *simp*

**qed**

The following are versions of the properties of the degrees of poly-

nomials, that abstract over the definition of the polynomial ring structure. In the theories *HOL–Algebra.Polynomial*s 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**  $f \in \text{carrier } (K[X]) - \{\mathbf{0}_{K[X]}\}$   
**assumes**  $g \in \text{carrier } (K[X]) - \{\mathbf{0}_{K[X]}\}$   
**assumes**  $\text{degree } f \neq \text{degree } g$   
**shows**  $\text{degree } (f \oplus_{K[X]} g) = \max (\text{degree } f) (\text{degree } g)$   
**unfolding** *univ-poly-add* **using** *assms(2,3,4)*  
**by** (*subst poly-add-degree-eq[OF assms(1)]*)  
*(auto simp:univ-poly-carrier univ-poly-zero)*

**lemma** (in ring) *degree-add*:  
 $\text{degree } (f \oplus_{K[X]} g) \leq \max (\text{degree } f) (\text{degree } g)$   
**unfolding** *univ-poly-add* **by** (*intro poly-add-degree*)

**lemma** (in domain) *degree-mult*:  
**assumes** *subring*  $K\ R$   
**assumes**  $f \in \text{carrier } (K[X]) - \{\mathbf{0}_{K[X]}\}$   
**assumes**  $g \in \text{carrier } (K[X]) - \{\mathbf{0}_{K[X]}\}$   
**shows**  $\text{degree } (f \otimes_{K[X]} g) = \text{degree } f + \text{degree } g$   
**unfolding** *univ-poly-mult* **using** *assms(2,3)*  
**by** (*subst poly-mult-degree-eq[OF assms(1)]*)  
*(auto simp:univ-poly-carrier univ-poly-zero)*

**lemma** (in ring) *degree-one*:  
 $\text{degree } (\mathbf{1}_{K[X]}) = 0$   
**unfolding** *univ-poly-one* **by** *simp*

**lemma** (in domain) *pow-non-zero*:  
 $x \in \text{carrier } R \implies x \neq \mathbf{0} \implies x [\wedge] (n :: \text{nat}) \neq \mathbf{0}$   
**using** *integral* **by** (*induction n, auto*)

**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** –  
**interpret**  $p:\text{domain } K[X]$   
**using** *univ-poly-is-domain[OF assms(1)]* **by** *simp*

**show** *?thesis*  
**proof** (*induction n*)  
**case**  $0$

```

    then show ?case by (simp add:univ-poly-one)
next
case (Suc n)
have degree (f [^]_K [X] Suc n) = degree (f [^]_K [X] n ⊗_{K[X]} f)
  by simp
also have ... = degree (f [^]_K [X] n) + degree f
  using p.pow-non-zero assms(2)
  by (subst degree-mult[OF assms(1)], auto)
also have ... = degree f * Suc n
  by (subst Suc, simp)
finally show ?case by simp
qed
qed

```

```

lemma (in ring) degree-var:
  degree (X_R) = 1
  unfolding var-def by simp

```

```

lemma (in domain) var-carr:
  fixes n :: nat
  assumes subring K R
  shows X_R ∈ carrier (K[X]) - {0_K [X]}
proof -
  have X_R ∈ carrier (K[X])
    using var-closed[OF assms(1)] by simp
  moreover have X ≠ 0_K [X]
    unfolding var-def univ-poly-zero by simp
  ultimately show ?thesis by simp
qed

```

```

lemma (in domain) var-pow-carr:
  fixes n :: nat
  assumes subring K R
  shows X_R [^]_K [X] n ∈ carrier (K[X]) - {0_K [X]}
proof -
  interpret p:domain K[X]
    using univ-poly-is-domain[OF assms(1)] by simp

  have X_R [^]_K [X] n ∈ carrier (K[X])
    using var-pow-closed[OF assms(1)] by simp
  moreover have X ≠ 0_K [X]
    unfolding var-def univ-poly-zero by simp
  hence X_R [^]_K [X] n ≠ 0_K [X]
    using var-closed(1)[OF assms(1)]
    by (intro p.pow-non-zero, auto)
  ultimately show ?thesis by simp
qed

```

**lemma** (in domain) var-pow-degree:  
**fixes**  $n :: \text{nat}$   
**assumes** subring  $K R$   
**shows**  $\text{degree } (X_R [\overset{\wedge}{\wedge}]_K [X] n) = n$   
**using** var-carr[OF assms(1)] degree-var  
**by** (subst degree-pow[OF assms(1)], auto)

**lemma** (in domain) finprod-non-zero:  
**assumes** finite  $A$   
**assumes**  $f \in A \rightarrow \text{carrier } R - \{0\}$   
**shows**  $(\otimes i \in A. f i) \in \text{carrier } R - \{0\}$   
**using** assms  
**proof** (induction  $A$  rule:finite-induct)  
**case** empty  
**then show** ?case **by** simp  
**next**  
**case** (insert  $x F$ )  
**have**  $\text{finprod } R f (\text{insert } x F) = f x \otimes \text{finprod } R f F$   
**using** insert **by** (subst finprod-insert, simp-all add:Pi-def)  
**also have**  $\dots \in \text{carrier } R - \{0\}$   
**using** integral insert **by** auto  
**finally show** ?case **by** simp  
**qed**

**lemma** (in domain) degree-prod:  
**assumes** finite  $A$   
**assumes** subring  $K R$   
**assumes**  $f \in A \rightarrow \text{carrier } (K[X]) - \{0_{K[X]}\}$   
**shows**  $\text{degree } (\otimes_{K[X]} i \in A. f i) = (\sum i \in A. \text{degree } (f i))$   
**using** assms  
**proof** –  
**interpret**  $p:\text{domain } K[X]$   
**using** univ-poly-is-domain[OF assms(2)] **by** simp

**show** ?thesis  
**using** assms(1,3)  
**proof** (induction  $A$  rule: finite-induct)  
**case** empty  
**then show** ?case **by** (simp add:univ-poly-one)  
**next**  
**case** (insert  $x F$ )  
**have**  $\text{degree } (\text{finprod } (K[X]) f (\text{insert } x F)) =$   
 $\text{degree } (f x \otimes_{K[X]} \text{finprod } (K[X]) f F)$   
**using** insert **by** (subst p.finprod-insert, auto)  
**also have**  $\dots = \text{degree } (f x) + \text{degree } (\text{finprod } (K[X]) f F)$   
**using** insert p.finprod-non-zero[OF insert(1)]  
**by** (subst degree-mult[OF assms(2)], simp-all)  
**also have**  $\dots = \text{degree } (f x) + (\sum i \in F. \text{degree } (f i))$   
**using** insert **by** (subst insert(3), auto)

```

    also have ... = ( $\sum i \in \text{insert } x F. \text{degree } (f i)$ )
      using insert by simp
    finally show ?case by simp
  qed
qed

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 -
  have a:set  $f \subseteq \text{carrier } R$ 
    using assms(1,2) univ-poly-carrier
  using subringE(1)[OF assms(1)] polynomial-incl
  by blast
  have b:set  $g \subseteq \text{carrier } R$ 
    using assms(1,3) univ-poly-carrier
  using subringE(1)[OF assms(1)] polynomial-incl
  by blast
  show ?thesis
    unfolding univ-poly-add poly-add-coeff[OF a b] by simp
qed

```

```

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 -
  have ?L =  $\text{coeff } (\text{map } (a\text{-inv } R) f) i$ 
    unfolding univ-poly-a-inv-def'[OF assms(1,2)] by simp
  also have ... = ?R by (induction f) auto
  finally show ?thesis by simp
qed

```

This is a version of geometric sums for commutative rings:

```

lemma (in cring) geom:
  fixes q:: nat
  assumes [simp]:  $a \in \text{carrier } R$ 
  shows  $(a \ominus \mathbf{1}) \otimes (\bigoplus_{i \in \{..<q\}}. a [\wedge] i) = (a [\wedge] q \ominus \mathbf{1})$ 
    (is ?lhs = ?rhs)
proof -
  have [simp]:  $a [\wedge] i \in \text{carrier } R$  for  $i :: \text{nat}$ 
    by (intro nat-pow-closed assms)
  have [simp]:  $\ominus \mathbf{1} \otimes x = \ominus x$  if  $x \in \text{carrier } R$  for  $x$ 
    using l-minus l-one one-closed that by presburger

  let ?cterm =  $(\bigoplus_{i \in \{1..<q\}}. a [\wedge] i)$ 

```

**have**  $?lhs = a \otimes (\bigoplus_{i \in \{..<q\}}. a [\uparrow] i) \ominus (\bigoplus_{i \in \{..<q\}}. a [\uparrow] i)$   
**unfolding** *a-minus-def* **by** (*subst l-distr, simp-all add:Pi-def*)  
**also have**  $\dots = (\bigoplus_{i \in \{..<q\}}. a \otimes a [\uparrow] i) \ominus (\bigoplus_{i \in \{..<q\}}. a [\uparrow] i)$   
**by** (*subst finsum-rdistr, simp-all add:Pi-def*)  
**also have**  $\dots = (\bigoplus_{i \in \{..<q\}}. a [\uparrow] (Suc\ i)) \ominus (\bigoplus_{i \in \{..<q\}}. a [\uparrow] i)$   
**by** (*subst nat-pow-Suc, simp-all add:m-comm*)  
**also have**  $\dots = (\bigoplus_{i \in Suc\ '\{..<q\}}. a [\uparrow] i) \ominus (\bigoplus_{i \in \{..<q\}}. a [\uparrow] i)$   
**by** (*subst finsum-reindex, simp-all*)  
**also have**  $\dots =$   
 $(\bigoplus_{i \in insert\ q\ \{1..<q\}}. a [\uparrow] i) \ominus$   
 $(\bigoplus_{i \in insert\ 0\ \{1..<q\}}. a [\uparrow] i)$   
**proof** (*cases q > 0*)  
**case** *True*  
**moreover have**  $Suc\ '\{..<q\} = insert\ q\ \{Suc\ 0..<q\}$   
**using** *True lessThan-atLeast0* **by** *fastforce*  
**moreover have**  $\{..<q\} = insert\ 0\ \{Suc\ 0..<q\}$   
**using** *True* **by** (*auto simp add:set-eq-iff*)  
**ultimately show** *?thesis*  
**by** (*intro arg-cong2[where f= $\lambda x\ y. x \ominus y$ ] finsum-cong*)  
*simp-all*  
**next**  
**case** *False*  
**then show** *?thesis* **by** (*simp, algebra*)  
**qed**  
**also have**  $\dots = (a [\uparrow] q \oplus ?cterm) \ominus (\mathbf{1} \oplus ?cterm)$   
**by** *simp*  
**also have**  $\dots = a [\uparrow] q \oplus ?cterm \oplus (\ominus \mathbf{1} \oplus \ominus ?cterm)$   
**unfolding** *a-minus-def* **by** (*subst minus-add, simp-all*)  
**also have**  $\dots = a [\uparrow] q \oplus (?cterm \oplus (\ominus \mathbf{1} \oplus \ominus ?cterm))$   
**by** (*subst a-assoc, simp-all*)  
**also have**  $\dots = a [\uparrow] q \oplus (?cterm \oplus (\ominus ?cterm \oplus \ominus \mathbf{1}))$   
**by** (*subst a-comm[where x= $\ominus \mathbf{1}$ ], simp-all*)  
**also have**  $\dots = a [\uparrow] q \oplus ((?cterm \oplus (\ominus ?cterm)) \oplus \ominus \mathbf{1})$   
**by** (*subst a-assoc, simp-all*)  
**also have**  $\dots = a [\uparrow] q \oplus (\mathbf{0} \oplus \ominus \mathbf{1})$   
**by** (*subst r-neg, simp-all*)  
**also have**  $\dots = a [\uparrow] q \ominus \mathbf{1}$   
**unfolding** *a-minus-def* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** (*in domain*) *rupture-eq-0-iff*:  
**assumes** *subfield K R p  $\in$  carrier (K[X]) q  $\in$  carrier (K[X])*  
**shows** *rupture-surj K p q =  $\mathbf{0}_{Rupt\ K\ p} \iff p\ pdivides\ q$*   
*(is ?lhs  $\iff$  ?rhs)*  
**proof** –  
**interpret** *h:ring-hom-ring K[X] (Rupt K p) (rupture-surj K p)*  
**using** *assms subfieldE* **by** (*intro rupture-surj-hom auto*)

```

have a:  $q \text{ pmod } p \in (\lambda q. q \text{ pmod } p) \text{ ` carrier } (K [X])$ 
  using assms(3) by simp
have  $\mathbf{0}_{K[X]} = \mathbf{0}_{K[X]} \text{ pmod } p$ 
  using assms(1,2) long-division-zero(2)
  by (simp add:univ-poly-zero)
hence b:  $\mathbf{0}_{K[X]} \in (\lambda q. q \text{ pmod } p) \text{ ` carrier } (K[X])$ 
  by (simp add:image-iff) auto

have ?lhs  $\longleftrightarrow$  rupture-surj  $K \text{ pmod } p =$ 
  rupture-surj  $K \text{ pmod } p (\mathbf{0}_{K[X]})$ 
  by (subst rupture-surj-composed-with-pmod[OF assms]) simp
also have ...  $\longleftrightarrow q \text{ pmod } p = \mathbf{0}_{K[X]}$ 
  using assms(3)
  by (intro inj-on-eq-iff[OF rupture-surj-inj-on[OF assms(1,2)]] a b)
also have ...  $\longleftrightarrow$  ?rhs
  unfolding univ-poly-zero
  by (intro pmod-zero-iff-pdivides[OF assms(1)] assms(2,3))
finally show ?thesis by simp
qed

```

## 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**  $h \in \text{ring-iso } R \text{ domain } R \text{ domain } S$

**shows**  $\text{map } h \in \text{ring-iso } (\text{poly-ring } R) (\text{poly-ring } S)$

**proof** (*rule ring-iso-memI*)

**interpret** *dr*: *domain*  $R$  **using** *assms(2)* **by** *blast*

**interpret** *ds*: *domain*  $S$  **using** *assms(3)* **by** *blast*

**interpret** *pdr*: *domain poly-ring*  $R$

**using** *dr.univ-poly-is-domain[OF dr.carrier-is-subring]* **by** *simp*

**interpret** *pds*: *domain poly-ring*  $S$

**using** *ds.univ-poly-is-domain[OF ds.carrier-is-subring]* **by** *simp*

**interpret** *h*: *ring-hom-ring*  $R \text{ domain } S \text{ h}$

**using** *dr.ring-axioms ds.ring-axioms assms(1)*

**by** (*intro ring-hom-ringI2, simp-all add:ring-iso-def*)

**let** *?R* = *poly-ring*  $R$

**let** *?S* = *poly-ring*  $S$

**have** *h-img*:  $h \text{ ` (carrier } R) = \text{carrier } S$

**using** *assms(1)* **unfolding** *ring-iso-def bij-betw-def* **by** *auto*

**have** *h-inj*: *inj-on*  $h \text{ (carrier } R)$

**using** *assms(1)* **unfolding** *ring-iso-def bij-betw-def* **by** *auto*

**hence** *h-non-zero-iff*:  $h \ x \neq \mathbf{0}_S$

**if**  $x \neq \mathbf{0}_R$   $x \in \text{carrier } R$  **for**  $x$

**using** *h.hom-zero dr.zero-closed inj-onD that* **by** *metis*

```

have norm-elim: ds.normalize (map h x) = map h x
  if x ∈ carrier (poly-ring R) for x
proof (cases x)
  case Nil then show ?thesis by simp
next
  case (Cons xh xt)
  have xh ∈ carrier R xh ≠ 0R
    using that unfolding Cons univ-poly-carrier[symmetric]
    unfolding polynomial-def by auto
  hence h xh ≠ 0S using h-non-zero-iff by simp
  then show ?thesis unfolding Cons by simp
qed

show t-1: map h x ∈ carrier ?S
  if x ∈ carrier ?R for x
  using that hd-in-set h-non-zero-iff hd-map
  unfolding univ-poly-carrier[symmetric] polynomial-def
  by (cases x, auto)

show map h (x ⊗?R y) = map h x ⊗?S map h y
  if x ∈ carrier ?R y ∈ carrier ?R for x y
proof –
  have map h (x ⊗?R y) = ds.normalize (map h (x ⊗?R y))
    using that by (intro norm-elim[symmetric],simp)
  also have ... = map h x ⊗?S map h y
    using that unfolding univ-poly-mult univ-poly-carrier[symmetric]
    unfolding polynomial-def
    by (intro h.poly-mult-hom'[of x y], auto)
  finally show ?thesis by simp
qed

show map h (x ⊕?R y) = map h x ⊕?S map h y
  if x ∈ carrier ?R y ∈ carrier ?R for x y
proof –
  have map h (x ⊕?R y) = ds.normalize (map h (x ⊕?R y))
    using that by (intro norm-elim[symmetric],simp)
  also have ... = map h x ⊕?S map h y
    using that
    unfolding univ-poly-add univ-poly-carrier[symmetric]
    unfolding polynomial-def
    by (intro h.poly-add-hom'[of x y], auto)
  finally show ?thesis by simp
qed

show map h 1?R = 1?S
  unfolding univ-poly-one by simp

let ?hinv = map (the-inv-into (carrier R) h)

```

```

have map h ∈ carrier ?R → carrier ?S
  using t-1 by simp
moreover have ?hinv x ∈ carrier ?R
  if x ∈ carrier ?S for x
proof (cases x = [])
  case True
  then show ?thesis
    by (simp add:univ-poly-carrier[symmetric] polynomial-def)
next
  case False
  have set-x: set x ⊆ h ` carrier R
    using that h-img unfolding univ-poly-carrier[symmetric]
    unfolding polynomial-def by auto
  have lead-coeff x ≠ 0_S lead-coeff x ∈ carrier S
    using that False unfolding univ-poly-carrier[symmetric]
    unfolding polynomial-def by auto
  hence the-inv-into (carrier R) h (lead-coeff x) ≠
    the-inv-into (carrier R) h 0_S
    using inj-on-the-inv-into[OF h-inj] inj-onD
    using ds.zero-closed h-img by metis
  hence the-inv-into (carrier R) h (lead-coeff x) ≠ 0_R
    unfolding h.hom-zero[symmetric]
    unfolding the-inv-into-f-f[OF h-inj dr.zero-closed] by simp
  hence lead-coeff (?hinv x) ≠ 0_R
    using False by (simp add:hd-map)
  moreover have the-inv-into (carrier R) h ` set x ⊆ carrier R
    using the-inv-into-into[OF h-inj] set-x
    by (intro image-subsetI) auto
  hence set (?hinv x) ⊆ carrier R by simp
  ultimately show ?thesis
    by (simp add:univ-poly-carrier[symmetric] polynomial-def)
qed
moreover have ?hinv (map h x) = x if x ∈ carrier ?R for x
proof –
  have set-x: set x ⊆ carrier R
    using that unfolding univ-poly-carrier[symmetric]
    unfolding polynomial-def by auto
  have ?hinv (map h x) =
    map (λy. the-inv-into (carrier R) h (h y)) x
    by simp
  also have ... = map id x
    using set-x by (intro map-cong)
    (auto simp add:the-inv-into-f-f[OF h-inj])
  also have ... = x by simp
  finally show ?thesis by simp
qed
moreover have map h (?hinv x) = x
  if x ∈ carrier ?S for x

```

```

proof –
  have set-x: set  $x \subseteq h \text{ ` carrier } R$ 
    using that h-imp unfolding univ-poly-carrier[symmetric]
    unfolding polynomial-def by auto
  have map h (?hinv  $x$ ) =
    map ( $\lambda y. h (the\text{-inv}\text{-into} (carrier\ R) h\ y))\ x$ 
    by simp
  also have  $\dots = map\ id\ x$ 
    using set-x by (intro map-cong)
    (auto simp add:f-the-inv-into-f[OF h-inj])
  also have  $\dots = x$  by simp
  finally show ?thesis by simp
qed
ultimately show bij-betw (map h) (carrier ?R) (carrier ?S)
  by (intro bij-betwI[where g=?hinv], auto)
qed

```

```

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 –
  note poly-iso = lift-iso-to-poly-ring[OF assms(2,3,4)]
  show ?thesis
    using ring-iso-memE(1)[OF poly-iso assms(1)] by simp
qed

```

```

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-on h (carrier R)
  shows  $map\ h\ f \in carrier (poly\text{-ring } S)$ 
proof –
  let  $?S = S \langle carrier := h \text{ ` carrier } R \rangle$ 

  interpret dr: domain R using assms(3) by blast
  interpret ds: domain S using assms(4) by blast
  interpret h1: ring-hom-ring  $R\ S\ h$ 
    using assms(2) ring-hom-ringI2 dr.ring-axioms
    using ds.ring-axioms by blast
  have subr: subring ( $h \text{ ` carrier } R$ )  $S$ 
    using h1.img-is-subring[OF dr.carrier-is-subring] by blast
  interpret h: ring-hom-ring ( $(h \text{ ` carrier } R)[X]_S$ ) poly-ring S id
    using ds.embed-hom[OF subr] by simp

  let  $?S = S \langle carrier := h \text{ ` carrier } R \rangle$ 
  have  $h \in ring\text{-hom } R\ ?S$ 
    using assms(2) unfolding ring-hom-def by simp

```

```

moreover have bij-betw  $h$  (carrier  $R$ ) (carrier  $?S$ )
  using assms(5) bij-betw-def by auto
ultimately have h-iso:  $h \in \text{ring-iso } R \ ?S$ 
  unfolding ring-iso-def by simp

have dom-S: domain  $?S$ 
  using ds.subring-is-domain[OF subr] by simp

note poly-iso = lift-iso-to-poly-ring[OF h-iso assms(3) dom-S]
have map  $h$   $f \in \text{carrier } (\text{poly-ring } ?S)$ 
  using ring-iso-memE(1)[OF poly-iso assms(1)] by simp
also have carrier (poly-ring  $?S$ ) =
  carrier (univ-poly  $S$  ( $h$  ‘ carrier  $R$ ))
  using ds.univ-poly-consistent[OF subr] by simp
also have  $\dots \subseteq \text{carrier } (\text{poly-ring } S)$ 
  using h.hom-closed by auto
finally show ?thesis by simp
qed

```

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

```

lemma divides-hom:
  assumes  $h \in \text{ring-iso } R \ S$ 
  assumes domain  $R$  domain  $S$ 
  assumes  $x \in \text{carrier } R$   $y \in \text{carrier } R$ 
  shows  $x \text{ divides}_R y \iff (h \ x) \text{ divides}_S (h \ y)$  (is ?lhs  $\iff$  ?rhs)
proof –
  interpret dr: domain  $R$  using assms(2) by blast
  interpret ds: domain  $S$  using assms(3) by blast
  interpret pdr: domain poly-ring  $R$ 
    using dr.univ-poly-is-domain[OF dr.carrier-is-subring] by simp
  interpret pds: domain poly-ring  $S$ 
    using ds.univ-poly-is-domain[OF ds.carrier-is-subring] by simp
  interpret h: ring-hom-ring  $R \ S \ h$ 
    using dr.ring-axioms ds.ring-axioms assms(1)
    by (intro ring-hom-ringI2, simp-all add:ring-iso-def)

have h-inj-on: inj-on  $h$  (carrier  $R$ )
  using assms(1) unfolding ring-iso-def bij-betw-def by auto
have h-img:  $h$  ‘ (carrier  $R$ ) = carrier  $S$ 
  using assms(1) unfolding ring-iso-def bij-betw-def by auto

have ?lhs  $\iff (\exists c \in \text{carrier } R. y = x \otimes_R c)$ 
  unfolding factor-def by simp
also have  $\dots \iff (\exists c \in \text{carrier } R. h \ y = h \ x \otimes_S h \ c)$ 
  using assms(4,5) inj-onD[OF h-inj-on]
  by (intro beq-cong, auto simp flip:h.hom-mult)
also have  $\dots \iff (\exists c \in \text{carrier } S. h \ y = h \ x \otimes_S c)$ 
  unfolding h-img[symmetric] by simp

```

also have ...  $\longleftrightarrow$  ?rhs  
 unfolding factor-def by simp  
 finally show ?thesis by simp  
 qed

lemma properfactor-hom:  
 assumes  $h \in \text{ring-iso } R S$   
 assumes domain  $R$  domain  $S$   
 assumes  $x \in \text{carrier } R$   $b \in \text{carrier } R$   
 shows properfactor  $R$   $b$   $x \longleftrightarrow$  properfactor  $S$   $(h b)$   $(h x)$   
 using divides-hom[OF assms(1,2,3)] assms(4,5)  
 unfolding properfactor-def by simp

lemma Units-hom:  
 assumes  $h \in \text{ring-iso } R S$   
 assumes domain  $R$  domain  $S$   
 assumes  $x \in \text{carrier } R$   
 shows  $x \in \text{Units } R \longleftrightarrow h x \in \text{Units } S$   
 proof –

interpret  $dr$ : domain  $R$  using assms(2) by blast  
 interpret  $ds$ : domain  $S$  using assms(3) by blast  
 interpret  $pdr$ : domain poly-ring  $R$   
 using  $dr.\text{univ-poly-is-domain}$ [OF  $dr.\text{carrier-is-subring}$ ] by simp  
 interpret  $pds$ : domain poly-ring  $S$   
 using  $ds.\text{univ-poly-is-domain}$ [OF  $ds.\text{carrier-is-subring}$ ] by simp  
 interpret  $h$ : ring-hom-ring  $R S$   $h$   
 using  $dr.\text{ring-axioms}$   $ds.\text{ring-axioms}$  assms(1)  
 by (intro ring-hom-ringI2, simp-all add:ring-iso-def)

have  $h\text{-img}$ :  $h \text{ ` } (\text{carrier } R) = \text{carrier } S$   
 using assms(1) unfolding ring-iso-def bij-betw-def by auto

have  $h\text{-inj-on}$ :  $\text{inj-on } h (\text{carrier } R)$   
 using assms(1) unfolding ring-iso-def bij-betw-def by auto

hence  $h\text{-one-iff}$ :  $h x = \mathbf{1}_S \longleftrightarrow x = \mathbf{1}_R$  if  $x \in \text{carrier } R$  for  $x$   
 using  $h.\text{hom-one}$  that by (metis  $dr.\text{one-closed}$   $\text{inj-onD}$ )

have  $x \in \text{Units } R \longleftrightarrow$   
 ( $\exists y \in \text{carrier } R. x \otimes_R y = \mathbf{1}_R \wedge y \otimes_R x = \mathbf{1}_R$ )  
 using assms unfolding Units-def by auto  
 also have ...  $\longleftrightarrow$   
 ( $\exists y \in \text{carrier } R. h x \otimes_S h y = h \mathbf{1}_R \wedge h y \otimes_S h x = h \mathbf{1}_R$ )  
 using  $h\text{-one-iff}$  assms by (intro  $bex\text{-cong}$ , simp-all flip: $h.\text{hom-mult}$ )  
 also have ...  $\longleftrightarrow$   
 ( $\exists y \in \text{carrier } S. h x \otimes_S y = h \mathbf{1}_R \wedge y \otimes_S h x = \mathbf{1}_S$ )  
 unfolding  $h\text{-img}$ [symmetric] by simp  
 also have ...  $\longleftrightarrow h x \in \text{Units } S$

using *assms h.hom-closed* **unfolding** *Units-def* **by auto**  
**finally show** *?thesis* **by simp**  
**qed**

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

**proof** –

**have** *h-img*:  $h \ ` \ (\text{carrier } R) = \text{carrier } S$   
**using** *assms(1)* **unfolding** *ring-iso-def bij-betw-def* **by auto**

**have**  $\text{irreducible } R \ x \longleftrightarrow (x \notin \text{Units } R \wedge$   
 $(\forall b \in \text{carrier } R. \text{properfactor } R \ b \ x \longrightarrow b \in \text{Units } R))$   
**unfolding** *Divisibility.irreducible-def* **by simp**

**also have**  $\dots \longleftrightarrow (x \notin \text{Units } R \wedge$   
 $(\forall b \in \text{carrier } R. \text{properfactor } S \ (h \ b) \ (h \ x) \longrightarrow b \in \text{Units } R))$   
**using** *properfactor-hom[OF assms(1,2,3)] assms(4)* **by simp**

**also have**  $\dots \longleftrightarrow (h \ x \notin \text{Units } S \wedge$   
 $(\forall b \in \text{carrier } R. \text{properfactor } S \ (h \ b) \ (h \ x) \longrightarrow h \ b \in \text{Units } S))$   
**using** *assms(4) Units-hom[OF assms(1,2,3)]* **by simp**

**also have**  $\dots \longleftrightarrow (h \ x \notin \text{Units } S \wedge$   
 $(\forall b \in h \ ` \ \text{carrier } R. \text{properfactor } S \ b \ (h \ x) \longrightarrow b \in \text{Units } S))$   
**by simp**

**also have**  $\dots \longleftrightarrow \text{irreducible } S \ (h \ x)$   
**unfolding** *h-img Divisibility.irreducible-def* **by simp**  
**finally show** *?thesis* **by simp**

**qed**

**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)**

**proof** –

**note** *lift-iso* = *lift-iso-to-poly-ring[OF assms(1,2,3)]*  
**interpret** *dr*:  $\text{domain } R$  **using** *assms(2)* **by blast**  
**interpret** *ds*:  $\text{domain } S$  **using** *assms(3)* **by blast**  
**interpret** *pdr*:  $\text{domain poly-ring } R$   
**using** *dr.univ-poly-is-domain[OF dr.carrier-is-subring]* **by simp**  
**interpret** *pds*:  $\text{domain poly-ring } S$   
**using** *ds.univ-poly-is-domain[OF ds.carrier-is-subring]* **by simp**

**have** *mh-inj-on*:  $\text{inj-on } (\text{map } h) \ (\text{carrier } (\text{poly-ring } R))$   
**using** *lift-iso* **unfolding** *ring-iso-def bij-betw-def* **by auto**  
**moreover have**  $\text{map } h \ \mathbf{0}_{\text{poly-ring } R} = \mathbf{0}_{\text{poly-ring } S}$

by (*simp add:univ-poly-zero*)  
**ultimately have** *mh-zero-iff*:  
 $map\ h\ f = \mathbf{0}_{poly\text{-}ring\ S} \longleftrightarrow f = \mathbf{0}_{poly\text{-}ring\ R}$   
 using *assms(4)* by (*metis pdr.zero-closed inj-onD*)

**have** *?lhs*  $\longleftrightarrow (f \neq \mathbf{0}_{poly\text{-}ring\ R} \wedge irreducible\ (poly\text{-}ring\ R)\ f)$   
**unfolding** *ring-irreducible-def* by *simp*  
**also have** ...  $\longleftrightarrow$   
 $(f \neq \mathbf{0}_{poly\text{-}ring\ R} \wedge irreducible\ (poly\text{-}ring\ S)\ (map\ h\ f))$   
 using *irreducible-hom[OF lift-iso] pdr.domain-axioms*  
 using *assms(4) pds.domain-axioms* by *simp*  
**also have** ...  $\longleftrightarrow$   
 $(map\ h\ f \neq \mathbf{0}_{poly\text{-}ring\ S} \wedge irreducible\ (poly\text{-}ring\ S)\ (map\ h\ f))$   
 using *mh-zero-iff* by *simp*  
**also have** ...  $\longleftrightarrow$  *?rhs*  
**unfolding** *ring-irreducible-def* by *simp*  
**finally show** *?thesis* by *simp*  
**qed**

**lemma** *ring-hom-cong*:  
**assumes**  $\bigwedge x. x \in carrier\ R \implies f'\ x = f\ x$   
**assumes** *ring R*  
**assumes**  $f \in ring\text{-}hom\ R\ S$   
**shows**  $f' \in ring\text{-}hom\ R\ S$   
**proof** –  
**interpret** *ring R* using *assms(2)* by *simp*  
**show** *?thesis*  
 using *assms(1) ring-hom-memE[OF assms(3)]*  
 by (*intro ring-hom-memI, auto*)  
**qed**

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**  $(\lambda x. (J <+>_R x)) \in ring\text{-}hom\ (R\ Quot\ I)\ (R\ Quot\ J)$   
**proof** (*rule ring-hom-memI*)  
**interpret** *ji: ideal J R*  
 using *assms(2)* by *simp*  
**interpret** *ii: ideal I R*  
 using *assms(1)* by *simp*

**have**  $a: J <+>_R I = J$   
 using *assms(3)* **unfolding** *set-add-def set-mult-def* by *auto*

**show**  $J <+>_R x \in carrier\ (R\ Quot\ J)$   
**if**  $x \in carrier\ (R\ Quot\ I)$  **for**  $x$

**proof** –  
**have**  $\exists y \in \text{carrier } R. x = I +> y$   
**using** *that unfolding FactRing-def A-RCOSETS-def'* **by** *simp*  
**then obtain**  $y$  **where**  $y\text{-def}: y \in \text{carrier } R \ x = I +> y$   
**by** *auto*  
**have**  $J \langle + \rangle_R (I +> y) = (J \langle + \rangle_R I) +> y$   
**using**  $y\text{-def}(1)$  **by** *(subst a-setmult-rcos-assoc) auto*  
**also have**  $\dots = J +> y$  **using**  $a$  **by** *simp*  
**finally have**  $J \langle + \rangle_R (I +> y) = J +> y$  **by** *simp*  
**thus** *?thesis*  
**using**  $y\text{-def}$  **unfolding** *FactRing-def A-RCOSETS-def'* **by** *auto*  
**qed**

**show**  $J \langle + \rangle_R x \otimes_R \text{Quot } I \ y =$   
 $(J \langle + \rangle_R x) \otimes_R \text{Quot } J (J \langle + \rangle_R y)$   
**if**  $x \in \text{carrier } (R \ \text{Quot } I) \ y \in \text{carrier } (R \ \text{Quot } I)$   
**for**  $x \ y$

**proof** –  
**have**  $\exists x1 \in \text{carrier } R. x = I +> x1 \ \exists y1 \in \text{carrier } R. y = I +> y1$   
**using** *that unfolding FactRing-def A-RCOSETS-def'* **by** *auto*  
**then obtain**  $x1 \ y1$   
**where**  $x1\text{-def}: x1 \in \text{carrier } R \ x = I +> x1$   
**and**  $y1\text{-def}: y1 \in \text{carrier } R \ y = I +> y1$   
**by** *auto*  
**have**  $J \langle + \rangle_R x \otimes_R \text{Quot } I \ y = J \langle + \rangle_R (I +> x1 \otimes y1)$   
**using**  $x1\text{-def} \ y1\text{-def}$   
**by** *(simp add: FactRing-def ii.rcoset-mult-add)*  
**also have**  $\dots = (J \langle + \rangle_R I) +> x1 \otimes y1$   
**using**  $x1\text{-def}(1) \ y1\text{-def}(1)$   
**by** *(subst a-setmult-rcos-assoc) auto*  
**also have**  $\dots = J +> x1 \otimes y1$   
**using**  $a$  **by** *simp*  
**also have**  $\dots = [\text{mod } J:] (J +> x1) \otimes (J +> y1)$   
**using**  $x1\text{-def}(1) \ y1\text{-def}(1)$  **by** *(subst ji.rcoset-mult-add, auto)*  
**also have**  $\dots =$   
 $[\text{mod } J:] ((J \langle + \rangle_R I) +> x1) \otimes ((J \langle + \rangle_R I) +> y1)$   
**using**  $a$  **by** *simp*  
**also have**  $\dots =$   
 $[\text{mod } J:] (J \langle + \rangle_R (I +> x1)) \otimes (J \langle + \rangle_R (I +> y1))$   
**using**  $x1\text{-def}(1) \ y1\text{-def}(1)$   
**by** *(subst (1 2) a-setmult-rcos-assoc) auto*  
**also have**  $\dots = (J \langle + \rangle_R x) \otimes_R \text{Quot } J (J \langle + \rangle_R y)$   
**using**  $x1\text{-def} \ y1\text{-def}$  **by** *(simp add: FactRing-def)*  
**finally show** *?thesis* **by** *simp*  
**qed**

**show**  $J \langle + \rangle_R x \oplus_R \text{Quot } I \ y =$   
 $(J \langle + \rangle_R x) \oplus_R \text{Quot } J (J \langle + \rangle_R y)$   
**if**  $x \in \text{carrier } (R \ \text{Quot } I) \ y \in \text{carrier } (R \ \text{Quot } I)$

**for**  $x\ y$   
**proof** –  
**have**  $\exists x1 \in \text{carrier } R. x = I +> x1 \ \exists y1 \in \text{carrier } R. y = I +> y1$   
**using** *that unfolding FactRing-def A-RCOSETS-def'* **by** *auto*  
**then obtain**  $x1\ y1$   
**where**  $x1\text{-def}: x1 \in \text{carrier } R\ x = I +> x1$   
**and**  $y1\text{-def}: y1 \in \text{carrier } R\ y = I +> y1$   
**by** *auto*  
**have**  $J \langle + \rangle_R x \oplus_R \text{Quot } I\ y =$   
 $J \langle + \rangle_R ((I +> x1) \langle + \rangle_R (I +> y1))$   
**using**  $x1\text{-def}\ y1\text{-def}$  **by** *(simp add:FactRing-def)*  
**also have**  $\dots = J \langle + \rangle_R (I +> (x1 \oplus y1))$   
**using**  $x1\text{-def}\ y1\text{-def}$  *ii.a-rcos-sum* **by** *simp*  
**also have**  $\dots = (J \langle + \rangle_R I) +> (x1 \oplus y1)$   
**using**  $x1\text{-def}\ y1\text{-def}$  **by** *(subst a-setmult-rcos-assoc) auto*  
**also have**  $\dots = J +> (x1 \oplus y1)$   
**using**  $a$  **by** *simp*  
**also have**  $\dots =$   
 $((J \langle + \rangle_R I) +> x1) \langle + \rangle_R ((J \langle + \rangle_R I) +> y1)$   
**using**  $x1\text{-def}\ y1\text{-def}$  *ji.a-rcos-sum a* **by** *simp*  
**also have**  $\dots =$   
 $J \langle + \rangle_R (I +> x1) \langle + \rangle_R (J \langle + \rangle_R (I +> y1))$   
**using**  $x1\text{-def}\ y1\text{-def}$  **by** *(subst (1 2) a-setmult-rcos-assoc) auto*  
**also have**  $\dots = (J \langle + \rangle_R x) \oplus_R \text{Quot } J (J \langle + \rangle_R y)$   
**using**  $x1\text{-def}\ y1\text{-def}$  **by** *(simp add:FactRing-def)*  
**finally show** *?thesis* **by** *simp*  
**qed**

**have**  $J \langle + \rangle_R \mathbf{1}_R \text{Quot } I = J \langle + \rangle_R (I +> \mathbf{1})$   
**unfolding** *FactRing-def* **by** *simp*  
**also have**  $\dots = (J \langle + \rangle_R I) +> \mathbf{1}$   
**by** *(subst a-setmult-rcos-assoc) auto*  
**also have**  $\dots = J +> \mathbf{1}$  **using**  $a$  **by** *simp*  
**also have**  $\dots = \mathbf{1}_R \text{Quot } J$   
**unfolding** *FactRing-def* **by** *simp*  
**finally show**  $J \langle + \rangle_R \mathbf{1}_R \text{Quot } I = \mathbf{1}_R \text{Quot } J$   
**by** *simp*  
**qed**

**lemma** *(in ring) quot-carr:*

**assumes** *ideal I R*  
**assumes**  $y \in \text{carrier } (R \text{Quot } I)$   
**shows**  $y \subseteq \text{carrier } R$

**proof** –

**interpret** *ideal I R* **using** *assms(1)* **by** *simp*  
**have**  $y \in \text{a-rcosets } I$   
**using** *assms(2)* **unfolding** *FactRing-def* **by** *simp*  
**then obtain**  $v$  **where**  $y\text{-def}: y = I +> v\ v \in \text{carrier } R$   
**unfolding** *A-RCOSETS-def'* **by** *auto*

```

have  $I \rightarrow v \subseteq \text{carrier } R$ 
  using  $y\text{-def}(2)$   $a\text{-r-coset-subset-}G$   $a\text{-subset}$  by  $\text{presburger}$ 
thus  $y \subseteq \text{carrier } R$  unfolding  $y\text{-def}$  by  $\text{simp}$ 
qed

```

```

lemma (in  $\text{ring}$ )  $\text{set-add-zero}$ :
  assumes  $A \subseteq \text{carrier } R$ 
  shows  $\{0\} \langle + \rangle_R A = A$ 
proof –
  have  $\{0\} \langle + \rangle_R A = (\bigcup x \in A. \{0 \oplus x\})$ 
    using  $\text{assms}$  unfolding  $\text{set-add-def}$   $\text{set-mult-def}$  by  $\text{simp}$ 
  also have  $\dots = (\bigcup x \in A. \{x\})$ 
    using  $\text{assms}$  by ( $\text{intro arg-cong}$ [where  $f = \text{Union}$ ]  $\text{image-cong}$ ,  $\text{auto}$ )
  also have  $\dots = A$  by  $\text{simp}$ 
  finally show  $?thesis$  by  $\text{simp}$ 
qed

```

Adapted from the proof of  $\text{domain.polynomial-rupture}$

```

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

```

```

proof –
  let  $?surj = \text{rupture-surj } K p$ 

```

```

interpret  $UP$ :  $\text{domain } K[X]$ 
  using  $\text{univ-poly-is-domain}[OF \text{ assms}(1)]$  .
interpret  $h$ :  $\text{ring-hom-ring } K[X] Rupt K p ?surj$ 
  using  $\text{rupture-surj-hom}(2)[OF \text{ assms}(1,2)]$  .

```

```

have  $(h.S.eval) (\text{map } (?surj \circ \text{poly-of-const}) q) (?surj X) =$ 
   $?surj ((UP.eval) (\text{map } \text{poly-of-const } q) X)$ 
using  $h.eval\text{-hom}[OF UP.\text{carrier-is-subring } \text{var-closed}(1)[OF \text{ assms}(1)]$ 
   $\text{map-norm-in-poly-ring-carrier}[OF \text{ assms}(1,3)]$  by  $\text{simp}$ 
also have  $\dots = ?surj q$ 
  unfolding  $\text{sym}[OF \text{ eval-rewrite}[OF \text{ assms}(1,3)]]$  ..
finally show  $?thesis$  by  $\text{simp}$ 

```

**qed**

## 2.4 Divisibility

```

lemma (in  $\text{field}$ )  $f\text{-comm-group-1}$ :
  assumes  $x \in \text{carrier } R$   $y \in \text{carrier } R$ 
  assumes  $x \neq 0$   $y \neq 0$ 
  assumes  $x \otimes y = 0$ 
  shows  $\text{False}$ 
  using  $\text{integral assms}$  by  $\text{auto}$ 

```

lemma (in field) f-comm-group-2:  
 assumes  $x \in \text{carrier } R$   
 assumes  $x \neq \mathbf{0}$   
 shows  $\exists y \in \text{carrier } R - \{\mathbf{0}\}. y \otimes x = \mathbf{1}$   
 proof –  
 have  $x\text{-unit}: x \in \text{Units } R$  using field-Units assms by simp  
 thus ?thesis unfolding Units-def by auto  
 qed

sublocale field < mult-of: comm-group mult-of R  
 rewrites mult (mult-of R) = mult R  
 and one (mult-of R) = one R  
 using f-comm-group-1 f-comm-group-2  
 by (auto intro!: comm-groupI m-assoc m-comm)

lemma (in domain) div-neg:  
 assumes  $a \in \text{carrier } R$   $b \in \text{carrier } R$   
 assumes  $a$  divides  $b$   
 shows  $a$  divides  $(\ominus b)$   
 proof –  
 obtain  $r1$  where  $r1\text{-def}: r1 \in \text{carrier } R$   $a \otimes r1 = b$   
 using assms by (auto simp: factor-def)  
  
 have  $a \otimes (\ominus r1) = \ominus (a \otimes r1)$   
 using assms(1)  $r1\text{-def}(1)$  by algebra  
 also have  $\dots = \ominus b$   
 using  $r1\text{-def}(2)$  by simp  
 finally have  $\ominus b = a \otimes (\ominus r1)$  by simp  
 moreover have  $\ominus r1 \in \text{carrier } R$   
 using  $r1\text{-def}(1)$  by simp  
 ultimately show ?thesis  
 by (auto simp: factor-def)  
 qed

lemma (in domain) div-sum:  
 assumes  $a \in \text{carrier } R$   $b \in \text{carrier } R$   $c \in \text{carrier } R$   
 assumes  $a$  divides  $b$   
 assumes  $a$  divides  $c$   
 shows  $a$  divides  $(b \oplus c)$   
 proof –  
 obtain  $r1$  where  $r1\text{-def}: r1 \in \text{carrier } R$   $a \otimes r1 = b$   
 using assms by (auto simp: factor-def)  
  
 obtain  $r2$  where  $r2\text{-def}: r2 \in \text{carrier } R$   $a \otimes r2 = c$   
 using assms by (auto simp: factor-def)  
  
 have  $a \otimes (r1 \oplus r2) = (a \otimes r1) \oplus (a \otimes r2)$   
 using assms(1)  $r1\text{-def}(1)$   $r2\text{-def}(1)$  by algebra

also have  $\dots = b \oplus c$   
 using *r1-def(2) r2-def(2)* by *simp*  
 finally have  $b \oplus c = a \otimes (r1 \oplus r2)$  by *simp*  
 moreover have  $r1 \oplus r2 \in \text{carrier } R$   
 using *r1-def(1) r2-def(1)* by *simp*  
 ultimately show *?thesis*  
 by (*auto simp:factor-def*)  
 qed

**lemma** (*in domain*) *div-sum-iff*:  
 assumes  $a \in \text{carrier } R$   $b \in \text{carrier } R$   $c \in \text{carrier } R$   
 assumes *a divides b*  
 shows *a divides (b ⊕ c) ⟷ a divides c*  
**proof**  
 assume *a divides (b ⊕ c)*  
 moreover have *a divides (⊖ b)*  
 using *div-neg assms(1,2,4)* by *simp*  
 ultimately have *a divides ((b ⊕ c) ⊕ (⊖ b))*  
 using *div-sum assms* by *simp*  
 also have  $\dots = c$  using *assms(1,2,3)* by *algebra*  
 finally show *a divides c* by *simp*  
**next**  
 assume *a divides c*  
 thus *a divides (b ⊕ c)*  
 using *assms* by (*intro div-sum*) *auto*  
 qed

**lemma** (*in comm-monoid*) *irreducible-prod-unit*:  
 assumes  $f \in \text{carrier } G$   $x \in \text{Units } G$   
 shows *irreducible G f = irreducible G (x ⊗ f)* (*is ?L = ?R*)  
**proof**  
 assume *?L*  
 thus *?R* using *irreducible-prod-II assms* by *auto*  
**next**  
 have  $\text{inv } x \otimes (x \otimes f) = (\text{inv } x \otimes x) \otimes f$   
 using *assms* by (*intro m-assoc[symmetric]*) *auto*  
 also have  $\dots = f$  using *assms* by *simp*  
 finally have *0: inv x ⊗ (x ⊗ f) = f* by *simp*  
 assume *?R*  
 hence *irreducible G (inv x ⊗ (x ⊗ f))* using *irreducible-prod-II*  
*assms* by *blast*  
 thus *?L* using *0* by *simp*  
 qed

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*  $f. (\exists as. f = \text{fmset } G as \wedge \text{wfactors } G as x \wedge \text{set } as \subseteq \text{carrier } G)$ )

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

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

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**  $a \sim b \longleftrightarrow (\text{fmset } G as = \text{fmset } G bs)$

**proof** –

**have**  $a \sim b \longleftrightarrow (a \text{ divides } b \wedge b \text{ divides } a)$   
**by** (*simp add: associated-def*)  
**also have**  $\dots \longleftrightarrow$   
 $(\text{fmset } G as \subseteq\# \text{fmset } G bs \wedge \text{fmset } G bs \subseteq\# \text{fmset } G as)$   
**using** *divides-as-fmsubset assms* **by** *blast*  
**also have**  $\dots \longleftrightarrow (\text{fmset } G as = \text{fmset } G bs)$  **by** *auto*

**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *factor-mset-aux-1*:

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

**proof** –

**define**  $H$  **where**  $H = \{as. \text{wfactors } G as a \wedge \text{set } as \subseteq \text{carrier } G\}$   
**have**  $b: as \in H$   
**using**  $H\text{-def}$  **assms** **by** *simp*

**have**  $c: x \in H \implies y \in H \implies \text{fmset } G x = \text{fmset } G y$  **for**  $x y$   
**unfolding**  $H\text{-def}$  **using**  $\text{assoc-as-fmset-eq}$   
**using**  $\text{associated-refl}$  **assms** **by** *blast*

**have**  $\text{factor-mset } G a = (\text{THE } f. \exists as \in H. f = \text{fmset } G as)$   
**by** (*simp add: factor-mset-def H-def, metis*)

**also have**  $\dots = \text{fmset } G as$   
**using**  $b c$   
**by** (*intro the1-equality*) **blast+**  
**finally have**  $\text{factor-mset } G a = \text{fmset } G as$  **by** *simp*

**thus** *?thesis*  
**using**  $b$  **unfolding**  $H\text{-def}$  **by** *auto*  
**qed**

**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** –

**obtain**  $as$  **where**  $as\text{-def}: \text{wfactors } G as a$  **set**  $as \subseteq \text{carrier } G$   
**using**  $\text{wfactors-exist}$  **assms** **by** *blast*  
**thus** *?thesis* **using**  $\text{factor-mset-aux-1}$  **assms** **by** *blast*

**qed**

**lemma** *factor-mset-set*:

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

**proof** –

**obtain**  $as$  **where**  $as\text{-def}: \text{factor-mset } G a = \text{fmset } G as$   
 $\text{wfactors } G as a$  **set**  $as \subseteq \text{carrier } G$   
**using**  $\text{factor-mset-aux}$  **assms** **by** *blast*

**hence**  $x \in \# \text{ fmset } G \text{ as}$   
**using** *assms* **by** *simp*  
**hence**  $x \in \text{assocs } G \text{ ' set as}$   
**using** *assms as-def* **by** (*simp add:fmset-def*)  
**hence**  $\exists y. y \in \text{set as} \wedge x = \text{assocs } G y$   
**by** *auto*  
**moreover have**  $y \in \text{carrier } G \wedge \text{irreducible } G y$   
**if**  $y \in \text{set as}$  **for**  $y$   
**using** *as-def that wfactors-def*  
**by** (*simp add: wfactors-def*) *auto*  
**ultimately show** *?thesis*  
**using** *that* **by** *blast*  
**qed**

**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** –  
**obtain** *as* **where** *as-def*:  
 $\text{factor-mset } G a = \text{fmset } G a$   
 $\text{wfactors } G a \text{ as set as} \subseteq \text{carrier } G$   
**using** *factor-mset-aux assms* **by** *blast*  
**obtain** *bs* **where** *bs-def*:  
 $\text{factor-mset } G b = \text{fmset } G b$   
 $\text{wfactors } G b \text{ bs set bs} \subseteq \text{carrier } G$   
**using** *factor-mset-aux assms(2)* **by** *blast*  
**have**  $a \otimes b \in \text{carrier } G$  **using** *assms* **by** *auto*  
**then obtain** *cs* **where** *cs-def*:  
 $\text{factor-mset } G (a \otimes b) = \text{fmset } G cs$   
 $\text{wfactors } G cs (a \otimes b)$   
 $\text{set } cs \subseteq \text{carrier } G$   
**using** *factor-mset-aux assms* **by** *blast*  
**have**  $\text{fmset } G cs = \text{fmset } G as + \text{fmset } G bs$   
**using** *as-def bs-def cs-def assms*  
**by** (*intro mult-wfactors-fmset[where a=a and b=b]*) *auto*  
**thus** *?thesis*  
**using** *as-def bs-def cs-def* **by** *auto*  
**qed**

**lemma** *factor-mset-unit*:  $\text{factor-mset } G \mathbf{1} = \{\#\}$   
**proof** –  
**have**  $\text{factor-mset } G \mathbf{1} = \text{factor-mset } G (\mathbf{1} \otimes \mathbf{1})$   
**by** *simp*  
**also have**  $\dots = \text{factor-mset } G \mathbf{1} + \text{factor-mset } G \mathbf{1}$   
**by** (*intro factor-mset-mult, auto*)  
**finally show**  $\text{factor-mset } G \mathbf{1} = \{\#\}$   
**by** *simp*  
**qed**

**lemma** *factor-mset-irred*:  
**assumes**  $x \in \text{carrier } G$  *irreducible*  $G$   $x$   
**shows**  $\text{factor-mset } G$   $x = \text{image-mset } (\text{assocs } G)$   $\{\#x\}$   
**proof** –  
**have**  $\text{wfactors } G$   $[x]$   $x$   
**using** *assms* **by** (*simp add:wfactors-def*)  
**hence**  $\text{factor-mset } G$   $x = \text{fmset } G$   $[x]$   
**using** *factor-mset-aux-1* *assms* **by** *simp*  
**also have**  $\dots = \text{image-mset } (\text{assocs } G)$   $\{\#x\}$   
**by** (*simp add:fmset-def*)  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *factor-mset-divides*:  
**assumes**  $a \in \text{carrier } G$   $b \in \text{carrier } G$   
**shows**  $a$  *divides*  $b \iff \text{factor-mset } G$   $a \subseteq\# \text{factor-mset } G$   $b$   
**proof** –  
**obtain** *as* **where** *as-def*:  
 $\text{factor-mset } G$   $a = \text{fmset } G$   $as$   
 $\text{wfactors } G$   $as$   $a$  *set*  $as \subseteq \text{carrier } G$   
**using** *factor-mset-aux* *assms* **by** *blast*  
**obtain** *bs* **where** *bs-def*:  
 $\text{factor-mset } G$   $b = \text{fmset } G$   $bs$   
 $\text{wfactors } G$   $bs$   $b$  *set*  $bs \subseteq \text{carrier } G$   
**using** *factor-mset-aux* *assms*(2) **by** *blast*  
**hence**  $a$  *divides*  $b \iff \text{fmset } G$   $as \subseteq\# \text{fmset } G$   $bs$   
**using** *as-def* *bs-def* *assms*  
**by** (*intro divides-as-fmsubset*) *auto*  
**also have**  $\dots \iff \text{factor-mset } G$   $a \subseteq\# \text{factor-mset } G$   $b$   
**using** *as-def* *bs-def* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *factor-mset-sim*:  
**assumes**  $a \in \text{carrier } G$   $b \in \text{carrier } G$   
**shows**  $a \sim b \iff \text{factor-mset } G$   $a = \text{factor-mset } G$   $b$   
**using** *factor-mset-divides* *assms*  
**by** (*simp add:associated-def*) *auto*

**lemma** *factor-mset-prod*:  
**assumes** *finite*  $A$   
**assumes**  $f ' A \subseteq \text{carrier } G$   
**shows**  $\text{factor-mset } G$   $(\otimes a \in A. f a) =$   
 $(\sum a \in A. \text{factor-mset } G (f a))$   
**using** *assms*  
**proof** (*induction A rule:finite-induct*)  
**case** *empty*  
**then show** *?case* **by** (*simp add:factor-mset-unit*)  
**next**

**case** (*insert x F*)  
**have** *factor-mset G (finprod G f (insert x F)) =*  
*factor-mset G (f x  $\otimes$  finprod G f F)*  
**using insert by** (*subst finprod-insert*) *auto*  
**also have** *... = factor-mset G (f x) + factor-mset G (finprod G f F)*  
**using insert by** (*intro factor-mset-mult finprod-closed*) *auto*  
**also have**  
*... = factor-mset G (f x) + ( $\sum a \in F.$  factor-mset G (f a))*  
**using insert by** *simp*  
**also have** *... = ( $\sum a \in \text{insert } x \text{ } F.$  factor-mset G (f a))*  
**using insert by** *simp*  
**finally show** *?case by simp*  
**qed**

**lemma** *factor-mset-pow:*  
**assumes** *a  $\in$  carrier G*  
**shows** *factor-mset G (a [ $\wedge$ ] n) = repeat-mset n (factor-mset G a)*  
**proof** (*induction n*)  
**case** *0*  
**then show** *?case by (simp add:factor-mset-unit)*  
**next**  
**case** (*Suc n*)  
**have** *factor-mset G (a [ $\wedge$ ] Suc n) = factor-mset G (a [ $\wedge$ ] n  $\otimes$  a)*  
**by** *simp*  
**also have** *... = factor-mset G (a [ $\wedge$ ] n) + factor-mset G a*  
**using** *assms by (intro factor-mset-mult) auto*  
**also have** *... = repeat-mset n (factor-mset G a) + factor-mset G a*  
**using** *Suc by simp*  
**also have** *... = repeat-mset (Suc n) (factor-mset G a)*  
**by** *simp*  
**finally show** *?case by simp*  
**qed**

**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))*  
**using** *assms*  
**by** (*induction F rule:finite-induct, simp, simp*)

**lemma** *decomp-mset:*  
*( $\sum x \in \text{set-mset } R.$  replicate-mset (count R x) x) = R*  
**by** (*rule multiset-eqI, simp add:count-sum count-eq-zero-iff*)

**lemma** *factor-mset-count:*  
**assumes** *a  $\in$  carrier G d  $\in$  carrier G irreducible G d*  
**shows** *count (factor-mset G a) (assoc G d) = multiplicity G d a*  
**proof** –  
**have** *a:*

*count (factor-mset G a) (assocs G d) ≥ m*  $\longleftrightarrow$  *d [∧] m divides a*  
(is ?lhs  $\longleftrightarrow$  ?rhs) **for** *m*

**proof** –

**have** ?lhs  $\longleftrightarrow$  *replicate-mset m (assocs G d) ⊆# factor-mset G a*  
**by** (*simp add:count-le-replicate-mset-subset-eq*)

**also have** ...  $\longleftrightarrow$  *factor-mset G (d [∧] m) ⊆# factor-mset G a*

**using** *assms(2,3)* **by** (*simp add:factor-mset-pow factor-mset-irred*)

**also have** ...  $\longleftrightarrow$  ?rhs

**using** *assms(1,2)* **by** (*subst factor-mset-divides*) *auto*

**finally show** ?thesis **by** *simp*

**qed**

**define** *M* **where** *M* = {(*m::nat*). *d [∧] m divides a*}

**have** *M-alt*: *M* = {*m. m ≤ count (factor-mset G a) (assocs G d)*}  
**using** *a* **by** (*simp add:M-def*)

**hence** *Max M* = *count (factor-mset G a) (assocs G d)*

**by** (*intro Max-eqI, auto*)

**thus** ?thesis

**unfolding** *multiplicity-def M-def* **by** *auto*

**qed**

**lemma** *multiplicity-ge-iff*:

**assumes** *d ∈ carrier G irreducible G d a ∈ carrier G*

**shows** *multiplicity G d a ≥ k*  $\longleftrightarrow$  *d [∧] k divides a*

(is ?lhs  $\longleftrightarrow$  ?rhs)

**proof** –

**have** ?lhs  $\longleftrightarrow$  *count (factor-mset G a) (assocs G d) ≥ k*

**using** *factor-mset-count[OF assms(3,1,2)]* **by** *simp*

**also have** ...  $\longleftrightarrow$  *replicate-mset k (assocs G d) ⊆# factor-mset G a*

**by** (*subst count-le-replicate-mset-subset-eq, simp*)

**also have** ...  $\longleftrightarrow$

*repeat-mset k (factor-mset G d) ⊆# factor-mset G a*

**by** (*subst factor-mset-irred[OF assms(1,2)], simp*)

**also have** ...  $\longleftrightarrow$  *factor-mset G (d [∧]<sub>G</sub> k) ⊆# factor-mset G a*

**by** (*subst factor-mset-pow[OF assms(1)], simp*)

**also have** ...  $\longleftrightarrow$  *(d [∧] k) divides<sub>G</sub> a*

**using** *assms(1) factor-mset-divides[OF - assms(3)]* **by** *simp*

**finally show** ?thesis **by** *simp*

**qed**

**lemma** *multiplicity-gt-0-iff*:

**assumes** *d ∈ carrier G irreducible G d a ∈ carrier G*

**shows** *multiplicity G d a > 0*  $\longleftrightarrow$  *d divides a*

**using** *multiplicity-ge-iff[OF assms(1,2,3), where k=1] assms*

**by** *auto*

**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**  $\text{count } (\text{factor-mset } G a) y = 0$   
**using**  $\text{factor-mset-set } [OF \text{ assms}(1)] \text{ assms}(2)$  **by**  $(\text{metis count-inI})$

**lemma** *factor-mset-choose*:

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

**have**  $b: \text{irreducible } G x$  **if**  $a: x \in \# R$  **for**  $x$

**proof** –

**have**  $x\text{-carr}: x \in \text{carrier } G$

**using**  $a \text{ assms}(2)$  **by** *auto*

**have**  $\text{assocs } G x \in \text{assocs } G \text{ ' set-mset } R$

**using**  $a$  **by** *simp*

**hence**  $\text{assocs } G x \in \# \text{factor-mset } G a$

**using**  $\text{assms}(3)$   $a$  *in-image-mset* **by** *metis*

**then obtain**  $z$  **where**  $z\text{-def}$ :

$z \in \text{carrier } G$   $\text{irreducible } G z$   $\text{assocs } G x = \text{assocs } G z$

**using**  $\text{factor-mset-set } \text{assms}(1)$  **by** *metis*

**have**  $z \sim x$  **using**  $z\text{-def}(1,3)$  *assocs-eqD*  $x\text{-carr}$  **by** *simp*

**thus**  $?thesis$  **using**  $z\text{-def}(1,2)$   $x\text{-carr}$  *irreducible-cong* **by** *simp*

**qed**

**have**  $\text{factor-mset } G ?rhs =$

$(\sum_{x \in \text{set-mset } R}. \text{factor-mset } G (x [\wedge] \text{count } R x))$

**using**  $\text{assms}(2)$  **by**  $(\text{subst } \text{factor-mset-prod}, \text{auto})$

**also have**  $\dots =$

$(\sum_{x \in \text{set-mset } R}. \text{repeat-mset } (\text{count } R x) (\text{factor-mset } G x))$

**using**  $\text{assms}(2)$  **by**  $(\text{intro } \text{sum.cong}, \text{auto } \text{simp } \text{add:factor-mset-pow})$

**also have**  $\dots = (\sum_{x \in \text{set-mset } R}.$

$\text{repeat-mset } (\text{count } R x) (\text{image-mset } (\text{assocs } G) \{\#x\#}))$

**using**  $\text{assms}(2)$   $b$  **by**  $(\text{intro } \text{sum.cong}, \text{auto } \text{simp } \text{add:factor-mset-irred})$

**also have**  $\dots = (\sum_{x \in \text{set-mset } R}.$

$\text{image-mset } (\text{assocs } G) (\text{replicate-mset } (\text{count } R x) x))$

**by** *simp*

**also have**  $\dots = \text{image-mset } (\text{assocs } G)$

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

**by**  $(\text{simp } \text{add: image-mset-sum})$

**also have**  $\dots = \text{image-mset } (\text{assocs } G) R$

**by**  $(\text{simp } \text{add:decomp-mset})$

**also have**  $\dots = \text{factor-mset } G a$

**using**  $\text{assms}$  **by** *simp*

**finally have**  $\text{factor-mset } G ?rhs = \text{factor-mset } G a$  **by** *simp*

**moreover have**  $(\bigotimes_{x \in \text{set-mset } R}. x [\wedge] \text{count } R x) \in \text{carrier } G$

**using**  $\text{assms}(2)$  **by**  $(\text{intro } \text{finprod-closed}, \text{auto})$

**ultimately show**  $?thesis$

**using**  $\text{assms}(1)$  **by**  $(\text{subst } \text{factor-mset-sim}) \text{auto}$

qed

**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** –

**have**  $\text{count } (\text{factor-mset } G a) d \leq \text{count } (\text{factor-mset } G b) d$  **for**  $d$

**proof** (*cases*  $\exists y \in \text{carrier } G. \text{irreducible } G y \wedge d = \text{assocs } G y$ )

**case** *True*

**then obtain**  $y$  **where**  $y$ -def:

*irreducible*  $G y$   $y \in \text{carrier } G$   $d = \text{assocs } G y$

**by** *blast*

**then obtain**  $z$  **where**  $z$ -def:  $z \in R$   $y \sim z$

**using** *assms*(3) **unfolding** *canonical-irreducibles-def* **by** *metis*

**have**  $z$ -more: *irreducible*  $G z$   $z \in \text{carrier } G$

**using**  $z$ -def(1) *assms*(3)

**unfolding** *canonical-irreducibles-def* **by** *auto*

**have**  $y \in \text{assocs } G z$  **using**  $z$ -def(2)  $z$ -more(2)  $y$ -def(2)

**by** (*simp add: closure-ofI2*)

**hence**  $d$ -def:  $d = \text{assocs } G z$

**using**  $y$ -def(2,3)  $z$ -more(2) *assocs-repr-independence*

**by** *blast*

**have**  $\text{count } (\text{factor-mset } G a) d = \text{multiplicity } G z a$

**unfolding**  $d$ -def

**by** (*intro factor-mset-count[OF assms(1) z-more(2,1)]*)

**also have**  $\dots \leq \text{multiplicity } G z b$

**using** *assms*(4)  $z$ -def(1) **by** *simp*

**also have**  $\dots = \text{count } (\text{factor-mset } G b) d$

**unfolding**  $d$ -def

**by** (*intro factor-mset-count[symmetric, OF assms(2) z-more(2,1)]*)

**finally show** *?thesis* **by** *simp*

**next**

**case** *False*

**have**  $\text{count } (\text{factor-mset } G a) d = 0$  **using** *False*

**by** (*intro factor-mset-count-2[OF assms(1)], simp*)

**moreover have**  $\text{count } (\text{factor-mset } G b) d = 0$  **using** *False*

**by** (*intro factor-mset-count-2[OF assms(2)], simp*)

**ultimately show** *?thesis* **by** *simp*

qed

**hence**  $\text{factor-mset } G a \subseteq\# \text{factor-mset } G b$

**unfolding** *subsetq-mset-def* **by** *simp*

**thus** *?thesis* **using** *factor-mset-divides assms(1,2)* **by** *simp*

qed

**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** (*cases*  $x \in\# \ A$ )  
**case** *True*  
**hence**  $(f \ y = f \ x \wedge y \in\# \ A) = (y = x)$  **for**  $y$   
**by** (*meson*  $\text{assms}(1) \ \text{assms}(3) \ \text{inj-onD} \ \text{subsetD}$ )  
**hence**  $(f \ -' \ \{f \ x\} \cap \text{set-mset } A) = \{x\}$   
**by** (*simp*  $\text{add:set-eq-iff}$ )  
**thus** *?thesis*  
**by** (*subst*  $\text{count-image-mset}, \ \text{simp}$ )  
**next**  
**case** *False*  
**hence**  $x \notin \text{set-mset } A$  **by** *simp*  
**hence**  $f \ x \notin f \ ' \ \text{set-mset } A$  **using**  $\text{assms}$   
**by** (*simp*  $\text{add:inj-on-image-mem-iff}$ )  
**hence**  $\text{count } (\text{image-mset } f \ A) \ (f \ x) = 0$   
**by** (*simp*  $\text{add:count-eq-zero-iff}$ )  
**thus** *?thesis* **by** (*metis*  $\text{count-inI} \ \text{False}$ )  
**qed**

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**

$\text{finite } \{d. d \in R \wedge \text{multiplicity } G \ d \ a > 0\}$   
 $a \sim (\bigotimes d \in \{d. d \in R \wedge \text{multiplicity } G \ d \ a > 0\}.$   
 $d \ [\bigwedge \text{multiplicity } G \ d \ a] \ (\text{is } a \sim \text{?rhs})$

**proof** –

**have** *r-1*:  $R \subseteq \{x. x \in \text{carrier } G \wedge \text{irreducible } G \ x\}$

**using**  $\text{assms}(1)$  **unfolding** *canonical-irreducibles-def* **by** *simp*

**have** *r-2*:  $\bigwedge x \ y. x \in R \implies y \in R \implies x \sim y \implies x = y$

**using**  $\text{assms}(1)$  **unfolding** *canonical-irreducibles-def* **by** *simp*

**have** *assocs-inj*:  $\text{inj-on } (\text{assocs } G) \ R$

**using** *r-1* *r-2* *assocs-eqD* **by** (*intro* *inj-onI*, *blast*)

**define**  $R'$  **where**

$R' = (\sum d \in \{d. d \in R \wedge \text{multiplicity } G \ d \ a > 0\}.$   
 $\text{replicate-mset } (\text{multiplicity } G \ d \ a) \ d)$

**have**  $\text{count } (\text{factor-mset } G \ a) \ (\text{assocs } G \ x) > 0$

**if**  $x \in R$   $0 < \text{multiplicity } G \ x \ a$  **for**  $x$

**using**  $\text{assms}$  *r-1* *r-2* **that**

**by** (*subst*  $\text{factor-mset-count}[OF \ \text{assms}(2)]$ ) *auto*

**hence**  $\text{assocs } G \ ' \ \{d \in R. 0 < \text{multiplicity } G \ d \ a\}$

$\subseteq \text{set-mset } (\text{factor-mset } G \ a)$

**by** (*intro* *image-subsetI*, *simp*)

**hence**  $a$ :finite (assoc $s$   $G$  ‘ { $d \in R$ .  $0 < \text{multiplicity } G \ d \ a$ })  
**using** finite-subset **by** auto

**show** finite { $d \in R$ .  $0 < \text{multiplicity } G \ d \ a$ }  
**using** assoc $s$ -inj inj-on-subset[OF assoc $s$ -inj]  
**by** (intro finite-imageD[OF  $a$ ], simp)

**hence** count- $R'$ :  
 count  $R' \ d =$  (if  $d \in R$  then multiplicity  $G \ d \ a$  else 0)  
**for**  $d$   
**by** (auto simp add: $R'$ -def count-sum)

**have** set- $R'$ : set-mset  $R' =$  { $d \in R$ .  $0 < \text{multiplicity } G \ d \ a$ }  
**unfolding** set-mset-def **using** count- $R'$  **by** auto

**have** count (image-mset (assoc $s$   $G$ )  $R'$ )  $x =$   
 count (factor-mset  $G \ a$ )  $x$  **for**  $x$   
**proof** (cases  $\exists x'. x' \in R \wedge x = \text{assoc}s \ G \ x'$ )  
**case** True  
**then obtain**  $x'$  **where**  $x'$ -def:  $x' \in R \ x = \text{assoc}s \ G \ x'$   
**by** blast  
**have** count (image-mset (assoc $s$   $G$ )  $R'$ )  $x =$  count  $R' \ x'$   
**using** assoc $s$ -inj inj-on-subset[OF assoc $s$ -inj]  $x'$ -def  
**by** (subst  $x'$ -def(2), subst count-image-mset-inj[OF assoc $s$ -inj])  
 (auto simp:set- $R'$ )  
**also have** ... = multiplicity  $G \ x' \ a$   
**using** count- $R' \ x'$ -def **by** simp  
**also have** ... = count (factor-mset  $G \ a$ ) (assoc $s$   $G \ x'$ )  
**using**  $x'$ -def(1) r-1  
**by** (subst factor-mset-count[OF assms(2)]) auto  
**also have** ... = count (factor-mset  $G \ a$ )  $x$   
**using**  $x'$ -def(2) **by** simp  
**finally show** ?thesis **by** simp

**next**  
**case** False  
**have**  $a$ : $x \neq \text{assoc}s \ G \ z$   
**if**  $a1$ :  $z \in \text{carrier } G$  **and**  $a2$ : irreducible  $G \ z$  **for**  $z$   
**proof** –  
**obtain**  $v$  **where**  $v$ -def:  $v \in R \ z \sim v$   
**using**  $a1 \ a2$  assms(1)  
**unfolding** canonical-irreducibles-def **by** auto  
**hence**  $z \in \text{assoc}s \ G \ v$   
**using**  $a1 \ r-1 \ v$ -def(1) **by** (simp add: closure-ofI2)  
**hence** assoc $s \ G \ z = \text{assoc}s \ G \ v$   
**using**  $a1 \ r-1 \ v$ -def(1) assoc $s$ -repr-independence  
**by** auto  
**moreover have**  $x \neq \text{assoc}s \ G \ v$   
**using** False  $v$ -def(1) **by** simp  
**ultimately show** ?thesis **by** simp

```

qed

have count (image-mset (assocs G) R') x = 0
  using False count-R' by (simp add: count-image-mset) auto
also have ... = count (factor-mset G a) x
  using a
  by (intro factor-mset-count-2[OF assms(2), symmetric]) auto
finally show ?thesis by simp
qed

hence image-mset (assocs G) R' = factor-mset G a
  by (rule multiset-eqI)

moreover have set-mset R'  $\subseteq$  carrier G
  using r-1 by (auto simp add:set-R')
ultimately have a  $\sim$  ( $\bigotimes_{x \in \text{set-mset } R'} x$  [ $\uparrow$ ] count R' x)
  using assms(2) by (intro factor-mset-choose, auto)
also have ... = ?rhs
  using set-R' assms r-1 r-2
  by (intro finprod-cong', auto simp add:count-R')
finally show a  $\sim$  ?rhs by simp
qed

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 (rule ccontr)
  assume a:  $\neg(1 < \text{order } R)$ 
  have  $\{0_R, 1_R\} \subseteq \text{carrier } R$  by auto
  hence card  $\{0_R, 1_R\} \leq \text{card } (\text{carrier } R)$ 
    using card-mono finite-carrier by blast
  also have ...  $\leq 1$  using a by (simp add:order-def)
  finally have card  $\{0_R, 1_R\} \leq 1$  by blast

```

thus *False* by *simp*  
qed

lemma (in *finite-field*) *order-pow-eq-self*:  
 assumes  $x \in \text{carrier } R$   
 shows  $x [\wedge] (\text{order } R) = x$   
 proof (cases  $x = 0$ )  
 case *True*  
 have  $\text{order } R > 0$   
 using *assms*(1) *order-gt-0-iff-finite finite-carrier* by *simp*  
 then obtain  $n$  where *n-def*:  $\text{order } R = \text{Suc } n$   
 using *lessE* by *blast*  
 have  $x [\wedge] (\text{order } R) = 0$   
 unfolding *n-def* using *True* by (*subst nat-pow-Suc, simp*)  
 thus ?thesis using *True* by *simp*  
 next  
 case *False*  
 have  $x\text{-carr}: x \in \text{carrier } (\text{mult-of } R)$   
 using *False assms* by *simp*

have *carr-non-empty*:  $\text{card } (\text{carrier } R) > 0$   
 using *order-gt-0-iff-finite finite-carrier*  
 unfolding *order-def* by *simp*  
 have  $x [\wedge] (\text{order } R) = x [\wedge]_{\text{mult-of } R} (\text{order } R)$   
 by (*simp add: nat-pow-mult-of*)  
 also have  $\dots = x [\wedge]_{\text{mult-of } R} (\text{order } (\text{mult-of } R) + 1)$   
 using *carr-non-empty* unfolding *order-def*  
 by (*intro arg-cong[where f= $\lambda t. x [\wedge]_{\text{mult-of } R } t$ ] (simp)*)  
 also have  $\dots = x$   
 using *x-carr*  
 by (*simp add: mult-of.pow-order-eq-1*)  
 finally show  $x [\wedge] (\text{order } R) = x$   
 by *simp*  
 qed

lemma (in *finite-field*) *order-pow-eq-self'*:  
 assumes  $x \in \text{carrier } R$   
 shows  $x [\wedge] (\text{order } R \wedge d) = x$   
 proof (induction  $d$ )  
 case 0  
 then show ?case using *assms* by *simp*  
 next  
 case (*Suc d*)  
 have  $x [\wedge] \text{order } R \wedge (\text{Suc } d) = x [\wedge] (\text{order } R \wedge d * \text{order } R)$   
 by (*simp add: mult.commute*)  
 also have  $\dots = (x [\wedge] (\text{order } R \wedge d)) [\wedge] \text{order } R$   
 using *assms* by (*simp add: nat-pow-pow*)  
 also have  $\dots = (x [\wedge] (\text{order } R \wedge d))$   
 using *order-pow-eq-self assms* by *simp*

also have ... =  $x$   
 using *Suc* by *simp*  
 finally show *?case* by *simp*  
 qed

end

lemma *finite-fieldI*:  
 assumes *field*  $R$   
 assumes *finite* (*carrier*  $R$ )  
 shows *finite-field*  $R$   
 using *assms*  
 unfolding *finite-field-def* *finite-field-axioms-def*  
 by *auto*

lemma (in *domain*) *finite-domain-units*:  
 assumes *finite* (*carrier*  $R$ )  
 shows  $Units\ R = carrier\ R - \{0\}$  (is *?lhs = ?rhs*)  
 proof  
 have  $Units\ R \subseteq carrier\ R$  by (*simp add:Units-def*)  
 moreover have  $0 \notin Units\ R$   
 by (*meson zero-is-prime(1) primeE*)  
 ultimately show  $Units\ R \subseteq carrier\ R - \{0\}$  by *blast*

next

have  $x \in Units\ R$  if  $a: x \in carrier\ R - \{0\}$  for  $x$   
 proof –  
 have  $x\text{-carr}: x \in carrier\ R$  using  $a$  by *blast*  
 define  $f$  where  $f = (\lambda y. y \otimes_R x)$   
 have *inj-on*  $f$  (*carrier*  $R$ ) unfolding *f-def*  
 by (*rule inj-onI, metis DiffD1 DiffD2 a m-rcancel insertI1*)  
 hence  $card\ (carrier\ R) = card\ (f\ ' carrier\ R)$   
 by (*metis card-image*)  
 moreover have  $f\ ' carrier\ R \subseteq carrier\ R$  unfolding *f-def*  
 by (*rule image-subsetI, simp add: ring.ring-simprules x-carr*)  
 ultimately have  $f\ ' carrier\ R = carrier\ R$   
 using *card-subset-eq assms* by *metis*  
 moreover have  $1_R \in carrier\ R$  by *simp*  
 ultimately have  $\exists y \in carrier\ R. f\ y = 1_R$   
 by (*metis image-iff*)  
 then obtain  $y$   
 where  $y\text{-carrier}: y \in carrier\ R$   
 and  $y\text{-left-inv}: y \otimes_R x = 1_R$   
 using *f-def* by *blast*  
 hence  $y\text{-right-inv}: x \otimes_R y = 1_R$   
 by (*metis DiffD1 a cring-simprules(14)*)  
 show  $x \in Units\ R$   
 using  $y\text{-carrier}$   $y\text{-left-inv}$   $y\text{-right-inv}$   
 by (*metis DiffD1 a divides-one factor-def*)  
 qed

```

thus ?rhs  $\subseteq$  ?lhs by auto
qed

```

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 -
  interpret domain R using assms by auto
  have Units R = carrier R - {0R}
    using finite-domain-units[OF assms(2)] by simp
  then have field R
    by (simp add: assms(1) field.intro field-axioms.intro)
  thus ?thesis
    using assms(2) finite-fieldI by auto
qed

```

```

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

```

```

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

```

```

private abbreviation I where I  $\equiv$  IdlZ {int n}

```

```

private lemma ideal-I: ideal I Z
  by (simp add: int.genideal-ideal)

```

```

lemma int-cosetI:
  assumes u mod (int n) = v mod (int n)
  shows IdlZ {int n} +>Z u = IdlZ {int n} +>Z v
proof -
  have u - v  $\in$  I
    by (metis Idl-subset-eq-dvd assms int-Idl-subset-ideal mod-eq-dvd-iff)
  thus ?thesis
    using ideal-I int.quotient-eq-iff-same-a-r-cos by simp
qed

```

```

lemma zfact-iso-inj:
  inj-on (zfact-iso n) {.. $n$ }
proof (rule inj-onI)
  fix x y
  assume a:x  $\in$  {.. $n$ } y  $\in$  {.. $n$ }
  assume zfact-iso n x = zfact-iso n y
  hence I +>Z (int x) = I +>Z (int y)

```

by (simp add:zfact-iso-def)  
 hence  $\text{int } x - \text{int } y \in I$   
 by (subst int.quotient-eq-iff-same-a-r-cos[OF ideal-I], auto)  
 hence  $\text{int } x \bmod \text{int } n = \text{int } y \bmod \text{int } n$   
 by (meson Idl-subset-eq-dvd int-Idl-subset-ideal mod-eq-dvd-iff)  
 thus  $x = y$   
 using a by simp  
 qed

**lemma** zfact-iso-ran:  
 $\text{zfact-iso } n \text{ ' } \{..<n\} = \text{carrier } (\text{ZFact } (\text{int } n))$   
**proof** –  
 have  $\text{zfact-iso } n \text{ ' } \{..<n\} \subseteq \text{carrier } (\text{ZFact } (\text{int } n))$   
 unfolding zfact-iso-def ZFact-def FactRing-simps  
 using int.a-rcosetsI by auto  
 moreover have  $x \in \text{zfact-iso } n \text{ ' } \{..<n\}$   
 if  $a:x \in \text{carrier } (\text{ZFact } (\text{int } n))$  for x  
**proof** –  
 obtain y where y-def:  $x = I +>_{\mathcal{Z}} y$   
 using a unfolding ZFact-def FactRing-simps by auto  
 define z where  $\langle z = \text{nat } (y \bmod \text{int } n) \rangle$   
 with n-gt-0 have z-def:  $\langle \text{int } z \bmod \text{int } n = y \bmod \text{int } n \rangle \langle z < n \rangle$   
 by (simp-all add: z-def nat-less-iff)  
 have  $x = I +>_{\mathcal{Z}} y$   
 by (simp add:y-def)  
 also have  $\dots = I +>_{\mathcal{Z}} (\text{int } z)$   
 by (intro int-cosetI, simp add:z-def)  
 also have  $\dots = \text{zfact-iso } n \ z$   
 by (simp add:zfact-iso-def)  
 finally have  $x = \text{zfact-iso } n \ z$   
 by simp  
 thus  $x \in \text{zfact-iso } n \text{ ' } \{..<n\}$   
 using z-def(2) by blast  
 qed  
 ultimately show ?thesis by auto  
 qed

**lemma** zfact-iso-bij:  
 $\text{bij-betw } (\text{zfact-iso } n) \ \{..<n\} \ (\text{carrier } (\text{ZFact } (\text{int } n)))$   
 using bij-betw-def zfact-iso-inj zfact-iso-ran by blast

**lemma** card-zfact-carr:  $\text{card } (\text{carrier } (\text{ZFact } (\text{int } n))) = n$   
 using bij-betw-same-card[OF zfact-iso-bij] by simp

**lemma** fin-zfact:  $\text{finite } (\text{carrier } (\text{ZFact } (\text{int } n)))$   
 using card-zfact-carr n-gt-0 card-ge-0-finite by force

end

**lemma** *zfact-prime-is-finite-field*:  
**assumes** *Factorial-Ring.prime*  $p$   
**shows** *finite-field* ( $ZFact$  ( $int$   $p$ ))  
**proof** –  
**have**  $p-gt-0$ :  $p > 0$  **using** *assms(1)* *prime-gt-0-nat* **by** *simp*  
**have** *Factorial-Ring.prime* ( $int$   $p$ )  
**using** *assms* **by** *simp*  
**moreover** **have** *finite* (*carrier* ( $ZFact$  ( $int$   $p$ )))  
**using** *fin-zfact[OF p-gt-0]* **by** *simp*  
**ultimately** **show** *?thesis*  
**by** (*intro* *finite-domains-are-fields* *ZFact-prime-is-domain*, *auto*)  
**qed**

**definition** *int-embed* ::  $- \Rightarrow int \Rightarrow -$  **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* ( $R$  ( $\lfloor$  *carrier* :=  $K$   $\rfloor$ ))  $i$   $k$   
*(is ?lhs = ?rhs)*  
**proof** –  
**have**  $a$ : *subgroup*  $K$  (*add-monoid*  $R$ )  
**using** *assms(1)* *subring.axioms* **by** *auto*  
**have** *add-pow*  $R$   $i$   $k$  =  $k$  [ $\wedge$  *add-monoid*  $R$  ( $\lfloor$  *carrier* :=  $K$   $\rfloor$ )]  $i$   
**using** *add.int-pow-consistent[OF a assms(2)]* **by** *simp*  
**also** **have** ... = *?rhs*  
**unfolding** *add-pow-def* **by** *simp*  
**finally** **show** *?thesis* **by** *simp*  
**qed**

**lemma** (*in ring*) *int-embed-consistent*:  
**assumes** *subring*  $K$   $R$   
**shows** *int-embed*  $R$   $i$  = *int-embed* ( $R$  ( $\lfloor$  *carrier* :=  $K$   $\rfloor$ ))  $i$   
**proof** –  
**have**  $a$ :  $\mathbf{1} = \mathbf{1}_R$  ( $\lfloor$  *carrier* :=  $K$   $\rfloor$ ) **by** *simp*  
**have**  $b$ :  $\mathbf{1}_{R(\lfloor carrier := K \rfloor)} \in K$   
**using** *assms* *subringE(3)* **by** *auto*  
**show** *?thesis*  
**unfolding** *int-embed-def a* **using**  $b$  *add-pow-consistent[OF assms(1)]*  
**by** *simp*  
**qed**

**lemma** (*in ring*) *int-embed-closed*:  
*int-embed*  $R$   $k \in carrier$   $R$   
**unfolding** *int-embed-def* **using** *add.int-pow-closed* **by** *simp*

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

**assumes** *subring*  $K R$   
**shows** *int-embed*  $R k \in K$   
**proof** –  
**let**  $?R' = R \langle \text{carrier} := K \rangle$   
**interpret**  $x:\text{ring } ?R'$   
**using** *subring-is-ring*[*OF assms*] **by** *simp*  
**have** *int-embed*  $R k = \text{int-embed } ?R' k$   
**using** *int-embed-consistent*[*OF assms*] **by** *simp*  
**also have**  $\dots \in K$   
**using** *x.int-embed-closed* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** (**in** *ring*) *int-embed-zero*:  
*int-embed*  $R 0 = \mathbf{0}_R$   
**by** (*simp add:int-embed-def add-pow-def*)

**lemma** (**in** *ring*) *int-embed-one*:  
*int-embed*  $R 1 = \mathbf{1}_R$   
**by** (*simp add:int-embed-def*)

**lemma** (**in** *ring*) *int-embed-add*:  
*int-embed*  $R (x+y) = \text{int-embed } R x \oplus_R \text{int-embed } R y$   
**by** (*simp add:int-embed-def add.int-pow-mult*)

**lemma** (**in** *ring*) *int-embed-inv*:  
*int-embed*  $R (-x) = \ominus_R \text{int-embed } R x$  (**is** *?lhs = ?rhs*)  
**proof** –  
**have** *?lhs*  $= \text{int-embed } R (-x) \oplus (\text{int-embed } R x \ominus \text{int-embed } R x)$   
**using** *int-embed-closed* **by** *simp*  
**also have**  
 $\dots = \text{int-embed } R (-x) \oplus \text{int-embed } R x \oplus (\ominus \text{int-embed } R x)$   
**using** *int-embed-closed* **by** (*subst a-minus-def, subst a-assoc, auto*)  
**also have**  $\dots = \text{int-embed } R (-x + x) \oplus (\ominus \text{int-embed } R x)$   
**by** (*subst int-embed-add, simp*)  
**also have**  $\dots = ?rhs$   
**using** *int-embed-closed*  
**by** (*simp add:int-embed-zero*)  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** (**in** *ring*) *int-embed-diff*:  
*int-embed*  $R (x-y) = \text{int-embed } R x \ominus_R \text{int-embed } R y$   
(**is** *?lhs = ?rhs*)  
**proof** –  
**have** *?lhs*  $= \text{int-embed } R (x + (-y))$  **by** *simp*  
**also have**  $\dots = ?rhs$   
**by** (*subst int-embed-add, simp add:a-minus-def int-embed-inv*)  
**finally show** *?thesis* **by** *simp*

qed

**lemma** (in ring) *int-embed-mult-aux*:  
 $int-embed\ R\ (x * int\ y) = int-embed\ R\ x \otimes int-embed\ R\ y$   
**proof** (induction y)  
  case 0  
  then show ?case by (simp add: int-embed-closed int-embed-zero)  
next  
  case (Suc y)  
  have  $int-embed\ R\ (x * int\ (Suc\ y)) = int-embed\ R\ (x + x * int\ y)$   
  by (simp add: algebra-simps)  
  also have  $\dots = int-embed\ R\ x \oplus int-embed\ R\ (x * int\ y)$   
  by (subst int-embed-add, simp)  
  also have  
   $\dots = int-embed\ R\ x \otimes \mathbf{1} \oplus int-embed\ R\ x \otimes int-embed\ R\ y$   
  using int-embed-closed  
  by (subst Suc, simp)  
  also have  $\dots = int-embed\ R\ x \otimes (int-embed\ R\ 1 \oplus int-embed\ R\ y)$   
  using int-embed-closed by (subst r-distr, simp-all add: int-embed-one)  
  also have  $\dots = int-embed\ R\ x \otimes int-embed\ R\ (1 + int\ y)$   
  by (subst int-embed-add, simp)  
  also have  $\dots = int-embed\ R\ x \otimes int-embed\ R\ (Suc\ y)$   
  by simp  
  finally show ?case by simp  
qed

**lemma** (in ring) *int-embed-mult*:  
 $int-embed\ R\ (x * y) = int-embed\ R\ x \otimes_R int-embed\ R\ y$   
**proof** (cases y  $\geq$  0)  
  case True  
  then obtain y' where y-def:  $y = int\ y'$   
  using nonneg-int-cases by auto  
  have  $int-embed\ R\ (x * y) = int-embed\ R\ (x * int\ y')$   
  unfolding y-def by simp  
  also have  $\dots = int-embed\ R\ x \otimes int-embed\ R\ y'$   
  by (subst int-embed-mult-aux, simp)  
  also have  $\dots = int-embed\ R\ x \otimes int-embed\ R\ y$   
  unfolding y-def by simp  
  finally show ?thesis by simp  
next  
  case False  
  then obtain y' where y-def:  $y = - int\ y'$   
  by (meson nle-le nonpos-int-cases)  
  have  $int-embed\ R\ (x * y) = int-embed\ R\ (-(x * int\ y'))$   
  unfolding y-def by simp  
  also have  $\dots = \ominus (int-embed\ R\ (x * int\ y'))$   
  by (subst int-embed-inv, simp)  
  also have  $\dots = \ominus (int-embed\ R\ x \otimes int-embed\ R\ y')$   
  by (subst int-embed-mult-aux, simp)

**also have**  $\dots = \text{int-embed } R \ x \otimes \oplus \text{ int-embed } R \ y'$   
**using** *int-embed-closed* **by** *algebra*  
**also have**  $\dots = \text{int-embed } R \ x \otimes \text{int-embed } R \ (-y')$   
**by** (*subst int-embed-inv, simp*)  
**also have**  $\dots = \text{int-embed } R \ x \otimes \text{int-embed } R \ y$   
**unfolding** *y-def* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** (*in ring*) *int-embed-ring-hom*:  
*ring-hom-ring int-ring R (int-embed R)*  
**proof** (*rule ring-hom-ringI*)  
**show** *ring int-ring* **using** *int.ring-axioms* **by** *simp*  
**show** *ring R* **using** *ring-axioms* **by** *simp*  
**show** *int-embed R x ∈ carrier R if x ∈ carrier Z for x*  
**using** *int-embed-closed* **by** *simp*  
**show** *int-embed R (x ⊗<sub>Z</sub> y) = int-embed R x ⊗ int-embed R y*  
**if** *x ∈ carrier Z y ∈ carrier Z for x y*  
**using** *int-embed-mult* **by** *simp*  
**show** *int-embed R (x ⊕<sub>Z</sub> y) = int-embed R x ⊕ int-embed R y*  
**if** *x ∈ carrier Z y ∈ carrier Z for x y*  
**using** *int-embed-add* **by** *simp*  
**show** *int-embed R 1<sub>Z</sub> = 1*  
**by** (*simp add:int-embed-one*)  
**qed**

**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**  $\text{char } R \leq x$   $\text{char } R > 0$   
**proof** –  
**have** *char-subring R ⊆ int-embed R ‘ ({0..<int x})*

```

proof (rule image-subsetI)
  fix  $y :: \text{int}$ 
  assume  $y \in \text{UNIV}$ 
  define  $u$  where  $u = y \text{ div } (\text{int } x)$ 
  define  $v$  where  $v = y \text{ mod } (\text{int } x)$ 
  have  $\text{int } x > 0$  using assms by simp
  hence  $y\text{-exp}: y = u * \text{int } x + v \ v \geq 0 \ v < \text{int } x$ 
    unfolding  $u\text{-def } v\text{-def}$  by simp-all
  have  $\text{int-embed } R \ y = \text{int-embed } R \ v$ 
    using int-embed-closed unfolding  $y\text{-exp}$ 
    by (simp add:int-embed-mult int-embed-add assms(2))
  also have  $\dots \in \text{int-embed } R \ ' \ (\{0..<\text{int } x\})$ 
    using  $y\text{-exp}(2,3)$  by simp
  finally show  $\text{int-embed } R \ y \in \text{int-embed } R \ ' \ \{0..<\text{int } x\}$ 
    by simp
qed
hence  $a:\text{char-subring } R = \text{int-embed } R \ ' \ \{0..<\text{int } x\}$ 
  by auto
hence  $\text{char } R = \text{card } (\text{int-embed } R \ ' \ (\{0..<\text{int } x\}))$ 
  unfolding  $\text{char-def } a$  by simp
also have  $\dots \leq \text{card } \{0..<\text{int } x\}$ 
  by (intro card-image-le, simp)
also have  $\dots = x$  by simp
finally show  $\text{char } R \leq x$  by simp
have  $1 = \text{card } \{\text{int-embed } R \ 0\}$  by simp
also have  $\dots \leq \text{card } (\text{int-embed } R \ ' \ \{0..<\text{int } x\})$ 
  using assms(1) by (intro card-mono finite-imageI, simp-all)
also have  $\dots = \text{char } R$ 
  unfolding  $\text{char-def } a$  by simp
finally show  $\text{char } R > 0$  by simp
qed

```

```

lemma (in ring) embed-char-eq-0:
   $\text{int-embed } R \ (\text{int } (\text{char } R)) = \mathbf{0}$ 
proof (cases finite (char-subring R))
  case True
  interpret  $h: \text{ring-hom-ring int-ring } R \ (\text{int-embed } R)$ 
    using int-embed-ring-hom by simp

  define  $A$  where  $A = \{0..\text{int } (\text{char } R)\}$ 
  have  $\text{card } (\text{int-embed } R \ ' \ A) \leq \text{card } (\text{char-subring } R)$ 
    by (intro card-mono[OF True] image-subsetI, simp)
  also have  $\dots = \text{char } R$ 
    unfolding  $\text{char-def}$  by simp
  also have  $\dots < \text{card } A$ 
    unfolding  $A\text{-def}$  by simp
  finally have  $\text{card } (\text{int-embed } R \ ' \ A) < \text{card } A$  by simp
  hence  $\neg \text{inj-on } (\text{int-embed } R) \ A$ 
    using pigeonhole by simp

```

```

then obtain  $x y$  where  $xy$ :
   $x \in A \ y \in A \ x \neq y \text{ int-embed } R \ x = \text{int-embed } R \ y$ 
  unfolding  $\text{inj-on-def}$  by  $\text{auto}$ 
define  $v$  where  $v = \text{nat } (\max x y - \min x y)$ 
have  $a:\text{int-embed } R \ v = \mathbf{0}$ 
  using  $xy \text{ int-embed-closed}$ 
  by  $(\text{cases } x < y, \text{simp-all add:int-embed-diff } v\text{-def})$ 
moreover have  $v > 0$ 
  using  $xy$  by  $(\text{cases } x < y, \text{simp-all add:v-def})$ 
ultimately have  $\text{char } R \leq v$  using  $\text{char-bound}$  by  $\text{simp}$ 
moreover have  $v \leq \text{char } R$ 
  using  $xy \ v\text{-def } A\text{-def}$  by  $(\text{cases } x < y, \text{simp-all})$ 
ultimately have  $\text{char } R = v$  by  $\text{simp}$ 
then show  $?thesis$  using  $a$  by  $\text{simp}$ 
next
  case  $\text{False}$ 
  hence  $\text{char } R = 0$ 
  unfolding  $\text{char-def}$  by  $\text{simp}$ 
  then show  $?thesis$  by  $(\text{simp add:int-embed-zero})$ 
qed

lemma  $(\text{in ring}) \ \text{embed-char-eq-0-iff}$ :
  fixes  $n :: \text{int}$ 
  shows  $\text{int-embed } R \ n = \mathbf{0} \iff \text{char } R \ \text{dvd } n$ 
proof  $(\text{cases } \text{char } R > 0)$ 
  case  $\text{True}$ 
  define  $r$  where  $r = n \ \text{mod } \text{char } R$ 
  define  $s$  where  $s = n \ \text{div } \text{char } R$ 
  have  $rs$ :  $r < \text{char } R \ r \geq 0 \ n = r + s * \text{char } R$ 
  using  $\text{True}$  by  $(\text{simp-all add:r-def } s\text{-def})$ 

  have  $\text{int-embed } R \ n = \text{int-embed } R \ r$ 
  using  $\text{int-embed-closed}$  unfolding  $rs(3)$ 
  by  $(\text{simp add: int-embed-add int-embed-mult embed-char-eq-0})$ 

  moreover have  $\text{nat } r < \text{char } R$  using  $rs$  by  $\text{simp}$ 
  hence  $\text{int-embed } R \ (\text{nat } r) \neq \mathbf{0} \vee \text{nat } r = 0$ 
  using  $\text{True char-bound not-less}$  by  $\text{blast}$ 
  hence  $\text{int-embed } R \ r \neq \mathbf{0} \vee r = 0$ 
  using  $rs$  by  $\text{simp}$ 

  ultimately have  $\text{int-embed } R \ n = \mathbf{0} \iff r = 0$ 
  using  $\text{int-embed-zero}$  by  $\text{auto}$ 
  also have  $r = 0 \iff \text{char } R \ \text{dvd } n$ 
  using  $r\text{-def}$  by  $\text{auto}$ 
  finally show  $?thesis$  by  $\text{simp}$ 
next
  case  $\text{False}$ 
  hence  $\text{char } R = 0$  by  $\text{simp}$ 

```

**hence**  $a:x > 0 \implies \text{int-embed } R (\text{int } x) \neq \mathbf{0}$  **for**  $x$   
**using** *char-bound* **by** *auto*

**have**  $c:\text{int-embed } R (\text{abs } x) \neq \mathbf{0} \iff \text{int-embed } R x \neq \mathbf{0}$  **for**  $x$   
**using** *int-embed-closed*  
**by** (*cases*  $x > 0$ , *simp*, *simp add:int-embed-inv*)

**have**  $\text{int-embed } R x \neq \mathbf{0}$  **if**  $b:x \neq 0$  **for**  $x$   
**proof** –  
**have**  $\text{nat } (\text{abs } x) > 0$  **using**  $b$  **by** *simp*  
**hence**  $\text{int-embed } R (\text{nat } (\text{abs } x)) \neq \mathbf{0}$   
**using**  $a$  **by** *blast*  
**hence**  $\text{int-embed } R (\text{abs } x) \neq \mathbf{0}$  **by** *simp*  
**thus** *?thesis* **using**  $c$  **by** *simp*  
**qed**

**hence**  $\text{int-embed } R n = \mathbf{0} \iff n = 0$   
**using** *int-embed-zero* **by** *auto*  
**also have**  $n = 0 \iff \text{char } R \text{ dvd } n$  **using** *False* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

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** (*rule ccontr*)  
**have**  $\neg(\text{char } R = 1)$   
**using** *embed-char-eq-0 int-embed-one* **by** *auto*  
**hence**  $\neg(\text{char } R \text{ dvd } 1)$  **using** *assms(1)* **by** *simp*  
**moreover assume**  $\neg(\text{prime } (\text{char } R))$   
**hence**  $\neg(\text{irreducible } (\text{char } R))$   
**using** *irreducible-imp-prime-elem-gcd prime-elem-nat-iff* **by** *blast*  
**ultimately obtain**  $p \ q$  **where** *pq-def*:  $p * q = \text{char } R$   $p > 1$   $q > 1$   
**using** *assms*  
**unfolding** *Factorial-Ring.irreducible-def* **by** *auto*  
**have**  $\text{int-embed } R p \otimes \text{int-embed } R q = \mathbf{0}$   
**using** *embed-char-eq-0 pq-def*  
**by** (*subst int-embed-mult[symmetric]*) (*metis of-nat-mult*)  
**hence**  $\text{int-embed } R p = \mathbf{0} \vee \text{int-embed } R q = \mathbf{0}$   
**using** *integral int-embed-closed* **by** *simp*  
**hence**  $p * q \leq p \vee p * q \leq q$   
**using** *char-bound pq-def* **by** *auto*  
**thus** *False*  
**using** *pq-def(2,3)* **by** *simp*  
**qed**

**lemma** (**in** *ring*) *char-ring-is-subring*:  
**subring** ( $\text{char-subring } R$ )  $R$   
**proof** –

**have** *subring* (*int-embed*  $R \text{ ' carrier int-ring}$ )  $R$   
**by** (*intro ring.carrier-is-subring int.ring-axioms*  
*ring-hom-ring.img-is-subring[OF int-embed-ring-hom]*)  
**thus** *?thesis* **by** *simp*  
**qed**

**lemma** (*in cring*) *char-ring-is-subring*:  
*subring* (*char-subring*  $R$ )  $R$   
**using** *subringI'[OF char-ring-is-subring]* **by** *auto*

**lemma** (*in domain*) *char-ring-is-subdomain*:  
*subdomain* (*char-subring*  $R$ )  $R$   
**using** *subdomainI'[OF char-ring-is-subring]* **by** *auto*

**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**  $f \text{ ' } A = B$   
**using** *assms* **by** *force*

This is the binomial expansion theorem for commutative rings.

**lemma** (*in cring*) *binomial-expansion*:  
**fixes**  $n :: \text{nat}$   
**assumes** [*simp*]:  $x \in \text{carrier } R \ y \in \text{carrier } R$   
**shows**  $(x \oplus y) [\wedge] n =$   
 $(\bigoplus k \in \{..n\}. \text{int-embed } R \ (n \text{ choose } k) \otimes x [\wedge] k \otimes y [\wedge] (n-k))$   
**proof** –  
**define**  $A$  **where**  $A = (\lambda k. \{A. A \subseteq \{..<n\} \wedge \text{card } A = k\})$

**have** *fin-A: finite* ( $A \ i$ ) **for**  $i$   
**unfolding**  $A\text{-def}$  **by** *simp*  
**have** *disj-A: pairwise*  $(\lambda i \ j. \text{disjnt } (A \ i) \ (A \ j)) \ \{..n\}$   
**unfolding** *pairwise-def disjnt-def A-def* **by** *auto*  
**have** *card-A: B ∈ A i ⇒ card B = i* **if**  $i \in \{..n\}$  **for**  $i \ B$   
**unfolding**  $A\text{-def}$  **by** *simp*  
**have** *card-A2: card* ( $A \ i$ ) =  $(n \text{ choose } i)$  **if**  $i \in \{..n\}$  **for**  $i$   
**unfolding**  $A\text{-def}$  **using** *n-subsets[where A={..<n}]* **by** *simp*

**have** *card-bound: card*  $A \leq n$   
**if**  $A \subseteq \{..<n\}$  **for**  $n \ A$   
**by** (*metis card-lessThan finite-lessThan card-mono that*)  
**have** *card-insert: card* (*insert*  $n \ A$ ) =  $\text{card } A + 1$   
**if**  $A \subseteq \{..<(n::\text{nat})\}$  **for**  $n \ A$   
**using** *finite-subset that* **by** (*subst card-insert-disjoint, auto*)

**have** *embed-distr: [m] · y = int-embed*  $R \ (\text{int } m) \otimes y$   
**if**  $y \in \text{carrier } R$  **for**  $m \ y$   
**unfolding** *int-embed-def add-pow-def* **using** *that*  
**by** (*simp add:add-pow-def[symmetric] int-pow-int add-pow-ldistr*)

```

have (x ⊕ y) [∧] n =
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n-card A))
proof (induction n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  have s1:
    insert n ‘ Pow {..<n} = {A. A ⊆ {..<n+1} ∧ n ∈ A}
    by (intro image-set-eqI[where g=λx. x ∩ {..<n}], auto)
  have s2:
    Pow {..<n} = {A. A ⊆ {..<n+1} ∧ n ∉ A}
    using lessThan-Suc by auto

have (x ⊕ y) [∧] Suc n = (x ⊕ y) [∧] n ⊗ (x ⊕ y) by simp
also have ... =
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n-card A)) ⊗
  (x ⊕ y)
  by (subst Suc, simp)
also have ... =
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n-card A)) ⊗ x ⊕
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n-card A)) ⊗ y
  by (subst r-distr, auto)
also have ... =
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n-card A) ⊗ x) ⊕
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n-card A) ⊗ y)
  by (simp add:finsum-ldistr)
also have ... =
  (⊕ A ∈ Pow {..<n}. x [∧] (card A+1) ⊗ y [∧] (n-card A)) ⊕
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n-card A+1))
  using m-assoc m-comm
  by (intro arg-cong2[where f=(⊕)] finsum-cong', auto)
also have ... =
  (⊕ A ∈ Pow {..<n}. x [∧] (card (insert n A))
    ⊗ y [∧] (n+1-card (insert n A))) ⊕
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n+1-card A))
  using finite-subset card-bound card-insert Suc-diff-le
  by (intro arg-cong2[where f=(⊕)] finsum-cong', simp-all)
also have ... =
  (⊕ A ∈ insert n ‘ Pow {..<n}. x [∧] (card A)
    ⊗ y [∧] (n+1-card A)) ⊕
  (⊕ A ∈ Pow {..<n}. x [∧] (card A) ⊗ y [∧] (n+1-card A))
  by (subst finsum-reindex, auto simp add:inj-on-def)
also have ... =
  (⊕ A ∈ {A. A ⊆ {..<n+1} ∧ n ∈ A}.
    x [∧] (card A) ⊗ y [∧] (n+1-card A)) ⊕
  (⊕ A ∈ {A. A ⊆ {..<n+1} ∧ n ∉ A}.
    x [∧] (card A) ⊗ y [∧] (n+1-card A))

```

by (intro arg-cong2[where f=( $\oplus$ )] finsum-cong' s1 s2, simp-all)  
 also have ... = ( $\bigoplus A \in$   
 $\{A. A \subseteq \{..<n+1\} \wedge n \in A\} \cup \{A. A \subseteq \{..<n+1\} \wedge n \notin A\}$ .  
 $x [\wedge] (\text{card } A) \otimes y [\wedge] (n+1 - \text{card } A)$ )  
 by (subst finsum-Un-disjoint, auto)  
 also have ... =  
 $(\bigoplus A \in \text{Pow } \{..<n+1\}. x [\wedge] (\text{card } A) \otimes y [\wedge] (n+1 - \text{card } A))$   
 by (intro finsum-cong', auto)  
 finally show ?case by simp  
 qed  
 also have ... =  
 $(\bigoplus A \in (\bigcup (A \text{ ' } \{..n\})). x [\wedge] (\text{card } A) \otimes y [\wedge] (n - \text{card } A))$   
 using card-bound by (intro finsum-cong', auto simp add:A-def)  
 also have ... =  
 $(\bigoplus k \in \{..n\}. (\bigoplus A \in A k. x [\wedge] (\text{card } A) \otimes y [\wedge] (n - \text{card } A)))$   
 using fin-A disj-A by (subst add.finprod-UN-disjoint, auto)  
 also have ... =  $(\bigoplus k \in \{..n\}. (\bigoplus A \in A k. x [\wedge] k \otimes y [\wedge] (n - k)))$   
 using card-A by (intro finsum-cong', auto)  
 also have ... =  
 $(\bigoplus k \in \{..n\}. \text{int-embed } R (\text{card } (A k)) \otimes x [\wedge] k \otimes y [\wedge] (n - k))$   
 using int-embed-closed  
 by (subst add.finprod-const, simp-all add:embed-distr m-assoc)  
 also have ... =  
 $(\bigoplus k \in \{..n\}. \text{int-embed } R (n \text{ choose } k) \otimes x [\wedge] k \otimes y [\wedge] (n - k))$   
 using int-embed-closed card-A2 by (intro finsum-cong', simp-all)  
 finally show ?thesis by simp  
 qed

lemma bin-prime-factor:

assumes prime p  
 assumes  $k > 0$   $k < p$   
 shows p dvd (p choose k)

proof –

have p dvd fact p  
 using assms(1) prime-dvd-fact-iff by auto  
 hence p dvd fact k \* fact (p - k) \* (p choose k)  
 using binomial-fact-lemma assms by simp  
 hence p dvd fact k  $\vee$  p dvd fact (p - k)  $\vee$  p dvd (p choose k)  
 by (simp add: assms(1) prime-dvd-mult-eq-nat)  
 thus p dvd (p choose k)  
 using assms(1,2,3) prime-dvd-fact-iff by auto

qed

theorem (in domain) freshmans-dream:

assumes char R > 0  
 assumes [simp]:  $x \in \text{carrier } R$   $y \in \text{carrier } R$   
 shows  $(x \oplus y) [\wedge] (\text{char } R) = x [\wedge] \text{char } R \oplus y [\wedge] \text{char } R$   
 (is ?lhs = ?rhs)

proof –

```

have  $c$ :prime (char R)
  using assms(1) characteristic-is-prime by auto
have  $a$ :int-embed R (char R choose  $i$ ) = 0
  if  $i \in \{..char R\} - \{0, char R\}$  for  $i$ 
proof -
  have  $i > 0$   $i < char R$  using that by auto
  hence char R dvd char R choose  $i$ 
  using  $c$  bin-prime-factor by simp
  thus ?thesis using embed-char-eq-0-iff by simp
qed

have  $?lhs = (\bigoplus k \in \{..char R\}. int-embed R (char R choose k)$ 
   $\otimes x [\wedge k \otimes y [\wedge (char R - k)])$ 
  using binomial-expansion[OF assms(2,3)] by simp
also have  $... = (\bigoplus k \in \{0, char R\}. int-embed R (char R choose k)$ 
   $\otimes x [\wedge k \otimes y [\wedge (char R - k)])$ 
  using  $a$  int-embed-closed
  by (intro add.finprod-mono-neutral-cong-right, simp, simp-all)
also have  $... = ?rhs$ 
  using int-embed-closed assms(1) by (simp add:int-embed-one a-comm)
finally show ?thesis by simp
qed

```

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  $?lhs = ?rhs$ )
  unfolding  $n$ -def
proof (induction m)
  case 0
  then show ?case by simp
next
  case (Suc m)
  have  $(x \oplus y) [\wedge (char R^{\wedge(m+1)}) =$ 
     $(x \oplus y) [\wedge (char R^{\wedge m} * char R)$ 
    by (simp add:mult.commute)
  also have  $... = ((x \oplus y) [\wedge (char R^{\wedge m})) [\wedge char R$ 
    using nat-pow-pow by simp
  also have  $... = (x [\wedge (char R^{\wedge m}) \oplus y [\wedge (char R^{\wedge m})) [\wedge char R$ 
    by (subst Suc, simp)
  also have  $... =$ 
     $(x [\wedge (char R^{\wedge m})) [\wedge char R \oplus (y [\wedge (char R^{\wedge m})) [\wedge char R$ 
    by (subst freshmans-dream[OF assms(1), symmetric], simp-all)

```

**also have** ... =  
 $x \ulcorner (char R^m * char R) \oplus y \ulcorner (char R^m * char R)$   
**by** (*simp add:nat-pow-pow*)  
**also have** ... =  $x \ulcorner (char R^{Suc m}) \oplus y \ulcorner (char R^{Suc m})$   
**by** (*simp add:mult.commute*)  
**finally show** ?*case* **by** *simp*  
**qed**

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**  $m = char R ^ k$

**shows** *ring-hom-cring*  $R R (\lambda x. x \ulcorner m)$

**proof** –

**have**  $a:(x \otimes y) \ulcorner m = x \ulcorner m \otimes y \ulcorner m$

**if**  $b:x \in carrier R y \in carrier R$  **for**  $x y$

**using**  $b$  *nat-pow-distrib* **by** *simp*

**have**  $b:(x \oplus y) \ulcorner m = x \ulcorner m \oplus y \ulcorner m$

**if**  $b:x \in carrier R y \in carrier R$  **for**  $x y$

**unfolding** *assms(2)* *freshmans-dream-ext*[*OF assms(1)*  $b$ ]

**by** *simp*

**have** *ring-hom-ring*  $R R (\lambda x. x \ulcorner m)$

**by** (*intro ring-hom-ringI a b ring-axioms, simp-all*)

**thus** ?*thesis*

**using** *RingHom.ring-hom-cringI is-cring* **by** *blast*

**qed**

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

**assumes**  $char R > 0$

**shows** *subfield* (*char-subring*  $R$ )  $R$

**proof** –

**interpret**  $d:domain R (\ulcorner carrier := char-subring R \urcorner)$

**using** *char-ring-is-subdomain subdomain-is-domain* **by** *simp*

**have** *finite* (*char-subring*  $R$ )

**using** *char-def assms* **by** (*metis card-ge-0-finite*)

**hence** *Units* ( $R (\ulcorner carrier := char-subring R \urcorner)$ )

= *char-subring*  $R - \{0\}$

**using** *d.finite-domain-units* **by** *simp*

**thus** ?*thesis*

**using** *subfieldI*[*OF char-ring-is-subcring*] **by** *simp*

**qed**

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

```

fixes A :: 'a set
shows card {xs. set xs  $\subseteq$  A  $\wedge$  length xs = n} = card A  $\wedge$  n
proof (cases finite A)
  case True
  then show ?thesis using card-lists-length-eq by auto
next
  case False
  hence inf-A: infinite A by simp
  show ?thesis
  proof (cases n = 0)
    case True
    hence card {xs. set xs  $\subseteq$  A  $\wedge$  length xs = n} = card {([] :: 'a list)}
      by (intro arg-cong[where f=card], auto simp add:set-eq-iff)
    also have ... = 1 by simp
    also have ... = card A  $\wedge$  n using True inf-A by simp
    finally show ?thesis by simp
  next
  case False
  hence inj (replicate n)
    by (meson inj-onI replicate-eq-replicate)
  hence inj-on (replicate n) A using inj-on-subset
    by (metis subset-UNIV)
  hence infinite (replicate n ' A)
    using inf-A finite-image-iff by auto
  moreover have
    replicate n ' A  $\subseteq$  {xs. set xs  $\subseteq$  A  $\wedge$  length xs = n}
    by (intro image-subsetI, auto)
  ultimately have infinite {xs. set xs  $\subseteq$  A  $\wedge$  length xs = n}
    using infinite-super by auto
  hence card {xs. set xs  $\subseteq$  A  $\wedge$  length xs = n} = 0 by simp
  then show ?thesis using inf-A False by simp
qed
qed

```

**lemma** (in ring) card-span:

```

assumes subfield K R
assumes independent K w
assumes set w  $\subseteq$  carrier R
shows card (Span K w) = card K  $\wedge$  (length w)
proof -
  define A where A = {x. set x  $\subseteq$  K  $\wedge$  length x = length w}
  define f where f = ( $\lambda$ x. combine x w)

  have x  $\in$  f ' A if a:x  $\in$  Span K w for x
  proof -
    obtain y where y  $\in$  A x = f y
    unfolding A-def f-def
    using unique-decomposition[OF assms(1,2) a] by auto
    thus ?thesis by simp
  qed

```

**qed**  
**moreover have**  $f x \in \text{Span } K w$  **if**  $a: x \in A$  **for**  $x$   
   **using** *Span-eq-combine-set*[*OF assms(1,3)*]  $a$   
   **unfolding** *A-def f-def* **by** *auto*  
**ultimately have**  $b:\text{Span } K w = f \text{ ' } A$  **by** *auto*

**have** *False* **if**  $a: x \in A y \in A f x = f y x \neq y$  **for**  $x y$   
**proof** –  
   **have**  $f x \in \text{Span } K w$  **using**  $b a$  **by** *simp*  
   **thus** *False*  
     **using** *a unique-decomposition*[*OF assms(1,2)*]  
     **unfolding** *f-def A-def* **by** *blast*

**qed**  
**hence** *f-inj: inj-on f A*  
   **unfolding** *inj-on-def* **by** *auto*

**have**  $\text{card } (\text{Span } K w) = \text{card } (f \text{ ' } A)$  **using**  $b$  **by** *simp*  
**also have**  $\dots = \text{card } A$  **by** (*intro card-image f-inj*)  
**also have**  $\dots = \text{card } K^{\text{length } w}$   
   **unfolding** *A-def* **by** (*intro card-lists-length-eq'*)  
**finally show** *?thesis* **by** *simp*

**qed**

**lemma** (**in** *ring*) *finite-carr-imp-char-ge-0*:  
   **assumes** *finite (carrier R)*  
   **shows**  $\text{char } R > 0$

**proof** –  
   **have**  $\text{char-subring } R \subseteq \text{carrier } R$   
     **using** *int-embed-closed* **by** *auto*  
   **hence** *finite (char-subring R)*  
     **using** *finite-subset assms* **by** *auto*  
   **hence**  $\text{card } (\text{char-subring } R) > 0$   
     **using** *card-range-greater-zero* **by** *simp*  
   **thus**  $\text{char } R > 0$   
     **unfolding** *char-def* **by** *simp*

**qed**

**lemma** (**in** *ring*) *char-consistent*:  
   **assumes** *subring H R*  
   **shows**  $\text{char } (R \upharpoonright \text{carrier } := H) = \text{char } R$

**proof** –  
   **show** *?thesis*  
     **using** *int-embed-consistent*[*OF assms(1)*]  
     **unfolding** *char-def* **by** *simp*

**qed**

**lemma** (**in** *ring-hom-ring*) *char-consistent*:  
   **assumes** *inj-on h (carrier R)*  
   **shows**  $\text{char } R = \text{char } S$

**proof** –

**have**  $a:h$  ( $\text{int-embed } R$  ( $\text{int } n$ )) =  $\text{int-embed } S$  ( $\text{int } n$ ) **for**  $n$   
**using**  $R.\text{int-embed-range}$ [ $OF$   $R.\text{carrier-is-subring}$ ]  
**using**  $R.\text{int-embed-range}$ [ $OF$   $R.\text{carrier-is-subring}$ ]  
**using**  $S.\text{int-embed-one}$   $R.\text{int-embed-one}$   
**using**  $S.\text{int-embed-zero}$   $R.\text{int-embed-zero}$   
**using**  $S.\text{int-embed-add}$   $R.\text{int-embed-add}$   
**by** ( $\text{induction } n, \text{ simp-all}$ )

**have**  $b:h$  ( $\text{int-embed } R$  ( $-(\text{int } n)$ )) =  $\text{int-embed } S$  ( $-(\text{int } n)$ ) **for**  $n$   
**using**  $R.\text{int-embed-range}$ [ $OF$   $R.\text{carrier-is-subring}$ ]  
**using**  $S.\text{int-embed-range}$ [ $OF$   $S.\text{carrier-is-subring}$ ]  $a$   
**by** ( $\text{simp add:}R.\text{int-embed-inv } S.\text{int-embed-inv}$ )

**have**  $c:h$  ( $\text{int-embed } R$   $n$ ) =  $\text{int-embed } S$   $n$  **for**  $n$   
**proof** ( $\text{cases } n \geq 0$ )  
**case**  $True$   
**then obtain**  $m$  **where**  $n = \text{int } m$   
**using**  $\text{nonneg-int-cases}$  **by**  $\text{auto}$   
**then show**  $?thesis$   
**by** ( $\text{simp add:}a$ )  
**next**  
**case**  $False$   
**hence**  $n \leq 0$  **by**  $\text{simp}$   
**then obtain**  $m$  **where**  $n = -\text{int } m$   
**using**  $\text{nonpos-int-cases}$  **by**  $\text{auto}$   
**then show**  $?thesis$  **by** ( $\text{simp add:}b$ )  
**qed**

**have**  $\text{char } S = \text{card } (h \text{ ' char-subring } R)$   
**unfolding**  $\text{char-def image-image } c$  **by**  $\text{simp}$   
**also have**  $\dots = \text{card } (\text{char-subring } R)$   
**using**  $R.\text{int-embed-range}$ [ $OF$   $R.\text{carrier-is-subring}$ ]  
**by** ( $\text{intro card-image inj-on-subset}[OF \text{assms}(1)]$ )  $\text{auto}$   
**also have**  $\dots = \text{char } R$  **unfolding**  $\text{char-def}$  **by**  $\text{simp}$   
**finally show**  $?thesis$   
**by**  $\text{simp}$   
**qed**

**definition**  $\text{char-iso} :: - \Rightarrow \text{int set} \Rightarrow 'a$   
**where**  $\text{char-iso } R \ x = \text{the-elem } (\text{int-embed } R \text{ ' } x)$

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

**lemma** (**in ring**)  $\text{char-iso}$ :  $\text{char-iso } R \in$   
 $\text{ring-iso } (ZFact (\text{char } R)) (R \setminus \text{carrier} := \text{char-subring } R)$

**proof** –

**interpret**  $h$ :  $\text{ring-hom-ring int-ring } R \text{ int-embed } R$   
**using**  $\text{int-embed-ring-hom}$  **by**  $\text{simp}$

```

have a-kernel  $Z R$  (int-embed  $R$ ) = {x. int-embed  $R$  x =  $\mathbf{0}$ }
  unfolding a-kernel-def kernel-def by simp
also have ... = {x. char  $R$  dvd x}
  using embed-char-eq-0-iff by simp
also have ... =  $PIdl_Z$  (int (char  $R$ ))
  unfolding cgenideal-def by auto
also have ... =  $Idl_Z$  {int (char  $R$ )}
  using int.cgenideal-eq-genideal by simp
finally have a:a-kernel  $Z R$  (int-embed  $R$ ) =  $Idl_Z$  {int (char  $R$ )}
  by simp
show ?thesis
  unfolding char-iso-def ZFact-def a[symmetric]
  by (intro h.FactRing-iso-set-aux)
qed

```

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*:

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

**proof** –

```

have a:char  $R > 0$ 
  using finite-carr-imp-char-ge-0[OF finite-carrier]
  by simp
let ?CR = char-subring  $R$ 

```

**obtain** *v* **where** *v-def*: *set*  $v = \text{carrier } R$

**using** *finite-carrier finite-list* **by** *auto*

**hence** *b*:*set*  $v \subseteq \text{carrier } R$  **by** *auto*

**have** *carrier*  $R = \text{set } v$  **using** *v-def* **by** *simp*

**also have** ...  $\subseteq \text{Span } ?CR v$

**using** *Span-base-incl*[*OF char-ring-is-subfield*[*OF a*] *b*] **by** *simp*

**finally have** *carrier*  $R \subseteq \text{Span } ?CR v$  **by** *simp*

**moreover have**  $\text{Span } ?CR v \subseteq \text{carrier } R$

**using** *int-embed-closed v-def* **by** (*intro Span-in-carrier, auto*)

**ultimately have** *Span-v*:  $\text{Span } ?CR v = \text{carrier } R$  **by** *simp*

**obtain** *w* **where** *w-def*:

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

*independent*  $?CR w$

$\text{Span } ?CR v = \text{Span } ?CR w$

**using** *b filter-base*[*OF char-ring-is-subfield*[*OF a*]]

**by** *metis*

**have** *Span-w*:  $\text{Span } ?CR w = \text{carrier } R$

**using** *w-def(3) Span-v* **by** *simp*

**hence** *order*  $R = \text{card } (\text{Span } ?CR w)$  **by** (*simp add:order-def*)

```

also have ... = card ?CR ^length w
  by (intro card-span char-ring-is-subfield[OF a] w-def(1,2))
finally have c:
  order R = char R ^length w
  by (simp add:char-def)
have length w > 0
  using finite-field-min-order c by auto
thus ?thesis using c by auto
qed

end

```

## 4 Formal Derivatives

```

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

```

```

definition pderiv (‹pderiv›) where
  pderivR x = ring.normalize R (
    map (λi. 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 –
  have coeff f i ∈ set f ∪ {0}
    using coeff-img(3) by auto
  also have ... ⊆ K ∪ {0}
    using assms(2) univ-poly-carrier polynomial-incl by blast
  also have ... ⊆ K
    using subringE[OF assms(1)] by simp
  finally show ?thesis by simp
qed

```

```

lemma pderiv-carr:
  assumes subring K R
  assumes f ∈ carrier (K[X])
  shows pderiv f ∈ carrier (K[X])
proof –
  have int-embed R i ⊗ coeff f i ∈ K for i
    using coeff-range[OF assms] int-embed-range[OF assms(1)]
    using subringE[OF assms(1)] by simp
  hence polynomial K (pderiv f)
    unfolding pderiv-def by (intro normalize-gives-polynomial, auto)

```

```

thus ?thesis
  using univ-poly-carrier by auto
qed

lemma pderiv-coeff:
  assumes subring K R
  assumes  $f \in \text{carrier } (K[X])$ 
  shows  $\text{coeff } (\text{pderiv } f) k = \text{int-embed } R (\text{Suc } k) \otimes \text{coeff } f (\text{Suc } k)$ 
    (is ?lhs = ?rhs)
proof (cases  $k + 1 < \text{length } f$ )
  case True
  define j where  $j = \text{length } f - k - 2$ 
  define d where
     $d = \text{map } (\lambda i. \text{int-embed } R i \otimes \text{coeff } f i) (\text{rev } [1..<\text{length } f])$ 

  have a:  $j+1 < \text{length } f$ 
    using True unfolding j-def by simp
  hence b:  $j < \text{length } [1..<\text{length } f]$ 
    by simp
  have c:  $k < \text{length } d$ 
    unfolding d-def using True by simp
  have d:  $\text{degree } d - k = j$ 
    unfolding d-def j-def by simp
  have e:  $\text{rev } [\text{Suc } 0..<\text{length } f] ! j = \text{length } f - 1 - j$ 
    using b by (subst rev-nth, auto)
  have f:  $\text{length } f - j - 1 = k+1$ 
    unfolding j-def using True by simp

  have  $\text{coeff } (\text{pderiv } f) k = \text{coeff } (\text{normalize } d) k$ 
    unfolding pderiv-def d-def by simp
  also have ... =  $\text{coeff } d k$ 
    using normalize-coeff by simp
  also have ... =  $d ! j$ 
    using c d by (subst coeff-nth, auto)
  also have
    ... =  $\text{int-embed } R (\text{length } f - j - 1) \otimes \text{coeff } f (\text{length } f - j - 1)$ 
    using b e unfolding d-def by simp
  also have ... = ?rhs
    using f by simp
  finally show ?thesis by simp
next
  case False
  hence  $\text{Suc } k \geq \text{length } f$ 
    by simp
  hence a:  $\text{coeff } f (\text{Suc } k) = \mathbf{0}$ 
    using coeff-img by blast
  have b:  $\text{coeff } (\text{pderiv } f) k = \mathbf{0}$ 
    unfolding pderiv-def normalize-coeff[symmetric] using False
    by (intro coeff-length, simp)

```

**show** *?thesis*  
**using** *int-embed-range[OF carrier-is-subring]* **by** (*simp add:a b*)  
**qed**

**lemma** *pderiv-const*:  
**assumes** *degree x = 0*  
**shows** *pderiv x = 0<sub>K[X]</sub>*  
**proof** (*cases length x = 0*)  
**case** *True*  
**then show** *?thesis* **by** (*simp add:univ-poly-zero pderiv-def*)  
**next**  
**case** *False*  
**hence** *length x = 1* **using** *assms* **by** *linarith*  
**then obtain** *y* **where** *x = [y]* **by** (*cases x, auto*)  
**then show** *?thesis* **by** (*simp add:univ-poly-zero pderiv-def*)  
**qed**

**lemma** *pderiv-var*:  
**shows** *pderiv X = 1<sub>K[X]</sub>*  
**unfolding** *var-def pderiv-def*  
**by** (*simp add:univ-poly-one int-embed-def*)

**lemma** *pderiv-zero*:  
**shows** *pderiv 0<sub>K[X]</sub> = 0<sub>K[X]</sub>*  
**unfolding** *pderiv-def univ-poly-zero* **by** *simp*

**lemma** *pderiv-add*:  
**assumes** *subring K R*  
**assumes** [*simp*]: *f ∈ carrier (K[X]) g ∈ carrier (K[X])*  
**shows** *pderiv (f ⊕<sub>K[X]</sub> g) = pderiv f ⊕<sub>K[X]</sub> pderiv g*  
*(is ?lhs = ?rhs)*

**proof** –  
**interpret** *p*: *ring (K[X])*  
**using** *univ-poly-is-ring[OF assms(1)]* **by** *simp*

**let** *?n = (λi. int-embed R i)*

**have** [*simp*]: *?n k ∈ carrier R* **for** *k*  
**using** *int-embed-range[OF carrier-is-subring]* **by** *auto*  
**have** [*simp*]: *coeff f k ∈ carrier R* **if** *f ∈ carrier (K[X])* **for** *k f*  
**using** *coeff-range[OF assms(1)]* **that**  
**using** *subringE(1)[OF assms(1)]* **by** *auto*

**have** *coeff ?lhs i = coeff ?rhs i* **for** *i*

**proof** –  
**have** *coeff ?lhs i = ?n (i+1) ⊗ coeff (f ⊕<sub>K[X]</sub> g) (i+1)*  
**by** (*simp add: pderiv-coeff[OF assms(1)]*)  
**also have** *... = ?n (i+1) ⊗ (coeff f (i+1) ⊕ coeff g (i+1))*

```

    by (subst coeff-add[OF assms], simp)
  also have ... = ?n (i+1)  $\otimes$  coeff f (i+1)
     $\oplus$  int-embed R (i+1)  $\otimes$  coeff g (i+1)
    by (subst r-distr, simp-all)
  also have ... = coeff (pderiv f) i  $\oplus$  coeff (pderiv g) i
    by (simp add: pderiv-coeff[OF assms(1)])
  also have ... = coeff (pderiv f  $\oplus_K$  [X] pderiv g) i
    using pderiv-carr[OF assms(1)]
    by (subst coeff-add[OF assms(1)], auto)
  finally show ?thesis by simp
qed
hence coeff ?lhs = coeff ?rhs by auto
thus ?lhs = ?rhs
  using pderiv-carr[OF assms(1)]
  by (subst coeff-iff-polynomial-cond[where K=K])
    (simp-all add:univ-poly-carrier)+
qed

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 -
  interpret p: cring (K[X])
    using univ-poly-is-cring[OF assms(1)] by simp

  have pderiv ( $\ominus_{K[X]}$  f) = pderiv ( $\ominus_{K[X]}$  f)  $\oplus_{K[X]}$   $\mathbf{0}_{K[X]}$ 
    using pderiv-carr[OF assms(1)]
    by (subst p.r-zero, simp-all)
  also have ... = pderiv ( $\ominus_{K[X]}$  f)  $\oplus_{K[X]}$  (pderiv f  $\ominus_{K[X]}$  pderiv f)
    using pderiv-carr[OF assms(1)] by simp
  also have ... = pderiv ( $\ominus_{K[X]}$  f)  $\oplus_{K[X]}$  pderiv f  $\ominus_{K[X]}$  pderiv f
    using pderiv-carr[OF assms(1)]
    unfolding a-minus-def by (simp add:p.a-assoc)
  also have ... = pderiv ( $\ominus_{K[X]}$  f  $\oplus_{K[X]}$  f)  $\ominus_{K[X]}$  pderiv f
    by (subst pderiv-add[OF assms(1)], simp-all)
  also have ... = pderiv  $\mathbf{0}_{K[X]}$   $\ominus_{K[X]}$  pderiv f
    by (subst p.l-neg, simp-all)
  also have ... =  $\mathbf{0}_{K[X]}$   $\ominus_{K[X]}$  pderiv f
    by (subst pderiv-zero, simp)
  also have ... =  $\ominus_{K[X]}$  pderiv f
    unfolding a-minus-def using pderiv-carr[OF assms(1)]
    by (subst p.l-zero, simp-all)
  finally show pderiv ( $\ominus_{K[X]}$  f) =  $\ominus_{K[X]}$  pderiv f
    by simp
qed

```

**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** –

**have**  $a:\text{set } f \subseteq \text{carrier } R$

**using** *assms(1,2)* *univ-poly-carrier*

**using** *subringE(1)[OF assms(1)]* *polynomial-incl* **by** *blast*

**have**  $b:\text{set } g \subseteq \text{carrier } R$

**using** *assms(1,3)* *univ-poly-carrier*

**using** *subringE(1)[OF assms(1)]* *polynomial-incl* **by** *blast*

**show** *?thesis*

**unfolding** *univ-poly-mult poly-mult-coeff[OF a b]* **by** *simp*

**qed**

**lemma** *pderiv-mult*:

**assumes** *subring*  $K R$

**assumes** [*simp*]:  $f \in \text{carrier } (K[X])$   $g \in \text{carrier } (K[X])$

**shows**  $\text{pderiv } (f \otimes_{K[X]} g) =$   
 $\text{pderiv } f \otimes_{K[X]} g \oplus_{K[X]} f \otimes_{K[X]} \text{pderiv } g$   
*(is ?lhs = ?rhs)*

**proof** –

**interpret**  $p:\text{cring } (K[X])$

**using** *univ-poly-is-cring[OF assms(1)]* **by** *simp*

**let**  $?n = (\lambda i. \text{int-embed } R i)$

**have**  $a[\text{simp}]:?n k \in \text{carrier } R$  **for**  $k$

**using** *int-embed-range[OF carrier-is-subring]* **by** *auto*

**have**  $b[\text{simp}]:\text{coeff } f k \in \text{carrier } R$  **if**  $f \in \text{carrier } (K[X])$  **for**  $k f$

**using** *coeff-range[OF assms(1)]*

**using** *subringE(1)[OF assms(1)]* **that** **by** *auto*

**have**  $\text{coeff } ?lhs i = \text{coeff } ?rhs i$  **for**  $i$

**proof** –

**have**  $\text{coeff } ?lhs i = ?n (i+1) \otimes \text{coeff } (f \otimes_{K[X]} g) (i+1)$

**using** *assms(2,3)* **by** (*simp add: pderiv-coeff[OF assms(1)]*)

**also have**  $\dots = ?n (i+1) \otimes$   
 $(\bigoplus k \in \{..i+1\}. \text{coeff } f k \otimes (\text{coeff } g (i + 1 - k)))$

**by** (*subst coeff-mult[OF assms], simp*)

**also have**  $\dots =$   
 $(\bigoplus k \in \{..i+1\}. ?n (i+1) \otimes (\text{coeff } f k \otimes \text{coeff } g (i + 1 - k)))$

**by** (*intro finsum-rdistr, simp-all add:Pi-def*)

**also have**  $\dots =$   
 $(\bigoplus k \in \{..i+1\}. ?n k \otimes (\text{coeff } f k \otimes \text{coeff } g (i + 1 - k)) \oplus$   
 $?n (i+1-k) \otimes (\text{coeff } f k \otimes \text{coeff } g (i + 1 - k)))$

**using** *int-embed-add[symmetric]* *of-nat-diff*

**by** (*intro finsum-cong'*)

$(\text{simp-all add:l-distr[symmetric] of-nat-diff})$   
**also have** ... =  
 $(\bigoplus k \in \{..i+1\}. ?n k \otimes \text{coeff } f k \otimes \text{coeff } g (i + 1 - k) \oplus$   
 $\text{coeff } f k \otimes (?n (i+1-k) \otimes \text{coeff } g (i + 1 - k)))$   
**using** *Pi-def a b m-assoc m-comm*  
**by**  $(\text{intro finsum-cong}' \text{ arg-cong2[where } f=(\oplus)], \text{ simp-all})$   
**also have** ... =  
 $(\bigoplus k \in \{..i+1\}. ?n k \otimes \text{coeff } f k \otimes \text{coeff } g (i+1-k)) \oplus$   
 $(\bigoplus k \in \{..i+1\}. \text{coeff } f k \otimes (?n (i+1-k) \otimes \text{coeff } g (i+1-k)))$   
**by**  $(\text{subst finsum-addf[symmetric], simp-all add:Pi-def})$   
**also have** ... =  
 $(\bigoplus k \in \text{insert } 0 \{1..i+1\}. ?n k \otimes \text{coeff } f k \otimes \text{coeff } g (i+1-k)) \oplus$   
 $(\bigoplus k \in \text{insert } (i+1) \{..i\}. \text{coeff } f k \otimes (?n (i+1-k) \otimes \text{coeff } g$   
 $(i+1-k)))$   
**using** *subringE(1)[OF assms(1)]*  
**by**  $(\text{intro arg-cong2[where } f=(\oplus)] \text{ finsum-cong}'$   
 $(\text{auto simp:set-eq-iff}))$   
**also have** ... =  
 $(\bigoplus k \in \{1..i+1\}. ?n k \otimes \text{coeff } f k \otimes \text{coeff } g (i+1-k)) \oplus$   
 $(\bigoplus k \in \{..i\}. \text{coeff } f k \otimes (?n (i+1-k) \otimes \text{coeff } g (i+1-k)))$   
**by**  $(\text{subst } (1 \ 2) \text{ finsum-insert, auto simp add:int-embed-zero})$   
**also have** ... =  
 $(\bigoplus k \in \text{Suc } \{..i\}. ?n k \otimes \text{coeff } f (k) \otimes \text{coeff } g (i+1-k)) \oplus$   
 $(\bigoplus k \in \{..i\}. \text{coeff } f k \otimes (?n (i+1-k) \otimes \text{coeff } g (i+1-k)))$   
**by**  $(\text{intro arg-cong2[where } f=(\oplus)] \text{ finsum-cong}'$   
 $(\text{simp-all add:Pi-def atMost-atLeast0}))$   
**also have** ... =  
 $(\bigoplus k \in \{..i\}. ?n (k+1) \otimes \text{coeff } f (k+1) \otimes \text{coeff } g (i-k)) \oplus$   
 $(\bigoplus k \in \{..i\}. \text{coeff } f k \otimes (?n (i+1-k) \otimes \text{coeff } g (i+1-k)))$   
**by**  $(\text{subst finsum-reindex, auto})$   
**also have** ... =  
 $(\bigoplus k \in \{..i\}. \text{coeff } (\text{pderiv } f) k \otimes \text{coeff } g (i-k)) \oplus$   
 $(\bigoplus k \in \{..i\}. \text{coeff } f k \otimes \text{coeff } (\text{pderiv } g) (i-k))$   
**using** *Suc-diff-le*  
**by**  $(\text{subst } (1 \ 2) \text{ pderiv-coeff[OF assms(1)]}$   
 $(\text{auto intro!: finsum-cong}'))$   
**also have** ... =  
 $\text{coeff } (\text{pderiv } f \otimes_{K[X]} g) i \oplus \text{coeff } (f \otimes_{K[X]} \text{pderiv } g) i$   
**using** *pderiv-carr[OF assms(1)]*  
**by**  $(\text{subst } (1 \ 2) \text{ coeff-mult[OF assms(1)], auto})$   
**also have** ... = *coeff ?rhs i*  
**using** *pderiv-carr[OF assms(1)]*  
**by**  $(\text{subst coeff-add[OF assms(1)], auto})$   
**finally show** *?thesis* **by** *simp*  
**qed**  
  
**hence** *coeff ?lhs = coeff ?rhs* **by** *auto*  
**thus** *?lhs = ?rhs*  
**using** *pderiv-carr[OF assms(1)]*

by (*subst coeff-iff-polynomial-cond*[**where**  $K=K$ ])  
 (*simp-all add:univ-poly-carrier*)  
**qed**

**lemma** *pderiv-pow*:  
 assumes  $n > (0 :: nat)$   
 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** –  
 interpret  $p: \text{cring } (K[X])$   
 using *univ-poly-is-cring*[*OF assms(2)*] **by** *simp*

let  $?n = \lambda n. \text{int-embed } (K[X]) n$

have [*simp*]:  $?n i \in \text{carrier } (K[X])$  **for**  $i$   
 using *p.int-embed-range*[*OF p.carrier-is-subring*] **by** *simp*

obtain  $m$  **where** *n-def*:  $n = \text{Suc } m$  **using** *assms(1)* *lessE* **by** *blast*  
 have  $pderiv (f [\wedge]_{K[X]} (m+1)) =$   
    $?n (m+1) \otimes_{K[X]} f [\wedge]_{K[X]} m \otimes_{K[X]} pderiv f$   
**proof** (*induction m*)  
 case 0  
 then show *?case*  
   using *pderiv-carr*[*OF assms(2)*] *assms(3)*  
   using *p.int-embed-one* **by** *simp*

**next**  
 case (*Suc m*)  
 have  $pderiv (f [\wedge]_{K[X]} (\text{Suc } m + 1)) =$   
    $pderiv (f [\wedge]_{K[X]} (m+1) \otimes_{K[X]} f)$   
   **by** *simp*  
 also have ... =  
    $pderiv (f [\wedge]_{K[X]} (m+1)) \otimes_{K[X]} f \oplus_{K[X]}$   
    $f [\wedge]_{K[X]} (m+1) \otimes_{K[X]} pderiv f$   
   **using** *assms(3)* **by** (*subst pderiv-mult*[*OF assms(2)*], *auto*)  
 also have ... =  
    $(?n (m+1) \otimes_{K[X]} f [\wedge]_{K[X]} m \otimes_{K[X]} pderiv f) \otimes_{K[X]} f$   
    $\oplus_{K[X]} f [\wedge]_{K[X]} (m+1) \otimes_{K[X]} pderiv f$   
   **by** (*subst Suc(1)*, *simp*)  
 also have  
   ... =  $?n (m+1) \otimes_{K[X]} (f [\wedge]_{K[X]} (m+1) \otimes_{K[X]} pderiv f)$   
    $\oplus_{K[X]} \mathbf{1}_{K[X]} \otimes_{K[X]} (f [\wedge]_{K[X]} (m+1) \otimes_{K[X]} pderiv f)$   
   **using** *assms(3)* *pderiv-carr*[*OF assms(2)*]  
   **apply** (*intro arg-cong2*[**where**  $f=(\oplus_{K[X]})$ ])  
   **apply** (*simp add:p.m-assoc*)

```

    apply (simp add:p.m-comm)
  by simp
also have
  ... = (?n (m+1) ⊕K[X] 1K[X]) ⊗K[X]
  (f [↑]K[X] (m+1) ⊗K[X] pderiv f)
  using assms(3) pderiv-carr[OF assms(2)]
  by (subst p.l-distr[symmetric], simp-all)
also have ... =
  (1K[X] ⊕K[X] ?n (m+1)) ⊗K[X]
  (f [↑]K[X] (m+1) ⊗K[X] pderiv f)
  using assms(3) pderiv-carr[OF assms(2)]
  by (subst p.a-comm, simp-all)
also have ... = ?n (1+ Suc m)
  ⊗K[X] f [↑]K[X] (Suc m) ⊗K[X] pderiv f
  using assms(3) pderiv-carr[OF assms(2)] of-nat-add
  apply (subst (2) of-nat-add, subst p.int-embed-add)
  by (simp add:p.m-assoc p.int-embed-one)
finally show ?case by simp
qed
thus ?thesis using n-def by auto
qed

```

```

lemma pderiv-var-pow:
  assumes n > (0::nat)
  assumes subring K R
  shows pderiv (X [↑]K[X] n) =
    int-embed (K[X]) n ⊗K[X] X [↑]K[X] (n-1)
proof -
  interpret p: cring (K[X])
  using univ-poly-is-cring[OF assms(2)] by simp

  have [simp]: int-embed (K[X]) i ∈ carrier (K[X]) for i
  using p.int-embed-range[OF p.carrier-is-subring] by simp

  show ?thesis
  using var-closed[OF assms(2)]
  using pderiv-var[where K=K] pderiv-carr[OF assms(2)]
  by (subst pderiv-pow[OF assms(1,2)], simp-all)
qed

```

```

lemma int-embed-consistent-with-poly-of-const:
  assumes subring K R
  shows int-embed (K[X]) m = poly-of-const (int-embed R m)
proof -
  define K' where K' = R (| carrier := K |)
  interpret p: cring (K[X])
  using univ-poly-is-cring[OF assms] by simp
  interpret d: domain K'

```

```

    unfolding K'-def
    using assms(1) subdomainI' subdomain-is-domain by simp
interpret h: ring-hom-ring K' K[X] poly-of-const
    unfolding K'-def
    using canonical-embedding-ring-hom[OF assms(1)] by simp

define n where n=nat (abs m)

have a1: int-embed (K[X]) (int n) = poly-of-const (int-embed K' n)
proof (induction n)
  case 0
  then show ?case by (simp add:d.int-embed-zero p.int-embed-zero)
next
  case (Suc n)
  then show ?case
    using d.int-embed-closed d.int-embed-add d.int-embed-one
    by (simp add:p.int-embed-add p.int-embed-one)
qed
also have ... = poly-of-const (int-embed R n)
  unfolding K'-def using int-embed-consistent[OF assms] by simp
finally have a:
  int-embed (K[X]) (int n) = poly-of-const (int-embed R (int n))
  by simp

have int-embed (K[X]) (-(int n)) =
  poly-of-const (int-embed K' (-(int n)))
  using d.int-embed-closed a1 by (simp add: p.int-embed-inv d.int-embed-inv)
also have ... = poly-of-const (int-embed R (-(int n)))
  unfolding K'-def using int-embed-consistent[OF assms] by simp
finally have b:
  int-embed (K[X]) (-(int n)) = poly-of-const (int-embed R (-(int n)))
  by simp

show ?thesis
  using a b n-def by (cases m ≥ 0, simp, simp)
qed

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**  $\text{finprod } R f A = \text{finprod } (\text{mult-of } R) f A$   
**using** *assms* **by** (*induction A rule:finite-induct, auto*)

**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** –  
**have** *finite*  $\{f. \text{set } f \subseteq K \wedge \text{length } f \leq n + 1\}$  (**is** *finite ?C*)  
**using** *assms(2) finite-lists-length-le* **by** *auto*  
**moreover** **have**  $?B \subseteq ?C$   
**by** (*intro subsetI*)  
*(auto simp:univ-poly-carrier[symmetric] polynomial-def)*  
**ultimately show** *a: finite ?B*  
**using** *finite-subset* **by** *auto*  
**moreover** **have**  $?A \subseteq ?B$   
**by** (*intro subsetI, simp*)  
**ultimately show** *finite ?A*  
**using** *finite-subset* **by** *auto*  
**qed**

**definition** *pmult* ::  $- \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow \text{nat}$  ( $\langle \text{pmult} \rangle$ )  
**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**  $m-i-p \equiv \text{monic-irreducible-poly}$

**locale** *polynomial-ring* = *field* +  
**fixes**  $K$   
**assumes** *polynomial-ring-assms: subfield K R*  
**begin**

**lemma** *K-subring: subring K R*  
**using** *polynomial-ring-assms subfieldE(1)* **by** *auto*

**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) be introduced as a sublocale it has no lasting effect for the field locale itself.

**lemmas**

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

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

by (simp add: carrier-is-subfield univ-poly-is-principal)

**end**

```

context field
begin

interpretation polynomial-ring R carrier R
  using carrier-is-subfield field-axioms
  by (simp add:polynomial-ring-def polynomial-ring-axioms-def)

lemma pdivides-mult-r:
  assumes a ∈ carrier (mult-of P)
  assumes b ∈ carrier (mult-of P)
  assumes c ∈ carrier (mult-of P)
  shows a ⊗P c pdivides b ⊗P c ⟷ a pdivides b
    (is ?lhs ⟷ ?rhs)
proof -
  have a:b ⊗P c ∈ carrier P - {0P}
    using assms p.mult-of.m-closed by force
  have b:a ⊗P c ∈ carrier P
    using assms by simp
  have c:b ∈ carrier P - {0P}
    using assms p.mult-of.m-closed by force
  have d:a ∈ carrier P using assms by simp
  have ?lhs ⟷ a ⊗P c dividesmult-of P b ⊗P c
    unfolding pdivides-def using p.divides-imp-divides-mult a b
    by (meson divides-mult-imp-divides)
  also have ... ⟷ a dividesmult-of P b
    using p.mult-of.divides-mult-r[OF assms] by simp
  also have ... ⟷ ?rhs
    unfolding pdivides-def using p.divides-imp-divides-mult c d
    by (meson divides-mult-imp-divides)
  finally show ?thesis by simp
qed

lemma lead-coeff-carr:
  assumes x ∈ carrier (mult-of P)
  shows lead-coeff x ∈ carrier R - {0}
proof (cases x)
  case Nil
  then show ?thesis using assms by (simp add:univ-poly-zero)
next
  case (Cons a list)
  hence a: polynomial (carrier R) (a # list)
    using assms univ-poly-carrier by auto
  have lead-coeff x = a
    using Cons by simp
  also have a ∈ carrier R - {0}
    using lead-coeff-not-zero a by simp
  finally show ?thesis by simp
qed

```

**lemma** *lead-coeff-poly-of-const*:  
**assumes**  $r \neq \mathbf{0}$   
**shows**  $\text{lead-coeff } (\text{poly-of-const } r) = r$   
**using** *assms*  
**by** (*simp add:poly-of-const-def*)

**lemma** *lead-coeff-mult*:  
**assumes**  $f \in \text{carrier } (\text{mult-of } P)$   
**assumes**  $g \in \text{carrier } (\text{mult-of } P)$   
**shows**  $\text{lead-coeff } (f \otimes_P g) = \text{lead-coeff } f \otimes \text{lead-coeff } g$   
**unfolding** *univ-poly-mult* **using** *assms*  
**using** *univ-poly-carrier* [where  $R=R$  and  $K=\text{carrier } R$ ]  
**by** (*subst poly-mult-lead-coeff*) (*simp-all add:univ-poly-zero*)

**lemma** *monic-poly-carr*:  
**assumes** *monic-poly*  $R$   $f$   
**shows**  $f \in \text{carrier } P$   
**using** *assms* **unfolding** *monic-poly-def* **by** *simp*

**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*  $R$   $(f \oplus_P g)$   
**proof** (*cases*  $g \neq \mathbf{0}_P$ )  
**case** *True*  
**define**  $n$  **where**  $n = \text{degree } f$   
**have**  $f \in \text{carrier } P - \{\mathbf{0}_P\}$   
**using** *assms(1)* *univ-poly-zero*  
**unfolding** *monic-poly-def* **by** *auto*  
**hence**  $\text{degree } (f \oplus_P g) = \max (\text{degree } f) (\text{degree } g)$   
**using** *assms(2,3)* *True*  
**by** (*subst degree-add-distinct, simp-all*)  
**also have**  $\dots = \text{degree } f$   
**using** *assms(3)* **by** *simp*  
**finally have**  $b: \text{degree } (f \oplus_P g) = n$   
**unfolding** *n-def* **by** *simp*  
**moreover have**  $n > 0$   
**using** *assms(3)* **unfolding** *n-def* **by** *simp*  
**ultimately have**  $\text{degree } (f \oplus_P g) \neq \text{degree } (\mathbf{0})$   
**by** *simp*  
**hence**  $a: f \oplus_P g \neq \mathbf{0}$  **by** *auto*

**have**  $\text{degree } \mathbf{0} = 0$  **by** *simp*  
**also have**  $\dots < \text{degree } f$   
**using** *assms(3)* **by** *simp*  
**finally have**  $\text{degree } f \neq \text{degree } \mathbf{0}$  **by** *simp*  
**hence**  $c: f \neq \mathbf{0}$  **by** *auto*

```

have  $d$ :  $\text{length } g \leq n$ 
  using  $\text{assms}(3)$  unfolding  $n\text{-def}$  by  $\text{simp}$ 

have  $\text{lead-coeff } (f \oplus_P g) = \text{coeff } (f \oplus_P g) \ n$ 
  using  $a \ b$  by  $(\text{cases } f \oplus_P g, \text{auto})$ 
also have  $\dots = \text{coeff } f \ n \oplus \text{coeff } g \ n$ 
  using  $\text{monic-poly-carr } \text{assms}$ 
  by  $(\text{subst } \text{coeff-add}, \text{auto})$ 
also have  $\dots = \text{lead-coeff } f \oplus \text{coeff } g \ n$ 
  using  $c$  unfolding  $n\text{-def}$  by  $(\text{cases } f, \text{auto})$ 
also have  $\dots = \mathbf{1} \oplus \mathbf{0}$ 
  using  $\text{assms}(1)$  unfolding  $\text{monic-poly-def}$ 
  unfolding  $\text{subst } \text{coeff-length}[OF \ d]$  by  $\text{simp}$ 
also have  $\dots = \mathbf{1}$ 
  by  $\text{simp}$ 
finally have  $\text{lead-coeff } (f \oplus_P g) = \mathbf{1}$  by  $\text{simp}$ 
moreover have  $f \oplus_P g \in \text{carrier } P$ 
  using  $\text{monic-poly-carr } \text{assms}$  by  $\text{simp}$ 
ultimately show  $?thesis$ 
  using  $a$  unfolding  $\text{monic-poly-def}$  by  $\text{auto}$ 
next
  case  $\text{False}$ 
  then show  $?thesis$  using  $\text{assms } \text{monic-poly-carr}$  by  $\text{simp}$ 
qed

```

```

lemma  $\text{monic-poly-one}$ :  $\text{monic-poly } R \ \mathbf{1}_P$ 
proof –
  have  $\mathbf{1}_P \in \text{carrier } P$ 
  by  $\text{simp}$ 
  thus  $?thesis$ 
  by  $(\text{simp } \text{add:univ-poly-one } \text{monic-poly-def})$ 
qed

```

```

lemma  $\text{monic-poly-var}$ :  $\text{monic-poly } R \ X$ 
proof –
  have  $X \in \text{carrier } P$ 
  using  $\text{var-closed}$  by  $\text{simp}$ 
  thus  $?thesis$ 
  by  $(\text{simp } \text{add:var-def } \text{monic-poly-def})$ 
qed

```

```

lemma  $\text{monic-poly-carr-2}$ :
  assumes  $\text{monic-poly } R \ f$ 
  shows  $f \in \text{carrier } (\text{mult-of } P)$ 
  using  $\text{assms}$  unfolding  $\text{monic-poly-def}$ 
  by  $(\text{simp } \text{add:univ-poly-zero})$ 

```

```

lemma  $\text{monic-poly-mult}$ :
  assumes  $\text{monic-poly } R \ f$ 

```

```

assumes monic-poly  $R$   $g$ 
shows monic-poly  $R$   $(f \otimes_P g)$ 
proof –
  have lead-coeff  $(f \otimes_P g) = \text{lead-coeff } f \otimes_R \text{lead-coeff } g$ 
    using assms monic-poly-carr-2
    by (subst lead-coeff-mult) auto
  also have  $\dots = \mathbf{1}$ 
    using assms unfolding monic-poly-def by simp
  finally have lead-coeff  $(f \otimes_P g) = \mathbf{1}_R$  by simp
  moreover have  $(f \otimes_P g) \in \text{carrier } (\text{mult-of } P)$ 
    using monic-poly-carr-2 assms by blast
  ultimately show ?thesis
    by (simp add:monic-poly-def univ-poly-zero)
qed

```

```

lemma monic-poly-pow:
  assumes monic-poly  $R$   $f$ 
  shows monic-poly  $R$   $(f [\wedge]_P (n::\text{nat}))$ 
  using assms monic-poly-one monic-poly-mult
  by (induction n, auto)

```

```

lemma monic-poly-prod:
  assumes finite  $A$ 
  assumes  $\bigwedge x. x \in A \implies \text{monic-poly } R (f x)$ 
  shows monic-poly  $R$   $(\text{finprod } P f A)$ 
  using assms
proof (induction A rule:finite-induct)
  case empty
  then show ?case by (simp add:monic-poly-one)
next
  case (insert x F)
  have  $a: f \in F \rightarrow \text{carrier } P$ 
    using insert monic-poly-carr by simp
  have  $b: f x \in \text{carrier } P$ 
    using insert monic-poly-carr by simp
  have monic-poly  $R$   $(f x \otimes_P \text{finprod } P f F)$ 
    using insert by (intro monic-poly-mult) auto
  thus ?case
    using insert a b by (subst p.finprod-insert, auto)
qed

```

```

lemma monic-poly-not-assoc:
  assumes monic-poly  $R$   $f$ 
  assumes monic-poly  $R$   $g$ 
  assumes  $f \sim_{(\text{mult-of } P)} g$ 
  shows  $f = g$ 
proof –
  obtain  $u$  where u-def:  $f = g \otimes_P u$   $u \in \text{Units } (\text{mult-of } P)$ 
    using p.mult-of.associatedD2 assms monic-poly-carr-2

```

by *blast*

hence  $u \in \text{Units } P$  by *simp*

then obtain  $v$  where  $v\text{-def}: u = [v] v \neq \mathbf{0}_R v \in \text{carrier } R$   
using *univ-poly-carrier-units* by *auto*

have  $\mathbf{1} = \text{lead-coeff } f$

using *assms(1)* by (*simp add:monic-poly-def*)

also have  $\dots = \text{lead-coeff } (g \otimes_P u)$

by (*simp add:u-def*)

also have  $\dots = \text{lead-coeff } g \otimes \text{lead-coeff } u$

using *assms(2) monic-poly-carr-2 v-def u-def(2)*

by (*subst lead-coeff-mult, auto simp add:univ-poly-zero*)

also have  $\dots = \text{lead-coeff } g \otimes v$

using *v-def* by *simp*

also have  $\dots = v$

using *assms(2) v-def(3)* by (*simp add:monic-poly-def*)

finally have  $\mathbf{1} = v$  by *simp*

hence  $u = \mathbf{1}_P$

using *v-def* by (*simp add:univ-poly-one*)

thus  $f = g$

using *u-def assms monic-poly-carr* by *simp*

qed

lemma *monic-poly-span*:

assumes  $x \in \text{carrier } (\text{mult-of } P)$  *irreducible* (*mult-of } P)  $x$*

shows  $\exists y. \text{monic-irreducible-poly } R y \wedge x \sim_{(\text{mult-of } P)} y$

proof –

define  $z$  where  $z = \text{poly-of-const } (\text{inv } (\text{lead-coeff } x))$

define  $y$  where  $y = x \otimes_P z$

have  $x\text{-carr}: x \in \text{carrier } (\text{mult-of } P)$  using *assms* by *simp*

hence  $lx\text{-ne-0}: \text{lead-coeff } x \neq \mathbf{0}$

and  $lx\text{-unit}: \text{lead-coeff } x \in \text{Units } R$

using *lead-coeff-carr[OF x-carr]* by (*auto simp add:field-Units*)

have  $lx\text{-inv-ne-0}: \text{inv } (\text{lead-coeff } x) \neq \mathbf{0}$

using *lx-unit*

by (*metis Units-closed Units-r-inv r-null zero-not-one*)

have  $lx\text{-inv-carr}: \text{inv } (\text{lead-coeff } x) \in \text{carrier } R$

using *lx-unit* by *simp*

have  $z \in \text{carrier } P$

using *lx-inv-carr poly-of-const-over-carrier*

*unfolding z-def* by *auto*

moreover have  $z \neq \mathbf{0}_P$

using *lx-inv-ne-0*

by (*simp add:z-def poly-of-const-def univ-poly-zero*)

ultimately have  $z\text{-carr}: z \in \text{carrier } (\text{mult-of } P)$  by *simp*

**have**  $z\text{-unit}$ :  $z \in \text{Units} (\text{mult-of } P)$   
**using**  $lx\text{-inv-ne-0 } lx\text{-inv-carr}$   
**by** ( $\text{simp add:univ-poly-carrier-units } z\text{-def poly-of-const-def}$ )  
**have**  $y\text{-exp}$ :  $y = x \otimes_{(\text{mult-of } P)} z$   
**by** ( $\text{simp add:y-def}$ )  
**hence**  $y\text{-carr}$ :  $y \in \text{carrier} (\text{mult-of } P)$   
**using**  $x\text{-carr } z\text{-carr } p.\text{mult-of.m-closed}$  **by**  $\text{simp}$

**have**  $\text{irreducible} (\text{mult-of } P) y$   
**unfolding**  $y\text{-def}$  **using**  $\text{assms } z\text{-unit } z\text{-carr}$   
**by** ( $\text{intro } p.\text{mult-of.irreducible-prod-rI, auto}$ )  
**moreover have**  $\text{lead-coeff } y = \mathbf{1}_R$   
**unfolding**  $y\text{-def}$  **using**  $x\text{-carr } z\text{-carr } lx\text{-inv-ne-0 } lx\text{-unit}$   
**by** ( $\text{simp add: lead-coeff-mult } z\text{-def lead-coeff-poly-of-const}$ )  
**hence**  $\text{monic-poly } R y$   
**using**  $y\text{-carr}$  **unfolding**  $\text{monic-poly-def}$   
**by** ( $\text{simp add:univ-poly-zero}$ )  
**ultimately have**  $\text{monic-irreducible-poly } R y$   
**using**  $p.\text{irreducible-mult-imp-irreducible } y\text{-carr}$   
**by** ( $\text{simp add:monic-irreducible-poly-def ring-irreducible-def}$ )  
**moreover have**  $y \sim_{(\text{mult-of } P)} x$   
**by** ( $\text{intro } p.\text{mult-of.associatedI2}[OF z\text{-unit}] y\text{-def } x\text{-carr}$ )  
**hence**  $x \sim_{(\text{mult-of } P)} y$   
**using**  $x\text{-carr } y\text{-carr}$  **by** ( $\text{simp add:p.mult-of.associated-sym}$ )  
**ultimately show**  $?thesis$  **by**  $\text{auto}$

qed

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

**proof** –

**have**  $sp\text{-1}$ :  
 $?S \subseteq \{x \in \text{carrier} (\text{mult-of } P). \text{irreducible} (\text{mult-of } P) x\}$   
**unfolding**  $\text{monic-irreducible-poly-def ring-irreducible-def}$   
**using**  $\text{monic-poly-carr}$   
**by** ( $\text{intro subsetI, simp add: } p.\text{irreducible-imp-irreducible-mult}$ )  
**have**  $sp\text{-2}$ :  $x = y$   
**if**  $x \in ?S y \in ?S x \sim_{(\text{mult-of } P)} y$  **for**  $x y$   
**using**  $\text{that monic-poly-not-assoc}$   
**by** ( $\text{simp add:monic-irreducible-poly-def}$ )

**have**  $sp\text{-3}$ :  $\exists y \in ?S. x \sim_{(\text{mult-of } P)} y$   
**if**  $x \in \text{carrier} (\text{mult-of } P) \text{irreducible} (\text{mult-of } P) x$  **for**  $x$   
**using**  $\text{that monic-poly-span}$  **by**  $\text{simp}$

**thus**  $?thesis$  **using**  $sp\text{-1 } sp\text{-2 } sp\text{-3}$   
**unfolding**  $\text{canonical-irreducibles-def}$  **by**  $\text{simp}$

qed

**lemma**  
**assumes** *monic-poly R a*  
**shows** *factor-monic-poly*:  
 $a = (\bigotimes_{p \in d} p) \in \{d. \text{monic-irreducible-poly } R \ d \wedge \text{pmult } d \ a > 0\}$ .  
 $d \ [\bigwedge]_P \text{pmult } d \ a$  (**is** *?lhs = ?rhs*)  
**and** *factor-monic-poly-fin*:  
 $\text{finite } \{d. \text{monic-irreducible-poly } R \ d \wedge \text{pmult } d \ a > 0\}$   
**proof** –  
**let** *?S* =  $\{d. \text{monic-irreducible-poly } R \ d\}$   
**let** *?T* =  $\{d. \text{monic-irreducible-poly } R \ d \wedge \text{pmult } d \ a > 0\}$   
**let** *?mip* = *monic-irreducible-poly R*  
  
**have** *sp-4*:  $a \in \text{carrier } (\text{mult-of } P)$   
**using** *assms monic-poly-carr-2*  
**unfolding** *monic-irreducible-poly-def* **by** *simp*  
  
**have** *b-1*:  $x \in \text{carrier } (\text{mult-of } P)$  **if** *?mip x* **for**  $x$   
**using** *that monic-poly-carr-2*  
**unfolding** *monic-irreducible-poly-def* **by** *simp*  
**have** *b-2*: *irreducible (mult-of P) x* **if** *?mip x* **for**  $x$   
**using** *that*  
**unfolding** *monic-irreducible-poly-def ring-irreducible-def*  
**by** (*simp add: monic-poly-carr p.irreducible-imp-irreducible-mult*)  
**have** *b-3*:  $x \in \text{carrier } P$  **if** *?mip x* **for**  $x$   
**using** *that monic-poly-carr*  
**unfolding** *monic-irreducible-poly-def*  
**by** *simp*  
  
**have** *a-carr*:  $a \in \text{carrier } P - \{\mathbf{0}_P\}$   
**using** *sp-4* **by** *simp*  
  
**have** *?T* =  $\{d. ?mip \ d \wedge \text{multiplicity } (\text{mult-of } P) \ d \ a > 0\}$   
**by** (*simp add: pmult-def*)  
**also have**  $\dots = \{d \in ?S. \text{multiplicity } (\text{mult-of } P) \ d \ a > 0\}$   
**using** *p.mult-of.multiplicity-gt-0-iff[OF b-1 b-2 sp-4]*  
**by** (*intro order-antisym subsetI, auto*)  
**finally have** *t*: *?T* =  $\{d \in ?S. \text{multiplicity } (\text{mult-of } P) \ d \ a > 0\}$   
**by** *simp*  
  
**show** *fin-T*: *finite ?T*  
**unfolding** *t*  
**using** *p.mult-of.split-factors(1)*  
 $[\text{OF } \text{monic-polys-are-canonical-irreducibles}]$   
**using** *sp-4* **by** *auto*  
  
**have** *a*:  $x \ [\bigwedge]_P \ (n::\text{nat}) \in \text{carrier } (\text{mult-of } P)$  **if** *?mip x* **for**  $x \ n$   
**proof** –  
**have** *monic-poly R (x [\bigwedge]\_P n)*

```

    using that monic-poly-pow
    unfolding monic-irreducible-poly-def by auto
    thus ?thesis
    using monic-poly-carr-2 by simp
qed

have ?lhs  $\sim_{(mult\text{-of } P)}$ 
  finprod (mult-of P)
    ( $\lambda d. d [\wedge]_{(mult\text{-of } P)} (multiplicity (mult\text{-of } P) d a)$ ) ?T
  unfolding t
  by (intro p.mult-of.split-factors(2)
      [OF monic-polys-are-canonical-irreducibles sp-4])
also have ... =
  finprod (mult-of P) ( $\lambda d. d [\wedge]_P (multiplicity (mult\text{-of } P) d a)$ ) ?T
  by (simp add:nat-pow-mult-of)
also have ... = ?rhs
  using fin-T a
  by (subst p.finprod-mult-of, simp-all add:pmult-def)
finally have ?lhs  $\sim_{(mult\text{-of } P)}$  ?rhs by simp
moreover have monic-poly R ?rhs
  using fin-T
  by (intro monic-poly-prod monic-poly-pow)
  (auto simp:monic-irreducible-poly-def)
ultimately show ?lhs = ?rhs
  using monic-poly-not-assoc assms monic-irreducible-poly-def
  by blast
qed

```

```

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

```

**proof** –

```

let ?mip = monic-irreducible-poly R

```

```

have b:  $d \in carrier P - \{0_P\}$  if ?mip d for d
  using that monic-poly-carr-2
  unfolding monic-irreducible-poly-def by simp
have a:  $d [\wedge]_P n \in carrier P - \{0_P\}$  if ?mip d for d and  $n :: nat$ 
  using b that monic-poly-pow
  unfolding monic-irreducible-poly-def
  by (simp add: p.pow-non-zero)

```

```

have degree f =
  degree ( $\bigotimes_{p d \in \{d. ?mip d \wedge pmult d f > 0\}} d [\wedge]_P pmult d f$ )
  using factor-monic-poly[OF assms(1)] by simp
also have ... =
  ( $\sum_{i \in \{d. ?mip d \wedge 0 < pmult d f\}} degree (i [\wedge]_P pmult i f)$ )

```

**using**  $a$  *assms*(1)  
**by** (*subst degree-prod*[*OF factor-monic-poly-fin*])  
(*simp-all add:Pi-def*)  
**also have** ... =  
 $(\sum i \in \{d. ?mip\ d \wedge 0 < pmult\ d\ f\}. degree\ i * pmult\ i\ f)$   
**using**  $b$  *degree-pow* **by** (*intro sum.cong, auto*)  
**also have** ... =  
 $(\sum d \in \{d. ?mip\ d \wedge 0 < pmult\ d\ f\}. pmult\ d\ f * degree\ d)$   
**by** (*simp add:mult commute*)  
**also have** ... =  
 $sum' (\lambda d. pmult\ d\ f * degree\ d) \{d. ?mip\ d \wedge 0 < pmult\ d\ f\}$   
**using** *sum.eq-sum factor-monic-poly-fin*[*OF assms*(1)] **by** *simp*  
**also have** ... =  $sum' (\lambda d. pmult\ d\ f * degree\ d) \{d. ?mip\ d\}$   
**by** (*intro sum.mono-neutral-cong-left' subsetI, auto*)  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *monic-poly-min-degree*:  
**assumes** *monic-irreducible-poly R f*  
**shows**  $degree\ f \geq 1$   
**using** *assms unfolding monic-irreducible-poly-def monic-poly-def*  
**by** (*intro pirreducible-degree*) *auto*

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

**proof**  
**assume**  $monic-irreducible-poly\ R\ f \wedge degree\ f = 1$   
**hence**  $a: monic-poly\ R\ f\ length\ f = 2$   
**unfolding** *monic-irreducible-poly-def* **by** *auto*  
**then obtain**  $u\ v$  **where**  $f-def: f = [u, v]$   
**by** (*cases f, simp, cases tl f, auto*)

**have**  $u = 1$  **using**  $a$  **unfolding** *monic-poly-def f-def* **by** *simp*  
**moreover have**  $v \in carrier\ R$   
**using**  $a$  **unfolding** *monic-poly-def univ-poly-carrier[symmetric]*  
**unfolding** *polynomial-def f-def* **by** *simp*  
**ultimately have**  $f = [1, \ominus(\ominus v)] (\ominus v) \in carrier\ R$   
**using** *a-inv-closed f-def* **by** *auto*  
**thus**  $(\exists x \in carrier\ R. f = [1_R, \ominus_R x])$  **by** *auto*

**next**  
**assume**  $(\exists x \in carrier\ R. f = [1, \ominus x])$   
**then obtain**  $x$  **where**  $f-def: f = [1, \ominus x]$   $x \in carrier\ R$  **by** *auto*  
**have**  $a: degree\ f = 1$  **using**  $f-def(2)$  **unfolding**  $f-def$  **by** *simp*  
**have**  $b: f \in carrier\ P$   
**using**  $f-def(2)$  **unfolding** *univ-poly-carrier[symmetric]*  
**unfolding**  $f-def$  *polynomial-def* **by** *simp*  
**have**  $c: pirreducible\ (carrier\ R)\ f$   
**by** (*intro degree-one-imp-pirreducible a b*)

**have**  $d$ : *lead-coeff*  $f = 1$  **unfolding**  $f$ -def **by** *simp*  
**show** *monic-irreducible-poly*  $R f \wedge \text{degree } f = 1$   
**using**  $a b c d$   
**unfolding** *monic-irreducible-poly-def monic-poly-def*  
**by** *auto*  
**qed**

**lemma** *multiplicity-ge-iff*:

**assumes** *monic-irreducible-poly*  $R d$   
**assumes**  $f \in \text{carrier } P - \{0_P\}$   
**shows**  $\text{pmult } d f \geq k \longleftrightarrow d \lceil_P k \text{ pdivides } f$   
**proof** –  
**have**  $a:f \in \text{carrier } (\text{mult-of } P)$   
**using** *assms(2)* **by** *simp*  
**have**  $b: d \in \text{carrier } (\text{mult-of } P)$   
**using** *assms(1) monic-poly-carr-2*  
**unfolding** *monic-irreducible-poly-def* **by** *simp*  
**have**  $c: \text{irreducible } (\text{mult-of } P) d$   
**using** *assms(1) monic-poly-carr-2*  
**using** *p.irreducible-imp-irreducible-mult*  
**unfolding** *monic-irreducible-poly-def*  
**unfolding** *ring-irreducible-def monic-poly-def*  
**by** *simp*  
**have**  $d: d \lceil_P k \in \text{carrier } P$  **using**  $b$  **by** *simp*  
  
**have**  $\text{pmult } d f \geq k \longleftrightarrow d \lceil_{(\text{mult-of } P)} k \text{ divides}_{(\text{mult-of } P)} f$   
**unfolding** *pmult-def*  
**by** (*intro p.mult-of.multiplicity-ge-iff a b c*)  
**also have**  $\dots \longleftrightarrow d \lceil_P k \text{ pdivides}_R f$   
**using** *p.divides-imp-divides-mult[OF d assms(2)]*  
**using** *divides-mult-imp-divides*  
**unfolding** *pdivides-def nat-pow-mult-of*  
**by** *auto*  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *multiplicity-ge-1-iff-pdivides*:

**assumes** *monic-irreducible-poly*  $R d f \in \text{carrier } P - \{0_P\}$   
**shows**  $\text{pmult } d f \geq 1 \longleftrightarrow d \text{ pdivides } f$   
**proof** –  
**have**  $d \in \text{carrier } P$   
**using** *assms(1) monic-poly-carr*  
**unfolding** *monic-irreducible-poly-def*  
**by** *simp*  
**thus** *?thesis*  
**using** *multiplicity-ge-iff[OF assms, where k=1]*  
**by** *simp*  
**qed**

**lemma** *divides-monic-poly*:  
**assumes** *monic-poly*  $R$   $f$  *monic-poly*  $R$   $g$   
**assumes**  $\bigwedge d.$  *monic-irreducible-poly*  $R$   $d$   
 $\implies$  *pmult*  $d$   $f \leq$  *pmult*  $d$   $g$   
**shows**  $f$  *pdivides*  $g$   
**proof** –  
**have**  $a:f \in$  *carrier* (*mult-of*  $P$ )  $g \in$  *carrier* (*mult-of*  $P$ )  
**using** *monic-poly-carr-2* *assms*(1,2) **by** *auto*

**have**  $f$  *divides*(*mult-of*  $P$ )  $g$   
**using** *assms*(3) **unfolding** *pmult-def*  
**by** (*intro* *p.mult-of.divides-iff-mult-mono*  
 $[OF$  *a monic-polys-are-canonical-irreducibles*]) *simp*  
**thus** *?thesis*  
**unfolding** *pdivides-def* **using** *divides-mult-imp-divides* **by** *simp*  
**qed**

**end**

**lemma** *monic-poly-hom*:  
**assumes** *monic-poly*  $R$   $f$   
**assumes**  $h \in$  *ring-iso*  $R$   $S$  *domain*  $R$  *domain*  $S$   
**shows** *monic-poly*  $S$  (*map*  $h$   $f$ )  
**proof** –  
**have**  $c: h \in$  *ring-hom*  $R$   $S$   
**using** *assms*(2) *ring-iso-def* **by** *auto*  
**have**  $e: f \in$  *carrier* (*poly-ring*  $R$ )  
**using** *assms*(1) **unfolding** *monic-poly-def* **by** *simp*

**have**  $a:f \neq []$   
**using** *assms*(1) **unfolding** *monic-poly-def* **by** *simp*  
**hence** *map*  $h$   $f \neq []$  **by** *simp*  
**moreover** **have** *lead-coeff*  $f = \mathbf{1}_R$   
**using** *assms*(1) **unfolding** *monic-poly-def* **by** *simp*  
**hence** *lead-coeff* (*map*  $h$   $f$ ) =  $\mathbf{1}_S$   
**using** *ring-hom-one*[ $OF$   $c$ ] **by** (*simp* *add: hd-map*[ $OF$   $a$ ])  
**ultimately** **show** *?thesis*  
**using** *carrier-hom*[ $OF$   $e$  *assms*(2–4)]  
**unfolding** *monic-poly-def* **by** *simp*  
**qed**

**lemma** *monic-irreducible-poly-hom*:  
**assumes** *monic-irreducible-poly*  $R$   $f$   
**assumes**  $h \in$  *ring-iso*  $R$   $S$  *domain*  $R$  *domain*  $S$   
**shows** *monic-irreducible-poly*  $S$  (*map*  $h$   $f$ )  
**proof** –  
**have**  $a:$   
 $pirreducible_R$  (*carrier*  $R$ )  $f$   
 $f \in$  *carrier* (*poly-ring*  $R$ )

```

    monic-poly R f
    using assms(1)
    unfolding monic-poly-def monic-irreducible-poly-def
    by auto

    have pirreducible_S (carrier S) (map h f)
      using a_pirreducible-hom assms by auto
    moreover have monic-poly S (map h f)
      using a_monic-poly-hom[OF - assms(2,3,4)] by simp
    ultimately show ?thesis
      unfolding monic-irreducible-poly-def by simp
qed

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 ∈ 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 -
  interpret p: principal-domain K[X]
    using univ-poly-is-principal[OF assms(1)] by simp

  interpret I: ideal PIdl_K[X] f K[X]
    using p.genideal-ideal[OF assms(2)] by simp

  interpret d: ring Rupt K f
    unfolding rupture-def using I.quotient-is-ring by simp

  have e: subring K R
    using assms(1) subfieldE(1) by auto

  interpret h:
    ring-hom-ring R (| carrier := K |)
      Rupt K f rupture-surj K f ∘ poly-of-const
    using rupture-surj-norm-is-hom[OF e assms(2)]

```

```

using ring-hom-ringI2 subring-is-ring d.ring-axioms e
by blast

have field (R (carrier := K))
  using assms(1) subfield-iff(2) by simp
hence subfield K (R (carrier := K))
  using ring.subfield-iff[OF subring-is-ring[OF e]] by simp
hence b: subfield (rupture-surj K f ' poly-of-const ' K) (Rupt K f)
  unfolding image-image comp-def[symmetric]
  by (intro h.img-is-subfield rupture-one-not-zero assms, simp)

have inj-on poly-of-const K
  using poly-of-const-inj inj-on-subset by auto
moreover have
  poly-of-const ' K  $\subseteq$  (( $\lambda q. q \text{ pmod } f$ ) ' carrier (K [X]))
proof (rule image-subsetI)
  fix x assume x  $\in$  K
  hence f:
    poly-of-const x  $\in$  carrier (K[X])
    degree (poly-of-const x) = 0
    using poly-of-const-over-subfield[OF assms(1)] by auto
  moreover
  have degree (poly-of-const x) < degree f
    using f(2) assms by simp
  hence poly-of-const x pmod f = poly-of-const x
    by (intro pmod-const(2)[OF assms(1)] f assms(2), simp)
  ultimately show
    poly-of-const x  $\in$  (( $\lambda q. q \text{ pmod } f$ ) ' carrier (K [X]))
    by force
qed
hence inj-on (rupture-surj K f) (poly-of-const ' K)
  using rupture-surj-inj-on[OF assms(1,2)] inj-on-subset by blast
ultimately show d: inj-on (rupture-surj K f  $\circ$  poly-of-const) K
  using comp-inj-on by auto

have a: d.dimension (degree f) (rupture-surj K f ' poly-of-const ' K)
  (carrier (Rupt K f))
  using rupture-dimension[OF assms(1-3)] by auto
then obtain base where base-def:
  set base  $\subseteq$  carrier (Rupt K f)
  d.independent (rupture-surj K f ' poly-of-const ' K) base
  length base = degree f
  d.Span (rupture-surj K f ' poly-of-const ' K) base =
  carrier (Rupt K f)
  using d.exists-base[OF b a] by auto
have order (Rupt K f) =
  card (d.Span (rupture-surj K f ' poly-of-const ' K) base)
  unfolding order-def base-def(4) by simp
also have ... =

```

$\text{card } (\text{rupture-surj } K f \text{ ' poly-of-const ' } K) \hat{\text{ length base}}$   
**using**  $d.\text{card-span}[OF b \text{ base-def}(2,1)]$  **by** *simp*  
**also have ...**  
 $= \text{card } ((\text{rupture-surj } K f \circ \text{poly-of-const}) \text{ ' } K) \hat{\text{ degree } f}$   
**using**  $\text{base-def}(3)$  *image-image* **unfolding** *comp-def* **by** *metis*  
**also have ...**  $= \text{card } K \hat{\text{ degree } f}$   
**by**  $(\text{subst } \text{card-image}[OF d], \text{simp})$   
**finally show**  $\text{order } (\text{Rupt } K f) = \text{card } K \hat{\text{ degree } f}$  **by** *simp*  
  
**have**  $\text{char } (\text{Rupt } K f) = \text{char } (R \text{ (} \text{carrier} := K \text{)})$   
**using**  $h.\text{char-consistent } d$  **by** *simp*  
**also have ...**  $= \text{char } R$   
**using**  $\text{char-consistent}[OF \text{subfieldE}(1)[OF \text{assms}(1)]]$  **by** *simp*  
**finally show**  $\text{char } (\text{Rupt } K f) = \text{char } R$  **by** *simp*  
**qed**

**definition** *gauss-poly* **where**

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

**context** *field*

**begin**

**interpretation** *polynomial-ring*  $R$  *carrier*  $R$

**unfolding** *polynomial-ring-def* *polynomial-ring-axioms-def*

**using** *field-axioms* *carrier-is-subfield* **by** *simp*

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** –

**define**  $q$  **where**  $q = m \text{ div } l$

**define**  $r$  **where**  $r = m \text{ mod } l$

**have**  $m\text{-def}$ :  $m = q * l + r$  **and**  $r\text{-range}$ :  $r < l$

**using** *assms* **by**  $(\text{auto } \text{simp } \text{add}:q\text{-def } r\text{-def})$

**have**  $\text{pow-sum-carr}$ :  $(\bigoplus_{P i \in \{..<q\}} (X [\bigwedge_P l] [\bigwedge_P i]) \in \text{carrier } P$

**using** *var-pow-closed*

**by**  $(\text{intro } p.\text{finsum-closed}, \text{simp})$

**have**  $(X [\bigwedge_P (q * l) \ominus_P \mathbf{1}_P]) = ((X [\bigwedge_P l] [\bigwedge_P q]) \ominus_P \mathbf{1}_P$

**using** *var-closed*

**by**  $(\text{subst } p.\text{nat-pow-pow}, \text{simp-all } \text{add}: \text{algebra-simps})$

**also have ...**  $=$

$(X [\bigwedge_P l \ominus_P \mathbf{1}_P]) \otimes_P (\bigoplus_{P i \in \{..<q\}} (X [\bigwedge_P l] [\bigwedge_P i])$

**using** *var-pow-closed*

by (*subst p.geom[symmetric], simp-all*)  
**finally have** *pow-sum-fact*:  $(X [\bigwedge]_P (q * l) \ominus_P \mathbf{1}_P) =$   
 $(X [\bigwedge]_P l \ominus_P \mathbf{1}_P) \otimes_P (\bigoplus_{P^i \in \{.. < q\}} (X_R [\bigwedge]_P l) [\bigwedge]_P i)$   
 by *simp*

**have**  $(X [\bigwedge]_P l \ominus_P \mathbf{1}_P) \text{ divides}_P (X [\bigwedge]_P (q * l) \ominus_P \mathbf{1}_P)$   
 by (*rule dividesI[OF pow-sum-carr pow-sum-fact]*)

**hence**  $c: (X [\bigwedge]_P l \ominus_P \mathbf{1}_P) \text{ divides}_P X [\bigwedge]_P r \otimes_P (X [\bigwedge]_P (q * l) \ominus_P \mathbf{1}_P)$   
 using *var-pow-closed*  
 by (*intro p.divides-prod-l, auto*)

**have**  $(X [\bigwedge]_P m \ominus_P \mathbf{1}_P) = X [\bigwedge]_P (r + q * l) \ominus_P \mathbf{1}_P$   
 unfolding *m-def* using *add commute* by *metis*  
**also have**  $\dots = (X [\bigwedge]_P r) \otimes_P (X [\bigwedge]_P (q * l) \oplus_P (\ominus_P \mathbf{1}_P))$   
 using *var-closed*  
 by (*subst p.nat-pow-mult, auto simp add:a-minus-def*)  
**also have**  $\dots = ((X [\bigwedge]_P r) \otimes_P (X [\bigwedge]_P (q * l) \oplus_P (\ominus_P \mathbf{1}_P)))$   
 $\oplus_P (X [\bigwedge]_P r) \ominus_P \mathbf{1}_P$   
 using *var-pow-closed*  
 by *algebra*  
**also have**  $\dots = (X [\bigwedge]_P r) \otimes_P (X [\bigwedge]_P (q * l) \ominus_P \mathbf{1}_P)$   
 $\oplus_P (X [\bigwedge]_P r) \ominus_P \mathbf{1}_P$   
 by *algebra*  
**also have**  $\dots = (X [\bigwedge]_P r) \otimes_P (X [\bigwedge]_P (q * l) \ominus_P \mathbf{1}_P)$   
 $\oplus_P ((X [\bigwedge]_P r) \ominus_P \mathbf{1}_P)$   
 unfolding *a-minus-def* using *var-pow-closed*  
 by (*subst p.a-assoc, auto*)  
**finally have**  $a: (X [\bigwedge]_P m \ominus_P \mathbf{1}_P) =$   
 $(X [\bigwedge]_P r) \otimes_P (X [\bigwedge]_P (q * l) \ominus_P \mathbf{1}_P) \oplus_P (X [\bigwedge]_P r \ominus_P \mathbf{1}_P)$   
 (*is - = ?x*)  
 by *simp*

**have** *xn-m-1-deg'*:  $\text{degree } (X [\bigwedge]_P n \ominus_P \mathbf{1}_P) = n$   
 if  $n > 0$  for  $n :: \text{nat}$   
**proof** –  
**have**  $\text{degree } (X [\bigwedge]_P n \ominus_P \mathbf{1}_P) = \text{degree } (X [\bigwedge]_P n \oplus_P \ominus_P \mathbf{1}_P)$   
 by (*simp add:a-minus-def*)  
**also have**  $\dots = \max (\text{degree } (X [\bigwedge]_P n)) (\text{degree } (\ominus_P \mathbf{1}_P))$   
 using *var-pow-closed var-pow-carr var-pow-degree*  
 using *univ-poly-a-inv-degree degree-one that*  
 by (*subst degree-add-distinct, auto*)  
**also have**  $\dots = n$   
 using *var-pow-degree degree-one univ-poly-a-inv-degree*  
 by *simp*  
**finally show** *?thesis* by *simp*

qed

```

have xn-m-1-deg:  $\text{degree } (X [\wedge]_P n \ominus_P \mathbf{1}_P) = n$  for  $n :: \text{nat}$ 
proof (cases  $n > 0$ )
  case True
    then show ?thesis using xn-m-1-deg' by auto
  next
    case False
      hence  $n = 0$  by simp
      hence  $\text{degree } (X [\wedge]_P n \ominus_P \mathbf{1}_P) = \text{degree } (\mathbf{0}_P)$ 
        by (intro arg-cong[where  $f = \text{degree}$ ], simp)
      then show ?thesis using False by (simp add:univ-poly-zero)
qed

have b:  $\text{degree } (X [\wedge]_P l \ominus_P \mathbf{1}_P) > \text{degree } (X_R [\wedge]_P r \ominus_P \mathbf{1}_P)$ 
  using r-range unfolding xn-m-1-deg by simp

have xn-m-1-carr:  $X [\wedge]_P n \ominus_P \mathbf{1}_P \in \text{carrier } P$  for  $n :: \text{nat}$ 
  unfolding a-minus-def
  by (intro p.a-closed var-pow-closed, simp)

have ?lhs  $\longleftrightarrow (X [\wedge]_P l \ominus_P \mathbf{1}_P) \text{ pdivides } ?x$ 
  by (subst a, simp)
also have  $\dots \longleftrightarrow (X [\wedge]_P l \ominus_P \mathbf{1}_P) \text{ pdivides } (X [\wedge]_P r \ominus_P \mathbf{1}_P)$ 
  unfolding pdivides-def
  by (intro p.div-sum-iff c var-pow-closed
    xn-m-1-carr p.a-closed p.m-closed)
also have  $\dots \longleftrightarrow r = 0$ 
proof (cases  $r = 0$ )
  case True
    have  $(X [\wedge]_P l \ominus_P \mathbf{1}_P) \text{ pdivides } \mathbf{0}_P$ 
      unfolding univ-poly-zero
      by (intro pdivides-zero xn-m-1-carr)
    also have  $\dots = (X [\wedge]_P r \ominus_P \mathbf{1}_P)$ 
      by (simp add:a-minus-def True) algebra
    finally show ?thesis using True by simp
  next
    case False
      hence  $\text{degree } (X [\wedge]_P r \ominus_P \mathbf{1}_P) > 0$  using xn-m-1-deg by simp
      hence  $X [\wedge]_P r \ominus_P \mathbf{1}_P \neq []$  by auto
      hence  $\neg (X [\wedge]_P l \ominus_P \mathbf{1}_P) \text{ pdivides } (X [\wedge]_P r \ominus_P \mathbf{1}_P)$ 
        using pdivides-imp-degree-le b xn-m-1-carr
        by (metis le-antisym less-or-eq-imp-le nat-neq-iff)
      thus ?thesis using False by simp
qed
also have  $\dots \longleftrightarrow l \text{ dvd } m$ 
  unfolding m-def using r-range assms by auto
finally show ?thesis
  by simp
qed

```

**lemma** *gauss-poly-factor*:  
**assumes**  $n > 0$   
**shows** *gauss-poly*  $R\ n = (X [\wedge]_P (n-1) \ominus_P \mathbf{1}_P) \otimes_P X$  (**is - = ?rhs**)  
**proof** –  
**have**  $a:1 + (n - 1) = n$   
**using** *assms* **by** *simp*  
**have** *gauss-poly*  $R\ n = X [\wedge]_P (1+(n-1)) \ominus_P X$   
**unfolding** *gauss-poly-def* **by** (*subst a, simp*)  
**also have**  $\dots = (X [\wedge]_P (n-1)) \otimes_P X \ominus_P \mathbf{1}_P \otimes_P X$   
**using** *var-closed* **by** *simp*  
**also have**  $\dots = ?rhs$   
**unfolding** *a-minus-def* **using** *var-closed l-one*  
**by** (*subst p.l-distr, auto, algebra*)  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *var-neq-zero*:  $X \neq \mathbf{0}_P$   
**by** (*simp add:var-def univ-poly-zero*)

**lemma** *var-pow-eq-one-iff*:  $X [\wedge]_P k = \mathbf{1}_P \iff k = (0::nat)$   
**proof** (*cases k=0*)  
**case** *True*  
**then show** *?thesis* **using** *var-closed(1)* **by** *simp*  
**next**  
**case** *False*  
**have**  $degree (X_R [\wedge]_P k) = k$   
**using** *var-pow-degree* **by** *simp*  
**also have**  $\dots \neq degree (\mathbf{1}_P)$  **using** *False degree-one* **by** *simp*  
**finally have**  $degree (X_R [\wedge]_P k) \neq degree \mathbf{1}_P$  **by** *simp*  
**then show** *?thesis* **by** *auto*  
**qed**

**lemma** *gauss-poly-carr*: *gauss-poly*  $R\ n \in carrier\ P$   
**using** *var-closed(1)*  
**unfolding** *gauss-poly-def* **by** *simp*

**lemma** *gauss-poly-degree*:  
**assumes**  $n > 1$   
**shows**  $degree (gauss-poly\ R\ n) = n$   
**proof** –  
**have**  $degree (gauss-poly\ R\ n) = max\ n\ 1$   
**unfolding** *gauss-poly-def a-minus-def*  
**using** *var-pow-carr var-carr degree-var*  
**using** *var-pow-degree univ-poly-a-inv-degree*  
**using** *assms* **by** (*subst degree-add-distinct, auto*)  
**also have**  $\dots = n$  **using** *assms* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *gauss-poly-not-zero*:  
**assumes**  $n > 1$   
**shows**  $\text{gauss-poly } R \ n \neq \mathbf{0}_P$   
**proof** –  
**have**  $\text{degree } (\text{gauss-poly } R \ n) \neq \text{degree } (\mathbf{0}_P)$   
**using** *assms* **by** (*subst gauss-poly-degree, simp-all add:univ-poly-zero*)  
**thus** *?thesis* **by** *auto*  
**qed**

**lemma** *gauss-poly-monic*:  
**assumes**  $n > 1$   
**shows**  $\text{monic-poly } R \ (\text{gauss-poly } R \ n)$   
**proof** –  
**have**  $\text{monic-poly } R \ (X \ [\frown]_P \ n)$   
**by** (*intro monic-poly-pow monic-poly-var*)  
**moreover** **have**  $\ominus_P X \in \text{carrier } P$   
**using** *var-closed* **by** *simp*  
**moreover** **have**  $\text{degree } (\ominus_P X) < \text{degree } (X \ [\frown]_P \ n)$   
**using** *assms univ-poly-a-inv-degree var-closed*  
**using** *degree-var*  
**unfolding** *var-pow-degree* **by** (*simp*)  
**ultimately** **show** *?thesis*  
**unfolding** *gauss-poly-def a-minus-def*  
**by** (*intro monic-poly-add-distinct, auto*)  
**qed**

**lemma** *geom-nat*:  
**fixes**  $q :: \text{nat}$   
**fixes**  $x :: - :: \{\text{comm-ring, monoid-mult}\}$   
**shows**  $(x-1) * (\sum i \in \{..<q\}. x^i) = x^q - 1$   
**by** (*induction q, auto simp:algebra-simps*)

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)*  
**proof** –  
**define**  $q$  **where**  $q = m \ \text{div} \ l$   
**define**  $r$  **where**  $r = m \ \text{mod} \ l$   
**have** *m-def*:  $m = q * l + r$  **and** *r-range*:  $r < l \ r \geq 0$   
**using** *assms* **by** (*auto simp add:q-def r-def*)  
  
**have**  $a^{l * q} - 1 = (a^l)^q - 1$   
**by** (*simp add: power-mult*)  
**also** **have**  $\dots = (a^l - 1) * (\sum i \in \{..<q\}. (a^l)^i)$

by *(subst geom-nat[symmetric], simp)*  
**finally have**  $a^{\wedge}(l * q) - 1 = (a^{\wedge}l - 1) * (\sum i \in \{..<q\}. (a^{\wedge}l)^{\wedge}i)$   
 by *simp*  
**hence**  $c:a^{\wedge}l - 1 \text{ dvd } a^{\wedge}r * (a^{\wedge}(q * l) - 1)$  **by** *(simp add:mult.commute)*

**have**  $a^{\wedge}m - 1 = a^{\wedge}(r + q * l) - 1$   
**unfolding** *m-def* **using** *add.commute* **by** *metis*  
**also have**  $\dots = (a^{\wedge}r) * (a^{\wedge}(q * l)) - 1$   
 by *(simp add: power-add)*  
**also have**  $\dots = ((a^{\wedge}r) * (a^{\wedge}(q * l) - 1)) + (a^{\wedge}r) - 1$   
 by *(simp add: right-diff-distrib)*  
**also have**  $\dots = (a^{\wedge}r) * (a^{\wedge}(q * l) - 1) + ((a^{\wedge}r) - 1)$   
 by *simp*  
**finally have** *a:*  
 $a^{\wedge}m - 1 = (a^{\wedge}r) * (a^{\wedge}(q * l) - 1) + ((a^{\wedge}r) - 1)$   
*(is - = ?x)*  
 by *simp*

**have**  $?lhs \longleftrightarrow (a^{\wedge}l - 1) \text{ dvd } ?x$   
 by *(subst a, simp)*  
**also have**  $\dots \longleftrightarrow (a^{\wedge}l - 1) \text{ dvd } (a^{\wedge}r - 1)$   
 using *c dvd-add-right-iff* **by** *auto*  
**also have**  $\dots \longleftrightarrow r = 0$

**proof**  
**assume**  $a^{\wedge}l - 1 \text{ dvd } a^{\wedge}r - 1$   
**hence**  $a^{\wedge}l - 1 \leq a^{\wedge}r - 1 \vee r = 0$   
 using *assms r-range zdvd-not-zless* **by** *force*  
**moreover have**  $a^{\wedge}r < a^{\wedge}l$  **using** *assms r-range* **by** *simp*  
**ultimately show**  $r = 0$  **by** *simp*

**next**  
**assume**  $r = 0$   
**thus**  $a^{\wedge}l - 1 \text{ dvd } a^{\wedge}r - 1$  **by** *simp*

**qed**  
**also have**  $\dots \longleftrightarrow l \text{ dvd } m$   
 using *r-def* **by** *auto*  
**finally show** *?thesis* **by** *simp*

**qed**

**lemma** *gauss-poly-div-gauss-poly-iff:*  
**assumes**  $m > 0 \ n > 0 \ a > 1$   
**shows** *gauss-poly*  $R (a^{\wedge}n)$  *pdivides*<sub>R</sub> *gauss-poly*  $R (a^{\wedge}m)$   
 $\longleftrightarrow n \text{ dvd } m$  **(is ?lhs=?rhs)**

**proof** –  
**have**  $a:a^{\wedge}m > 1$  **using** *assms one-less-power* **by** *blast*  
**hence** *a1:*  $a^{\wedge}m > 0$  **by** *linarith*  
**have**  $b:a^{\wedge}n > 1$  **using** *assms one-less-power* **by** *blast*  
**hence** *b1:*  $a^{\wedge}n > 0$  **by** *linarith*

**have**  $?lhs \longleftrightarrow$

```

(X [∧]P (a∧n-1) ⊖P 1P) ⊗P X pdivides
(X [∧]P (a∧m-1) ⊖P 1P) ⊗P X
using gauss-poly-factor a1 b1 by simp
also have ... ↔
(X [∧]P (a∧n-1) ⊖P 1P) pdivides
(X [∧]P (a∧m-1) ⊖P 1P)
using var-closed a b var-neq-zero
by (subst pdivides-mult-r, simp-all add:var-pow-eq-one-iff)
also have ... ↔ a∧n-1 dvd a∧m-1
using b
by (subst gauss-poly-div-gauss-poly-iff-1) simp-all
also have ... ↔ int (a∧n-1) dvd int (a∧m-1)
by (subst of-nat-dvd-iff, simp)
also have ... ↔ int a∧n-1 dvd int a∧m-1
using a b by (simp add:of-nat-diff)
also have ... ↔ n dvd m
using assms
by (subst gauss-poly-div-gauss-poly-iff-2) simp-all
finally show ?thesis by simp
qed

end

context finite-field
begin

interpretation polynomial-ring R carrier R
unfolding polynomial-ring-def polynomial-ring-axioms-def
using field-axioms carrier-is-subfield by simp

lemma div-gauss-poly-iff:
assumes n > 0
assumes monic-irreducible-poly R f
shows f pdividesR gauss-poly R (order R∧n) ↔ degree f dvd n
proof -
have f-carr: f ∈ carrier P
using assms(2) unfolding monic-irreducible-poly-def
unfolding monic-poly-def by simp
have f-deg: degree f > 0
using assms(2) monic-poly-min-degree by fastforce

define K where K = RuptR (carrier R) f
have field-K: field K
using assms(2) unfolding K-def monic-irreducible-poly-def
unfolding monic-poly-def
by (subst rupture-is-field-iff-pirreducible) auto
have a: order K = order R∧degree f
using rupture-order[OF carrier-is-subfield] f-carr f-deg
unfolding K-def order-def by simp

```

```

have char-K: char K = char R
  using rupture-char[OF carrier-is-subfield] f-carr f-deg
  unfolding K-def by simp

have card (carrier K) > 0
  using a f-deg finite-field-min-order unfolding order-def by simp
hence d: finite (carrier K) using card-ge-0-finite by auto
interpret f: finite-field K
  using field-K d by (intro finite-fieldI, simp-all)
interpret fp: polynomial-ring K (carrier K)
  unfolding polynomial-ring-def polynomial-ring-axioms-def
  using f.field-axioms f.carrier-is-subfield by simp

define  $\varphi$  where  $\varphi = \text{rupture-surj (carrier R) f}$ 
interpret h:ring-hom-ring P K  $\varphi$ 
  unfolding K-def  $\varphi$ -def using f-carr rupture-surj-hom by simp

have embed-inj: inj-on ( $\varphi \circ \text{poly-of-const}$ ) (carrier R)
  unfolding  $\varphi$ -def
  using embed-inj[OF carrier-is-subfield f-carr f-deg] by simp

interpret r:ring-hom-ring R P poly-of-const
  using canonical-embedding-ring-hom by simp

obtain rn where order R = char  $K^{\wedge}rn$  rn > 0
  unfolding char-K using finite-field-order by auto
hence ord-rn: order R  $\wedge n = \text{char } K^{\wedge}(rn * n)$  using assms(1)
  by (simp add: power-mult)

interpret q:ring-hom-cring K K  $\lambda x. x [\wedge]_K \text{order } R^{\wedge}n$ 
  using ord-rn
  by (intro f.frobenius-hom f.finite-carr-imp-char-ge-0 d, simp)

have o1: order R  $\wedge \text{degree } f > 1$ 
  using f-deg finite-field-min-order one-less-power
  by blast
hence o11: order R  $\wedge \text{degree } f > 0$  by linarith
have o2: order R  $\wedge n > 1$ 
  using assms(1) finite-field-min-order one-less-power
  by blast
hence o21: order R  $\wedge n > 0$  by linarith
let ?g1 = gauss-poly K (order R  $\wedge \text{degree } f$ )
let ?g2 = gauss-poly K (order R  $\wedge n$ )

have g1-monic: monic-poly K ?g1
  using f.gauss-poly-monic[OF o1] by simp

have c:x  $[\wedge]_K (\text{order } R^{\wedge} \text{degree } f) = x$  if b:x  $\in \text{carrier } K$  for x
  using b d order-pow-eq-self

```

**unfolding**  $a$ [*symmetric*]  
**by** (*intro f.order-pow-eq-self, auto*)

**have**  $k$ -*cycle*:

$\varphi$  (*poly-of-const*  $x$ ) [ $\uparrow$ ] <sub>$K$</sub>  (*order*  $R^{\wedge}n$ ) =  $\varphi$ (*poly-of-const*  $x$ )  
**if**  $k$ -*cycle-1*:  $x \in \text{carrier } R$  **for**  $x$

**proof** –

**have**  $\varphi$  (*poly-of-const*  $x$ ) [ $\uparrow$ ] <sub>$K$</sub>  (*order*  $R^{\wedge}n$ ) =  
 $\varphi$  (*poly-of-const* ( $x$  [ $\uparrow$ ] <sub>$R$</sub>  (*order*  $R^{\wedge}n$ )))  
**using**  $k$ -*cycle-1* **by** (*simp add: h.hom-nat-pow r.hom-nat-pow*)  
**also have** ... =  $\varphi$  (*poly-of-const*  $x$ )  
**using** *order-pow-eq-self'*  $k$ -*cycle-1* **by** *simp*  
**finally show** *?thesis* **by** *simp*

**qed**

**have** *roots-g1*:  $\text{pmult}_K d \ ?g1 \geq 1$

**if** *roots-g1-assms*: *degree*  $d = 1$  *monic-irreducible-poly*  $K d$  **for**  $d$

**proof** –

**obtain**  $x$  **where**  $x$ -*def*:  $x \in \text{carrier } K d = [\mathbf{1}_K, \ominus_K x]$   
**using** *f.degree-one-monic-poly roots-g1-assms* **by** *auto*  
**interpret**  $x$ :*ring-hom-cring poly-ring*  $K K$  ( $\lambda p. f.\text{eval } p x$ )  
**by** (*intro fp.eval-cring-hom x-def*)  
**have** *ring.eval*  $K \ ?g1 x = \mathbf{0}_K$   
**unfolding** *gauss-poly-def a-minus-def*  
**using** *fp.var-closed f.eval-var x-def c*  
**by** (*simp, algebra*)  
**hence**  $f.\text{is-root } ?g1 x$   
**using**  $x$ -*def* *f.gauss-poly-not-zero[OF o1]*  
**unfolding**  $f.\text{is-root-def univ-poly-zero}$  **by** *simp*  
**hence**  $[\mathbf{1}_K, \ominus_K x]$  *pdivides* <sub>$K$</sub>   $?g1$   
**using**  $f.\text{is-root-imp-pdivides}$   $f.\text{gauss-poly-carr}$  **by** *simp*  
**hence**  $d$  *pdivides* <sub>$K$</sub>   $?g1$  **by** (*simp add:x-def*)  
**thus**  $\text{pmult}_K d \ ?g1 \geq 1$   
**using** *that f.gauss-poly-not-zero f.gauss-poly-carr o1*  
**by** (*subst f.multiplicity-ge-1-iff-pdivides, simp-all*)

**qed**

**show** *?thesis*

**proof**

**assume**  $f:f$  *pdivides* <sub>$R$</sub>  *gauss-poly*  $R$  (*order*  $R^{\wedge}n$ )

**have**  $(\varphi X)$  [ $\uparrow$ ] <sub>$K$</sub>  (*order*  $R^{\wedge}n$ )  $\ominus_K (\varphi X_R) =$

$\varphi$  (*gauss-poly*  $R$  (*order*  $R^{\wedge}n$ ))

**unfolding** *gauss-poly-def a-minus-def* **using** *var-closed*

**by** (*simp add: h.hom-nat-pow*)

**also have** ... =  $\mathbf{0}_K$

**unfolding**  $K$ -*def*  $\varphi$ -*def* **using**  $f$ -*carr gauss-poly-carr*  $f$

**by** (*subst rupture-eq-0-iff, simp-all*)

**finally have**  $(\varphi X_R)$  [ $\uparrow$ ] <sub>$K$</sub>  (*order*  $R^{\wedge}n$ )  $\ominus_K (\varphi X_R) = \mathbf{0}_K$

**by** *simp*

**hence**  $g:(\varphi X) [\frown]_K (\text{order } R \hat{n}) = (\varphi X)$   
**using** *var-closed* **by** *simp*

**have** *roots-g2*:  $\text{pmult}_K d \text{ ?g2} \geq 1$   
**if** *roots-g2-assms*: *degree*  $d = 1$  *monic-irreducible-poly*  $K d$  **for**  $d$   
**proof** –  
**obtain**  $y$  **where** *y-def*:  $y \in \text{carrier } K d = [\mathbf{1}_K, \ominus_K y]$   
**using** *f.degree-one-monic-poly roots-g2-assms* **by** *auto*

**interpret** *x:ring-hom-cring poly-ring*  $K K (\lambda p. f.\text{eval } p y)$   
**by** (*intro fp.eval-cring-hom y-def*)  
**obtain**  $x$  **where** *x-def*:  $x \in \text{carrier } P y = \varphi x$   
**using** *y-def* **unfolding** *φ-def K-def rupture-def*  
**unfolding** *FactRing-def A-RCOSETS-def'*  
**by** *auto*

**let**  $\text{?}\tau = \lambda i. \text{poly-of-const } (\text{coeff } x i)$   
**have** *test*:  $\text{?}\tau i \in \text{carrier } P$  **for**  $i$   
**by** (*intro r.hom-closed coeff-range x-def*)  
**have** *test-2*:  $\text{coeff } x i \in \text{carrier } R$  **for**  $i$   
**by** (*intro coeff-range x-def*)

**have** *x-coeff-carr*:  $i \in \text{set } x \implies i \in \text{carrier } R$  **for**  $i$   
**using** *x-def(1)*  
**by** (*auto simp add:univ-poly-carrier[symmetric] polynomial-def*)

**have** *a:map*  $(\varphi \circ \text{poly-of-const}) x \in \text{carrier } (\text{poly-ring } K)$   
**using** *rupture-surj-norm-is-hom[OF f-carr]*  
**using** *domain-axioms f.domain-axioms embed-inj*  
**by** (*intro carrier-hom'[OF x-def(1)]*)  
*(simp-all add:φ-def K-def)*

**have**  $(\varphi x) [\frown]_K (\text{order } R \hat{n}) =$   
 $f.\text{eval } (\text{map } (\varphi \circ \text{poly-of-const}) x) (\varphi X) [\frown]_K (\text{order } R \hat{n})$   
**unfolding** *φ-def K-def*  
**by** (*subst rupture-surj-as-eval[OF f-carr x-def(1)], simp*)  
**also have** ... =  
 $f.\text{eval } (\text{map } (\lambda x. \varphi (\text{poly-of-const } x) [\frown]_K \text{order } R \hat{n}) x) (\varphi X)$   
**using** *a h.hom-closed var-closed(1)*  
**by** (*subst q.ring.eval-hom[OF f.carrier-is-subring]*)  
*(simp-all add:comp-def g)*

**also have** ... =  $f.\text{eval } (\text{map } (\lambda x. \varphi (\text{poly-of-const } x)) x) (\varphi X)$   
**using** *k-cycle x-coeff-carr*  
**by** (*intro arg-cong2[where f=f.eval] map-cong, simp-all*)

**also have** ... =  $(\varphi x)$   
**unfolding** *φ-def K-def*  
**by** (*subst rupture-surj-as-eval[OF f-carr x-def(1)], simp add:comp-def*)  
**finally have**  $\varphi x [\frown]_K \text{order } R \hat{n} = \varphi x$  **by** *simp*

**hence**  $y [\frown]_K (\text{order } R \hat{n}) = y$  **using** *x-def* **by** *simp*

```

hence ring.eval K ?g2 y = 0_K
  unfolding gauss-poly-def a-minus-def
  using fp.var-closed f.eval-var y-def
  by (simp, algebra)
hence f.is-root ?g2 y
  using y-def f.gauss-poly-not-zero[OF o2]
  unfolding f.is-root-def univ-poly-zero by simp
hence d pdivides_K ?g2
  unfolding y-def
  by (intro f.is-root-imp-pdivides f.gauss-poly-carr, simp)
thus pmult_K d ?g2 ≥ 1
  using that f.gauss-poly-carr f.gauss-poly-not-zero o2
  by (subst f.multiplicity-ge-1-iff-pdivides, auto)
qed

have inv-k-inj: inj-on (λx. ⊖_K x) (carrier K)
  by (intro inj-onI, metis f.minus-minus)
let ?mip = monic-irreducible-poly K

have sum' (λd. pmult_K d ?g1 * degree d) {d. ?mip d} = degree
?g1
  using f.gauss-poly-monic o1
  by (subst f.degree-monic-poly', simp-all)
also have ... = order K
  using f.gauss-poly-degree o1 a by simp
also have ... = card ((λk. [1_K, ⊖_K k]) ' carrier K)
  unfolding order-def using inj-onD[OF inv-k-inj]
  by (intro card-image[symmetric] inj-onI) (simp-all)
also have ... = card {d. ?mip d ∧ degree d = 1}
  using f.degree-one-monic-poly
  by (intro arg-cong[where f=card], simp add:set-eq-iff image-iff)
also have ... = sum (λd. 1) {d. ?mip d ∧ degree d = 1}
  by simp
also have ... = sum' (λd. 1) {d. ?mip d ∧ degree d = 1}
  by (intro sum.eq-sum[symmetric]
      finite-subset[OF - fp.finite-poly(1)[OF d]])
      (auto simp:monic-irreducible-poly-def monic-poly-def)
also have ... = sum' (λd. of-bool (degree d = 1)) {d. ?mip d}
  by (intro sum.mono-neutral-cong-left' subsetI, simp-all)
also have ... ≤ sum' (λd. of-bool (degree d = 1)) {d. ?mip d}
  by simp
finally have sum' (λd. pmult_K d ?g1 * degree d) {d. ?mip d}
  ≤ sum' (λd. of-bool (degree d = 1)) {d. ?mip d}
  by simp
moreover have
  pmult_K d ?g1 * degree d ≥ of-bool (degree d = 1)
  if v:monic-irreducible-poly K d for d
proof (cases degree d = 1)
case True

```

**then obtain**  $x$  **where**  $x \in \text{carrier } K \ d = [\mathbf{1}_K, \ominus_K x]$   
**using**  $f.\text{degree-one-monic-poly } v$  **by**  $\text{auto}$   
**hence**  $\text{pmult}_K \ d \ ?g1 \geq 1$   
**using**  $\text{roots-g1 } v$  **by**  $\text{simp}$   
**then show**  $?thesis$  **using**  $\text{True}$  **by**  $\text{simp}$   
**next**  
**case**  $\text{False}$   
**then show**  $?thesis$  **by**  $\text{simp}$   
**qed**  
**moreover have**  
 $\text{finite } \{d. ?mip \ d \wedge \text{pmult}_K \ d \ ?g1 * \text{degree } d > 0\}$   
**by**  $(\text{intro } \text{finite-subset}[OF - f.\text{factor-monic-poly-fin}[OF \ g1\text{-monic}]]$   
 $\text{subsetI}) \text{ simp}$   
**ultimately have**  $v2$ :  
 $\forall d \in \{d. ?mip \ d\}. \text{pmult}_K \ d \ ?g1 * \text{degree } d =$   
 $\text{of-bool } (\text{degree } d = 1)$   
**by**  $(\text{intro } \text{sum'-eq-iff}, \text{simp-all } \text{add:not-le})$   
**have**  $\text{pmult}_K \ d \ ?g1 \leq \text{pmult}_K \ d \ ?g2$  **if**  $?mip \ d$  **for**  $d$   
**proof**  $(\text{cases } \text{degree } d = 1)$   
**case**  $\text{True}$   
**hence**  $\text{pmult}_K \ d \ ?g1 = 1$  **using**  $v2$  **that** **by**  $\text{auto}$   
**also have**  $\dots \leq \text{pmult}_K \ d \ ?g2$   
**by**  $(\text{intro } \text{roots-g2 } \text{True } \text{that})$   
**finally show**  $?thesis$  **by**  $\text{simp}$   
**next**  
**case**  $\text{False}$   
**hence**  $\text{degree } d > 1$   
**using**  $f.\text{monic-poly-min-degree}[OF \ \text{that}]$  **by**  $\text{simp}$   
**hence**  $\text{pmult}_K \ d \ ?g1 = 0$  **using**  $v2$  **that** **by**  $\text{force}$   
**then show**  $?thesis$  **by**  $\text{simp}$   
**qed**  
**hence**  $?g1 \ \text{pdivides}_K \ ?g2$   
**using**  $o1 \ o2 \ f.\text{divides-monic-poly } f.\text{gauss-poly-monic}$  **by**  $\text{simp}$   
**thus**  $\text{degree } f \ \text{dvd } n$   
**by**  $(\text{subst } (\text{asm}) \ f.\text{gauss-poly-div-gauss-poly-iff}$   
 $[OF \ \text{assms}(1) \ f\text{-deg } \text{finite-field-min-order}], \text{ simp})$   
**next**  
**have**  $d:\varphi \ X_R \in \text{carrier } K$   
**by**  $(\text{intro } h.\text{hom-closed } \text{var-closed})$   
  
**have**  $\varphi \ (\text{gauss-poly } R \ (\text{order } R \ \widehat{\text{degree}} \ f)) =$   
 $(\varphi \ X_R) \ [\ ]_K \ (\text{order } R \ \widehat{\text{degree}} \ f) \ \ominus_K \ (\varphi \ X_R)$   
**unfolding**  $\text{gauss-poly-def } a\text{-minus-def}$  **using**  $\text{var-closed}$   
**by**  $(\text{simp } \text{add: } h.\text{hom-nat-pow})$   
**also have**  $\dots = \mathbf{0}_K$   
**using**  $c \ d$  **by**  $\text{simp}$   
**finally have**  $\varphi \ (\text{gauss-poly } R \ (\text{order } R \ \widehat{\text{degree}} \ f)) = \mathbf{0}_K$  **by**  $\text{simp}$   
**hence**  $f \ \text{pdivides}_R \ \text{gauss-poly } R \ (\text{order } R \ \widehat{\text{degree}} \ f)$   
**unfolding**  $K\text{-def } \varphi\text{-def}$  **using**  $f\text{-carr } \text{gauss-poly-carr}$

by (*subst (asm) rupture-eq-0-iff, simp-all*)  
**moreover assume** *degree f dvd n*  
  
**hence** *gauss-poly R (order R ^ degree f) pdivides*  
*(gauss-poly R (order R ^ n))*  
**using** *gauss-poly-div-gauss-poly-iff*  
*[OF assms(1) f-deg finite-field-min-order]*  
**by** *simp*  
**ultimately show** *f pdivides<sub>R</sub> gauss-poly R (order R ^ n)*  
**using** *f-carr a p.divides-trans unfolding pdivides-def* **by** *blast*  
**qed**  
**qed**

**lemma** *gauss-poly-splitted:*  
*splitted (gauss-poly R (order R))*  
**proof** –  
**have** *degree q ≤ 1* **if**  
*q ∈ carrier P*  
*pirreducible (carrier R) q*  
*q pdivides gauss-poly R (order R)* **for** *q*  
**proof** –  
**have** *q-carr: q ∈ carrier (mult-of P)*  
**using** *that unfolding ring-irreducible-def* **by** *simp*  
**moreover have** *irreducible (mult-of P) q*  
**using** *that unfolding ring-irreducible-def*  
**by** (*intro p.irreducible-imp-irreducible-mult that, simp-all*)  
**ultimately obtain** *p* **where** *p-def:*  
*monic-irreducible-poly R p q ~<sub>mult-of P</sub> P*  
**using** *monic-poly-span* **by** *auto*  
**have** *p-carr: p ∈ carrier P p ≠ []*  
**using** *p-def(1)*  
**unfolding** *monic-irreducible-poly-def monic-poly-def*  
**by** *auto*  
**moreover have** *p divides<sub>mult-of P</sub> q*  
**using** *associatedE[OF p-def(2)]* **by** *auto*  
**hence** *p pdivides q*  
**unfolding** *pdivides-def* **using** *divides-mult-imp-divides* **by** *simp*  
**moreover have** *q pdivides gauss-poly R (order R ^ 1)*  
**using** *that by simp*  
**ultimately have** *p pdivides gauss-poly R (order R ^ 1)*  
**unfolding** *pdivides-def* **using** *p.divides-trans* **by** *blast*  
**hence** *degree p dvd 1*  
**using** *div-gauss-poly-iff[where n=1] p-def(1)* **by** *simp*  
**hence** *degree p = 1* **by** *simp*  
**moreover have** *q divides<sub>mult-of P</sub> p*  
**using** *associatedE[OF p-def(2)]* **by** *auto*  
**hence** *q pdivides p*  
**unfolding** *pdivides-def* **using** *divides-mult-imp-divides* **by** *simp*  
**hence** *degree q ≤ degree p*

```

    using that p-carr
    by (intro pdivides-imp-degree-le) auto
    ultimately show ?thesis by simp
qed

```

```

thus ?thesis
  using gauss-poly-carr
  by (intro trivial-factors-imp-splitted, auto)
qed

```

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
     $\text{pmult}_R f (\text{gauss-poly } R (\text{order } R^{\wedge} n)) = \text{of-bool } (\text{degree } f \text{ dvd } n)$ 
proof (cases degree  $f$  dvd  $n$ )
  case True
  let ? $g$  = gauss-poly  $R (\text{order } R^{\wedge} n)$ 
  have  $f\text{-carr}: f \in \text{carrier } P f \neq []$ 
    using assms(2)
  unfolding monic-irreducible-poly-def monic-poly-def
  by auto

```

```

  have  $o2: \text{order } R^{\wedge} n > 1$ 
    using finite-field-min-order assms(1) one-less-power by blast
  hence  $o21: \text{order } R^{\wedge} n > 0$  by linarith

```

```

obtain  $d :: \text{nat}$  where order-dim:  $\text{order } R = \text{char } R^{\wedge} d$   $d > 0$ 
  using finite-field-order by blast
have  $d * n > 0$  using order-dim assms by simp
hence char-dvd-order:  $\text{int } (\text{char } R) \text{ dvd int } (\text{order } R^{\wedge} n)$ 
  unfolding order-dim
  using finite-carr-imp-char-ge-0[OF finite-carrier]
  by (simp add:power-mult[symmetric])

```

```

interpret  $h$ : ring-hom-ring  $R P$  poly-of-const
  using canonical-embedding-ring-hom by simp

```

```

have  $f$  pdivides $_R$  ? $g$ 
  using True div-gauss-poly-iff[OF assms] by simp
hence  $\text{pmult}_R f ?g \geq 1$ 
  using multiplicity-ge-1-iff-pdivides[OF assms(2)]
  using gauss-poly-carr gauss-poly-not-zero[OF o2]
  by auto
moreover have  $\text{pmult}_R f ?g < 2$ 
proof (rule ccontr)

```

**assume**  $\neg \text{pmult}_R f ?g < 2$   
**hence**  $\text{pmult}_R f ?g \geq 2$  **by** *simp*  
**hence**  $(f [\bigwedge]_P (2::\text{nat})) \text{pdivides}_R ?g$   
**using** *gauss-poly-carr gauss-poly-not-zero[OF o2]*  
**by**  $(\text{subst } (\text{asm } \text{multiplicity-ge-iff}[OF \text{assms}(2)]) \text{simp-all})$   
**hence**  $(f [\bigwedge]_P (2::\text{nat})) \text{divides}_{\text{mult-of } P} ?g$   
**unfolding** *pdivides-def*  
**using** *f-carr gauss-poly-not-zero o2 gauss-poly-carr*  
**by**  $(\text{intro } p.\text{divides-imp-divides-mult}) \text{simp-all}$   
**then obtain**  $h$  **where**  $h\text{-def}$ :  
 $h \in \text{carrier } (\text{mult-of } P)$   
 $?g = f [\bigwedge]_P (2::\text{nat}) \otimes_P h$   
**using** *dividesD* **by** *auto*  
**have**  $\ominus_P \mathbf{1}_P = \text{int-embed } P (\text{order } R \wedge n)$   
 $\otimes_P (X_R [\bigwedge]_P (\text{order } R \wedge n - 1)) \ominus_P \mathbf{1}_P$   
**using** *var-closed*  
**apply**  $(\text{subst } \text{int-embed-consistent-with-poly-of-const})$   
**apply**  $(\text{subst } \text{iffD2}[OF \text{embed-char-eq-0-iff char-dvd-order}])$   
**by**  $(\text{simp } \text{add:a-minus-def})$   
**also have**  $\dots = \text{pderiv}_R (X_R [\bigwedge]_P \text{order } R \wedge n) \ominus_P \text{pderiv}_R X_R$   
**using** *pderiv-var*  
**by**  $(\text{subst } \text{pderiv-var-pow}[OF o21], \text{simp})$   
**also have**  $\dots = \text{pderiv}_R ?g$   
**unfolding** *gauss-poly-def a-minus-def* **using** *var-closed*  
**by**  $(\text{subst } \text{pderiv-add}, \text{simp-all } \text{add:pderiv-inv})$   
**also have**  $\dots = \text{pderiv}_R (f [\bigwedge]_P (2::\text{nat}) \otimes_P h)$   
**using**  $h\text{-def}(2)$  **by** *simp*  
**also have**  $\dots = \text{pderiv}_R (f [\bigwedge]_P (2::\text{nat})) \otimes_P h$   
 $\oplus_P (f [\bigwedge]_P (2::\text{nat})) \otimes_P \text{pderiv}_R h$   
**using** *f-carr h-def*  
**by**  $(\text{intro } \text{pderiv-mult}, \text{simp-all})$   
**also have**  $\dots = \text{int-embed } P 2 \otimes_P f \otimes_P \text{pderiv}_R f \otimes_P h$   
 $\oplus_P f \otimes_P f \otimes_P \text{pderiv}_R h$   
**using** *f-carr*  
**by**  $(\text{subst } \text{pderiv-pow}, \text{simp-all } \text{add:numeral-eq-Suc})$   
**also have**  $\dots = f \otimes_P (\text{int-embed } P 2 \otimes_P \text{pderiv}_R f \otimes_P h)$   
 $\oplus_P f \otimes_P (f \otimes_P \text{pderiv}_R h)$   
**using** *f-carr pderiv-carr h-def p.int-embed-closed*  
**apply**  $(\text{intro } \text{arg-cong2}[\text{where } f=(\oplus_P)])$   
**by**  $(\text{subst } p.m\text{-comm}, \text{simp-all } \text{add:p.m-assoc})$   
**also have**  $\dots = f \otimes_P$   
 $(\text{int-embed } P 2 \otimes_P \text{pderiv}_R f \otimes_P h \oplus_P f \otimes_P \text{pderiv}_R h)$   
**using** *f-carr pderiv-carr h-def p.int-embed-closed*  
**by**  $(\text{subst } p.r\text{-distr}, \text{simp-all})$   
**finally have**  $\ominus_P \mathbf{1}_P = f \otimes_P$   
 $(\text{int-embed } P 2 \otimes_P \text{pderiv}_R f \otimes_P h \oplus_P f \otimes_P \text{pderiv}_R h)$   
 $(\text{is } - = f \otimes_P ?q)$   
**by** *simp*

```

hence  $f$  pdivides $R$   $\ominus_P \mathbf{1}_P$ 
  unfolding factor-def pdivides-def
  using f-carr pderiv-carr h-def p.int-embed-closed
  by auto
moreover have  $\ominus_P \mathbf{1}_P \neq \mathbf{0}_P$  by simp
ultimately have  $\text{degree } f \leq \text{degree } (\ominus_P \mathbf{1}_P)$ 
  using f-carr
  by (intro pdivides-imp-degree-le, simp-all add:univ-poly-zero)
also have  $\dots = 0$ 
  by (subst univ-poly-a-inv-degree, simp)
  (simp add:univ-poly-one)
finally have  $\text{degree } f = 0$  by simp

then show False
  using pirreducible-degree assms(2)
  unfolding monic-irreducible-poly-def monic-poly-def
  by fastforce
qed
ultimately have  $\text{pmult}_R f \text{ ?}g = 1$  by simp
then show ?thesis using True by simp
next
case False
have o2:  $\text{order } R^{\wedge n} > 1$ 
  using finite-field-min-order assms(1) one-less-power by blast

have  $\neg(f \text{ pdivides}_R \text{ gauss-poly } R (\text{order } R^{\wedge n}))$ 
  using div-gauss-poly-iff[OF assms] False by simp
hence  $\text{pmult}_R f (\text{gauss-poly } R (\text{order } R^{\wedge n})) = 0$ 
  using multiplicity-ge-1-iff-pdivides[OF assms(2)]
  using gauss-poly-carr gauss-poly-not-zero[OF o2] leI less-one
  by blast
then show ?thesis using False by simp
qed

```

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  $\text{order } R^{\wedge n} = (\sum d \mid d \text{ dvd } n. d * \text{card } \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = d\})$ 
(is ?lhs = ?rhs)

```

**proof** –

```

let ?G =  $\{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f \text{ dvd } n\}$ 

```

```

let ?D =  $\{f. \text{monic-irreducible-poly } R f\}$ 

```

```

have a: finite  $\{d. d \text{ dvd } n\}$  using finite-divisors-nat assms by simp

```

```

have b: finite  $\{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = k\}$  for k

```

**proof** –

```

have {f. monic-irreducible-poly R f ∧ degree f = k} ⊆
  {f. f ∈ carrier P ∧ degree f ≤ k}
  unfolding monic-irreducible-poly-def monic-poly-def by auto
moreover have finite {f. f ∈ carrier P ∧ degree f ≤ k}
  using finite-poly[OF finite-carrier] by simp
ultimately show ?thesis using finite-subset by simp
qed

have G-split: ?G =
  ∪ {f. monic-irreducible-poly R f ∧ degree f = d} | d. d dvd n}
  by auto
have c: finite ?G
  using a b by (subst G-split, auto)
have d: order  $R^{\wedge n} > 1$ 
  using assms finite-field-min-order one-less-power by blast

have ?lhs = degree (gauss-poly R (order  $R^{\wedge n}$ ))
  using d
  by (subst gauss-poly-degree, simp-all)
also have ... =
  sum' (λd. pmult R d (gauss-poly R (order  $R^{\wedge n}$ )) * degree d) ?D
  using d
  by (intro degree-monic-poly'[symmetric] gauss-poly-monic)
also have ... = sum' (λd. of-bool (degree d dvd n) * degree d) ?D
  using multiplicity-of-factor-of-gauss-poly[OF assms]
  by (intro sum.cong', auto)
also have ... = sum' (λd. degree d) ?G
  by (intro sum.mono-neutral-cong-right' subsetI, auto)
also have ... = (∑ d ∈ ?G. degree d)
  using c by (intro sum.eq-sum, simp)
also have ... =
  (∑ f ∈ (∪ d ∈ {d. d dvd n}.
  {f. monic-irreducible-poly R f ∧ degree f = d}). degree f)
  by (intro sum.cong, auto simp add:set-eq-iff)
also have ... = (∑ d | d dvd n. sum degree
  {f. monic-irreducible-poly R f ∧ degree f = d})
  using a b by (subst sum.UNION-disjoint, auto simp add:set-eq-iff)
also have ... = (∑ d | d dvd n. sum (λ-. d)
  {f. monic-irreducible-poly R f ∧ degree f = d})
  by (intro sum.cong, simp-all)
also have ... = ?rhs
  by (simp add:mult commute)
finally show ?thesis
  by simp
qed

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  $n * \text{card} \{f. \text{monic-irreducible-poly } R \ f \ \wedge \ \text{degree } f = n\} =$ 
     $(\sum d \mid d \ \text{dvd} \ n. \text{moebius-mu } d * (\text{order } R ^{(n \ \text{div} \ d})))$ 
    (is ?lhs = ?rhs)
proof –
  have ?lhs = dirichlet-prod moebius-mu  $(\lambda x. \text{int } (\text{order } R) ^ x) \ n$ 
    using card-irred-aux
    by (intro moebius-inversion assms) (simp flip:of-nat-power)
  also have  $\dots = \text{?rhs}$ 
    by (simp add:dirichlet-prod-def)
  finally show ?thesis by simp
qed

```

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 –
  have  $\text{int } (1 * \text{card} \{f. \text{monic-irreducible-poly } R \ f \ \wedge \ \text{degree } f = 1\})$ 
     $= \text{int } (\text{order } R)$ 
    by (subst card-irred, auto)
  thus ?thesis by simp
qed

```

```

lemma (in finite-field) card-irred-2:

```

$real (card \{f. monic-irreducible-poly R f \wedge degree f = 2\}) =$   
 $(real (order R)^2 - order R) / 2$

**proof** –

**have**  $x \text{ dvd } 2 \implies x = 1 \vee x = 2$  **for**  $x :: nat$   
**using** *nat-dvd-not-less*[**where**  $m=2$ ]  
**by** (*metis One-nat-def even-zero gcd-nat.strict-trans2*  
*less-2-cases nat-neq-iff pos2*)  
**hence**  $a: \{d. d \text{ dvd } 2\} = \{1, 2 :: nat\}$   
**by** (*auto simp add:set-eq-iff*)

**have**  $2 * real (card \{f. monic-irreducible-poly R f \wedge degree f = 2\})$   
 $= of-int (2 * card \{f. monic-irreducible-poly R f \wedge degree f = 2\})$   
**by** *simp*  
**also have**  $... =$   
 $of-int (\sum d \mid d \text{ dvd } 2. moebius-mu d * int (order R) ^ (2 \text{ div } d))$   
**by** (*subst card-irred, auto*)  
**also have**  $... = order R^2 - int (order R)$   
**by** (*subst a, simp*)  
**also have**  $... = real (order R)^2 - order R$   
**by** *simp*  
**finally have**  
 $2 * real (card \{f. monic-irreducible-poly R f \wedge degree f = 2\}) =$   
 $real (order R)^2 - order R$   
**by** *simp*  
**thus** *?thesis* **by** *simp*

**qed**

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

**proof** –

**let**  $?m = real (order R)$   
**have**  $a: ?m \geq 2$   
**using** *finite-field-min-order* **by** *simp*

**have**  $b: moebius-mu n \geq -(1 :: real)$  **for**  $n :: nat$   
**using** *abs-moebius-mu-le*[**where**  $n=n$ ]  
**unfolding** *abs-le-iff* **by** *auto*

**have**  $c: n > 0$  **using** *assms* **by** *simp*  
**have**  $d: x < n - 1$  **if**  $d\text{-assms}: x \text{ dvd } n \ x \neq n$  **for**  $x :: nat$

**proof** –

**have**  $x < n$   
**using** *d-assms dvd-nat-bounds c* **by** *auto*  
**moreover have**  $\neg(n-1 \text{ dvd } n)$  **using** *assms*  
**by** (*metis One-nat-def Suc-diff-Suc c diff-zero*  
*dvd-add-triv-right-iff nat-dvd-1-iff-1*)

*nat-neq-iff numeral-2-eq-2 plus-1-eq-Suc*  
**hence**  $x \neq n-1$  **using** *d-assms* **by** *auto*  
**ultimately show**  $x < n-1$  **by** *simp*  
**qed**

**have**  $?m \hat{n} / 2 = ?m \hat{n} - ?m \hat{n} / 2$  **by** *simp*  
**also have**  $\dots \leq ?m \hat{n} - ?m \hat{n} / ?m \hat{1}$   
**using** *a* **by** (*intro diff-mono divide-left-mono, simp-all*)  
**also have**  $\dots \leq ?m \hat{n} - ?m \hat{(n-1)}$   
**using** *a c* **by** (*subst power-diff, simp-all*)  
**also have**  $\dots \leq ?m \hat{n} - (?m \hat{(n-1)} - 1) / 1$  **by** *simp*  
**also have**  $\dots \leq ?m \hat{n} - (?m \hat{(n-1)} - 1) / (?m - 1)$   
**using** *a* **by** (*intro diff-left-mono divide-left-mono, simp-all*)  
**also have**  $\dots = ?m \hat{n} - (\sum i \in \{..<n-1\}. ?m \hat{i})$   
**using** *a* **by** (*subst geometric-sum, simp-all*)  
**also have**  $\dots \leq ?m \hat{n} - (\sum i \in \{k. k \text{ dvd } n \wedge k \neq n\}. ?m \hat{i})$   
**using** *d*  
**by** (*intro diff-mono sum-mono2 subsetI, auto simp add:not-less*)  
**also have**  $\dots = ?m \hat{n} + (\sum i \in \{k. k \text{ dvd } n \wedge k \neq n\}. (-1) * ?m \hat{i})$   
**by** (*subst sum-distrib-left[symmetric], simp*)  
**also have**  $\dots \leq \text{moebius-mu } 1 * ?m \hat{n} +$   
 $(\sum i \in \{k. k \text{ dvd } n \wedge k \neq n\}. \text{moebius-mu } (n \text{ div } i) * ?m \hat{i})$   
**using** *b*  
**by** (*intro add-mono sum-mono mult-right-mono*)  
*(simp-all add:not-less)*  
**also have**  $\dots = (\sum i \in \text{insert } n \{k. k \text{ dvd } n \wedge k \neq n\}.$   
 $\text{moebius-mu } (n \text{ div } i) * ?m \hat{i})$   
**using** *c* **by** (*subst sum.insert, auto*)  
**also have**  $\dots = (\sum i \in \{k. k \text{ dvd } n\}. \text{moebius-mu } (n \text{ div } i) * ?m \hat{i})$   
**by** (*intro sum.cong, auto simp add:set-eq-iff*)  
**also have**  $\dots = \text{dirichlet-prod } (\lambda i. ?m \hat{i}) \text{ moebius-mu } n$   
**unfolding** *dirichlet-prod-def* **by** (*intro sum.cong, auto*)  
**also have**  $\dots = \text{dirichlet-prod moebius-mu } (\lambda i. ?m \hat{i}) n$   
**using** *dirichlet-prod-commutes* **by** *metis*  
**also have**  $\dots =$   
 $\text{of-int } (\sum d \mid d \text{ dvd } n. \text{moebius-mu } d * \text{order } R \hat{(n \text{ div } d)})$   
**unfolding** *dirichlet-prod-def* **by** *simp*  
**also have**  $\dots = \text{of-int } (n *$   
 $\text{card } \{f. \text{monic-irreducible-poly } R f \wedge \text{length } f - 1 = n\})$   
**using** *card-irred[OF c]* **by** *simp*  
**also have**  $\dots = n * ?rhs$  **by** *simp*  
**finally have**  $?m \hat{n} / 2 \leq n * ?rhs$  **by** *simp*  
**hence**  $?m \hat{n} \leq 2 * n * ?rhs$  **by** *simp*  
**hence**  $?m \hat{n} / (2 * \text{real } n) \leq ?rhs$   
**using** *c* **by** (*subst pos-divide-le-eq, simp-all add:algebra-simps*)  
**thus** *thesis* **by** *simp*  
**qed**

lemma (in *finite-field*) *card-irred-gt-0*:

**assumes**  $d > 0$   
**shows**  $\text{real}(\text{order } R)^{\wedge} d / (2 * \text{real } d) \leq \text{real}(\text{card } \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = d\})$   
**(is**  $?L \leq ?R$ )  
**proof** –  
**consider**  $(a) d = 1 \mid (b) d = 2 \mid (c) d > 2$  **using** *assms* **by** *linarith*  
**thus** *?thesis*  
**proof** (*cases*)  
**case** *a*  
**hence**  $?L = \text{real}(\text{order } R)/2$  **by** *simp*  
**also have**  $\dots \leq \text{real}(\text{order } R)$  **using** *finite-field-min-order* **by** *simp*  
**also have**  $\dots = ?R$  **unfolding** *a card-irred-1* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**next**  
**case** *b*  
**hence**  $?L = \text{real}(\text{order } R^{\wedge} 2)/4 + 0$  **by** *simp*  
**also have**  $\dots \leq \text{real}(\text{order } R^{\wedge} 2)/4 + \text{real}(\text{order } R)/2 * (\text{real}(\text{order } R)/2 - 1)$   
**using** *finite-field-min-order* **by** (*intro add-mono mult-nonneg-nonneg*)  
*auto*  
**also have**  $\dots = (\text{real}(\text{order } R^{\wedge} 2) - \text{real}(\text{order } R))/2$   
**by** (*simp add:algebra-simps power2-eq-square*)  
**also have**  $\dots = ?R$  **unfolding** *b card-irred-2* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**next**  
**case** *c* **thus** *?thesis* **by** (*rule card-irred-gt-2*)  
**qed**  
**qed**

**lemma** (*in finite-field*) *exist-irred*:  
**assumes**  $n > 0$   
**obtains**  $f$  **where** *monic-irreducible-poly*  $R f$  *degree*  $f = n$   
**proof** –  
**have**  $0 < \text{real}(\text{order } R)^{\wedge} n / (2 * \text{real } n)$   
**using** *finite-field-min-order* *assms*  
**by** (*intro divide-pos-pos mult-pos-pos zero-less-power*) *auto*  
**also have**  $\dots \leq \text{real}(\text{card } \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = n\})$   
**(is**  $\dots \leq \text{real}(\text{card } ?A)$ )  
**by** (*intro card-irred-gt-0* *assms*)  
**finally have**  $0 < \text{card } \{f. \text{monic-irreducible-poly } R f \wedge \text{degree } f = n\}$   
**by** *auto*  
**hence**  $?A \neq \{\}$   
**by** (*metis card.empty nless-le*)  
**then obtain**  $f$  **where** *monic-irreducible-poly*  $R f$  *degree*  $f = n$   
**by** *auto*  
**thus** *?thesis* **using** *that* **by** *simp*  
**qed**

```

theorem existence:
  assumes  $n > 0$ 
  assumes Factorial-Ring.prime p
  shows  $\exists (F:: \text{int set list set ring}). \text{finite-field } F \wedge \text{order } F = p^{\wedge}n$ 
proof –
  interpret zf: finite-field ZFact (int p)
    using zfact-prime-is-finite-field assms by simp

  interpret zfp: polynomial-ring ZFact p carrier (ZFact p)
    unfolding polynomial-ring-def polynomial-ring-axioms-def
    using zf.field-axioms zf.carrier-is-subfield by simp

  have p-gt-0: p > 0 using prime-gt-0-nat assms(2) by simp

  obtain f where f-def:
    monic-irreducible-poly (ZFact (int p)) f
    degree f = n
    using zf.exist-irred assms by auto

  let ?F = Rupt(ZFact p) (carrier (ZFact p)) f
  have f ∈ carrier (poly-ring (ZFact (int p)))
    using f-def(1) zf.monic-poly-carr
    unfolding monic-irreducible-poly-def
    by simp
  moreover have degree f > 0
    using assms(1) f-def by simp
  ultimately have order ?F = card (carrier (ZFact p))^degree f
    by (intro zf.rupture-order[OF zf.carrier-is-subfield]) auto
  hence a:order ?F = p^{\wedge}n
    unfolding f-def(2) card-zfact-carr[OF p-gt-0] by simp

  have field ?F
    using f-def(1) zf.monic-poly-carr monic-irreducible-poly-def
    by (subst zfp.rupture-is-field-iff-pirreducible) auto
  moreover have order ?F > 0
    unfolding a using assms(1,2) p-gt-0 by simp
  ultimately have b:finite-field ?F
    using card-ge-0-finite
    by (intro finite-fieldI, auto simp add:Coset.order-def)

  show ?thesis
    using a b
    by (intro exI[where x=?F], simp)
qed

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  $\text{order } R = p^{\widehat{\text{degree } f}}$ 
  assumes  $x \in \text{carrier } R$ 
  assumes  $\text{eval } (\text{map } (\text{char-iso } R) f) x = \mathbf{0}$ 
  shows  $\text{ring-hom-ring } (\text{Rupt}_{(\text{ZFact } p)} (\text{carrier } (\text{ZFact } p)) f) R$ 
     $(\lambda g. \text{the-elem } ((\lambda g'. \text{eval } (\text{map } (\text{char-iso } R) g') x) 'g))$ 
proof –
  let  $?P = \text{poly-ring } (\text{ZFact } p)$ 

  have char-pos:  $\text{char } R > 0$ 
    using finite-carr-imp-char-ge-0[OF finite-carrier] by simp

  have p-prime: Factorial-Ring.prime  $p$ 
    unfolding p-def
    using characteristic-is-prime[OF char-pos] by simp

  interpret zf: finite-field  $\text{ZFact } p$ 
    using zfact-prime-is-finite-field p-prime by simp
  interpret pzf: principal-domain poly-ring  $(\text{ZFact } p)$ 
    using zf.univ-poly-is-principal[OF zf.carrier-is-subfield] by simp

  interpret i: ideal  $(\text{PIdl } ?P f) ?P$ 
    by (intro pzf.cgenideal-ideal assms(2))
  have rupt-carr:  $y \subseteq \text{carrier } (\text{poly-ring } (\text{ZFact } p))$ 
    if  $y \in \text{carrier } (\text{Rupt } \text{ZFact } p (\text{carrier } (\text{ZFact } p)) f)$  for  $y$ 
    using that pzf.quot-carr i.ideal-axioms by (simp add:rupture-def)

  have rupt-is-ring: ring  $(\text{Rupt } \text{ZFact } p (\text{carrier } (\text{ZFact } p)) f)$ 
    unfolding rupture-def by (intro i.quotient-is-ring)

  have  $\text{map } (\text{char-iso } R) \in$ 
    ring-iso  $?P (\text{poly-ring } (R \langle \text{carrier } := \text{char-subring } R \rangle))$ 
    using lift-iso-to-poly-ring[OF char-iso] zf.domain-axioms
    using char-ring-is-subdomain subdomain-is-domain
    by (simp add:p-def)
  moreover have  $(\text{char-subring } R)[X] =$ 
    poly-ring  $(R \langle \text{carrier } := \text{char-subring } R \rangle)$ 
    using univ-poly-consistent[OF char-ring-is-subring] by simp
  ultimately have

```

```

    map (char-iso R) ∈ ring-hom ?P ((char-subring R)[X])
  by (simp add:ring-iso-def)
moreover have (λp. eval p x) ∈ ring-hom ((char-subring R)[X]) R
  using eval-is-hom char-ring-is-subring assms(5) by simp
ultimately have
  (λp. eval p x) ∘ map (char-iso R) ∈ ring-hom ?P R
  using ring-hom-trans by blast
hence a:(λp. eval (map (char-iso R) p) x) ∈ ring-hom ?P R
  by (simp add:comp-def)
interpret h:ring-hom-ring ?P R (λp. eval (map (char-iso R) p) x)
  by (intro ring-hom-ringI2 pzf.ring-axioms a ring-axioms)

let ?h = (λp. eval (map (char-iso R) p) x)
let ?J = a-kernel (poly-ring (ZFact (int p))) R ?h

have ?h ‘ a-kernel (poly-ring (ZFact (int p))) R ?h ⊆ {0}
  by auto
moreover have
  0_{?P} ∈ a-kernel (poly-ring (ZFact (int p))) R ?h
  ?h 0_{?P} = 0
  unfolding a-kernel-def' by simp-all
hence {0} ⊆ ?h ‘ a-kernel (poly-ring (ZFact (int p))) R ?h
  by simp
ultimately have c:
  ?h ‘ a-kernel (poly-ring (ZFact (int p))) R ?h = {0}
  by auto

have d: PIdl_{?P} f ⊆ a-kernel ?P R ?h
proof (rule subsetI)
  fix y assume y ∈ PIdl_{?P} f
  then obtain y' where y'-def: y' ∈ carrier ?P y = y' ⊗_{?P} f
  unfolding cgenideal-def by auto
  have ?h y = ?h (y' ⊗_{?P} f) by (simp add:y'-def)
  also have ... = ?h y' ⊗ ?h f
  using y'-def assms(2) by simp
  also have ... = ?h y' ⊗ 0
  using assms(6) by simp
  also have ... = 0
  using y'-def by simp
  finally have ?h y = 0 by simp
  moreover have y ∈ carrier ?P using y'-def assms(2) by simp
  ultimately show y ∈ a-kernel ?P R ?h
  unfolding a-kernel-def kernel-def by simp
qed

have (λy. the-elem ((λp. eval (map (char-iso R) p) x) ‘ y))
  ∈ ring-hom (?P Quot ?J) R
  using h.the-elem-hom by simp
moreover have (λy. ?J <+>_{?P} y)

```

$\in \text{ring-hom } (\text{Rupt}_{(\text{ZFact } p)} (\text{carrier } (\text{ZFact } p)) f) (?P \text{ Quot } ?J)$   
**unfolding** *rupture-def* **using** *h.kernel-is-ideal d assms(2)*  
**by** (*intro pzf.quot-quot-hom pzf.cgenideal-ideal*) *auto*  
**ultimately have**  $(\lambda y. \text{the-elem } (?h \text{ ' } y)) \circ (\lambda y. ?J \langle + \rangle ?P y)$   
 $\in \text{ring-hom } (\text{Rupt}_{(\text{ZFact } p)} (\text{carrier } (\text{ZFact } p)) f) R$   
**using** *ring-hom-trans* **by** *blast*  
**hence**  $b: (\lambda y. \text{the-elem } (?h \text{ ' } (?J \langle + \rangle ?P y))) \in$   
 $\text{ring-hom } (\text{Rupt}_{(\text{ZFact } p)} (\text{carrier } (\text{ZFact } p)) f) R$   
**by** (*simp add:comp-def*)  
**have**  $?h \text{ ' } y = ?h \text{ ' } (?J \langle + \rangle ?P y)$   
**if**  $y \in \text{carrier } (\text{Rupt}_{\text{ZFact } p} (\text{carrier } (\text{ZFact } p)) f)$   
**for**  $y$   
**proof** –  
**have**  $y\text{-range: } y \subseteq \text{carrier } ?P$   
**using** *rupt-carr that* **by** *simp*  
**have**  $?h \text{ ' } y = \{0\} \langle + \rangle_R ?h \text{ ' } y$   
**using**  $y\text{-range}$  *h.hom-closed* **by** (*subst set-add-zero, auto*)  
**also have**  $\dots = ?h \text{ ' } ?J \langle + \rangle_R ?h \text{ ' } y$   
**by** (*subst c, simp*)  
**also have**  $\dots = ?h \text{ ' } (?J \langle + \rangle ?P y)$   
**by** (*subst set-add-hom[OF a - y-range], subst a-kernel-def'*) *auto*  
**finally show** *?thesis* **by** *simp*  
**qed**  
**hence**  $(\lambda y. \text{the-elem } (?h \text{ ' } y)) \in$   
 $\text{ring-hom } (\text{Rupt}_{(\text{ZFact } p)} (\text{carrier } (\text{ZFact } p)) f) R$   
**by** (*intro ring-hom-cong[OF - rupt-is-ring b]*) *simp*  
**thus** *?thesis*  
**by** (*intro ring-hom-ringI2 rupt-is-ring ring-axioms, simp*)  
**qed**

**lemma** (*in domain*) *pdivides-consistent*:  
**assumes** *subfield K R f*  $\in \text{carrier } (K[X])$   $g \in \text{carrier } (K[X])$   
**shows**  $f \text{ pdivides } g \longleftrightarrow f \text{ pdivides}_R (\text{carrier } := K) g$

**proof** –  
**have**  $a:\text{subring } K R$   
**using** *assms(1) subfieldE(1)* **by** *auto*  
**let**  $?S = R (\text{carrier } := K)$   
**have**  $f \text{ pdivides } g \longleftrightarrow f \text{ divides}_{K[X]} g$   
**using** *pdivides-iff-shell[OF assms]* **by** *simp*  
**also have**  $\dots \longleftrightarrow (\exists x \in \text{carrier } (K[X]). f \otimes_{K[X]} x = g)$   
**unfolding** *pdivides-def factor-def* **by** *auto*  
**also have**  $\dots \longleftrightarrow$   
 $(\exists x \in \text{carrier } (\text{poly-ring } ?S). f \otimes_{\text{poly-ring } ?S} x = g)$   
**using** *univ-poly-consistent[OF a]* **by** *simp*  
**also have**  $\dots \longleftrightarrow f \text{ divides}_{\text{poly-ring } ?S} g$   
**unfolding** *pdivides-def factor-def* **by** *auto*  
**also have**  $\dots \longleftrightarrow f \text{ pdivides}_{?S} g$   
**unfolding** *pdivides-def* **by** *simp*

**finally show** *?thesis* **by** *simp*  
**qed**

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

**assumes** *subfield*  $K\ R$   
**assumes** *monic-irreducible-poly* ( $R$  ( $\mid$  *carrier* :=  $K$   $\mid$ ))  $f$   
**assumes** *order*  $R = \text{card } K^{\wedge} \text{degree } f$   
**obtains**  $x$  **where** *eval*  $f\ x = \mathbf{0}$   $x \in \text{carrier } R$

**proof** –

**define**  $\tau :: 'a\ \text{list} \Rightarrow 'a\ \text{list}$  **where**  $\tau = \text{id}$   
**let**  $?K = R$  ( $\mid$  *carrier* :=  $K$   $\mid$ )  
**have** *finite*  $K$   
**using** *assms*(1) **by** (*intro finite-subset*[*OF* - *finite-carrier*], *simp*)  
**hence** *fin-K*: *finite* (*carrier* ( $?K$ ))  
**by** *simp*  
**interpret**  $f$ : *finite-field*  $?K$   
**using** *assms*(1) *subfield-iff fin-K finite-fieldI* **by** *blast*  
**have**  $b$ : *subring*  $K\ R$   
**using** *assms*(1) *subfieldE*(1) **by** *blast*  
**interpret**  $e$ : *ring-hom-ring* ( $K[X]$ ) (*poly-ring*  $R$ )  $\tau$   
**using** *embed-hom*[*OF*  $b$ ] **by** (*simp add*: $\tau$ -*def*)

**have**  $a$ : *card*  $K^{\wedge} \text{degree } f > 1$   
**using** *assms*(3) *finite-field-min-order* **by** *simp*  
**have**  $f \in \text{carrier}$  (*poly-ring*  $?K$ )  
**using**  $f$ .*monic-poly-carr* *assms*(2)  
**unfolding** *monic-irreducible-poly-def* **by** *simp*  
**hence**  $f$ -*carr-2*:  $f \in \text{carrier}$  ( $K[X]$ )  
**using** *univ-poly-consistent*[*OF*  $b$ ] **by** *simp*  
**have**  $f$ -*carr*:  $f \in \text{carrier}$  (*poly-ring*  $R$ )  
**using**  $e$ .*hom-closed*[*OF*  $f$ -*carr-2*] **unfolding**  $\tau$ -*def* **by** *simp*

**have**  $gp$ -*carr*: *gauss-poly*  $?K$  (*order*  $?K^{\wedge} \text{degree } f$ )  $\in \text{carrier}$  ( $K[X]$ )  
**using**  $f$ .*gauss-poly-carr univ-poly-consistent*[*OF*  $b$ ] **by** *simp*

**have** *gauss-poly*  $?K$  (*order*  $?K^{\wedge} \text{degree } f$ ) =  
*gauss-poly*  $?K$  (*card*  $K^{\wedge} \text{degree } f$ )  
**by** (*simp add*:*Coset.order-def*)

**also have** ... =

$X_{?K} [\bigwedge]_{\text{poly-ring } ?K} \text{card } K^{\wedge} \text{degree } f \ominus_{\text{poly-ring } ?K} X_{?K}$   
**unfolding** *gauss-poly-def* **by** *simp*

**also have** ... =  $X_R [\bigwedge]_{K[X]} \text{card } K^{\wedge} \text{degree } f \ominus_{K[X]} X_R$

**unfolding** *var-def* **using** *univ-poly-consistent*[*OF*  $b$ ] **by** *simp*

**also have** ... =  $\tau (X_R [\bigwedge]_{K[X]} \text{card } K^{\wedge} \text{degree } f \ominus_{K[X]} X_R)$

**unfolding**  $\tau$ -*def* **by** *simp*

**also have** ... = *gauss-poly*  $R$  (*card*  $K^{\wedge} \text{degree } f$ )

**unfolding** *gauss-poly-def a-minus-def* **using** *var-closed*[*OF*  $b$ ]

**by** (*simp add*: $e$ .*hom-nat-pow*, *simp add*: $\tau$ -*def*)

**finally have**  $gp$ -*consistent*: *gauss-poly*  $?K$  (*order*  $?K^{\wedge} \text{degree } f$ ) =

```

    gauss-poly R (card K ^ degree f)
    by simp

have deg-f: degree f > 0
  using f.monic-poly-min-degree[OF assms(2)] by simp

have splitted f
proof (cases degree f > 1)
  case True

  have f pdivides ?K gauss-poly ?K (order ?K ^ degree f)
    using f.div-gauss-poly-iff[OF deg-f assms(2)] by simp
  hence f pdivides gauss-poly ?K (order ?K ^ degree f)
    using pdivides-consistent[OF assms(1)] f-carr-2 gp-carr by simp
  hence f pdivides gauss-poly R (card K ^ degree f)
    using gp-consistent by simp
  moreover have splitted (gauss-poly R (card K ^ degree f))
    unfolding assms(3)[symmetric] using gauss-poly-splitted by simp
  moreover have gauss-poly R (card K ^ degree f) ≠ []
    using gauss-poly-not-zero a by (simp add: univ-poly-zero)
  ultimately show splitted f
    using pdivides-imp-splitted f-carr gauss-poly-carr by auto
  next
  case False
  hence degree f = 1 using deg-f by simp
  thus ?thesis using f-carr degree-one-imp-splitted by auto
qed
hence size (roots f) > 0
  using deg-f unfolding splitted-def by simp
then obtain x where x-def: x ∈ carrier R is-root f x
  using roots-mem-iff-is-root[OF f-carr]
  by (metis f-carr nonempty-has-size not-empty-rootsE)
have eval f x = 0
  using x-def is-root-def by blast
thus ?thesis using x-def using that by simp
qed

lemma (in finite-field) find-iso-from-zfact:
  defines p ≡ int (char R)
  assumes monic-irreducible-poly (ZFact p) f
  assumes order R = char R ^ degree f
  shows ∃ φ. φ ∈ ring-iso (Rupt (ZFact p) (carrier (ZFact p))) f) R
proof -
  have char-pos: char R > 0
    using finite-carr-imp-char-ge-0[OF finite-carrier] by simp

  interpret zf: finite-field ZFact p
    unfolding p-def using zfact-prime-is-finite-field
    using characteristic-is-prime[OF char-pos] by simp

```

```

interpret zfp: polynomial-ring ZFact p carrier (ZFact p)
  unfolding polynomial-ring-def polynomial-ring-axioms-def
  using zf.field-axioms zf.carrier-is-subfield by simp

let ?f' = map (char-iso R) f
let ?F = Rupt(ZFact p) (carrier (ZFact p)) f

have domain (R⟦carrier := char-subring R⟧)
  using char-ring-is-subdomain subdomain-is-domain by simp

hence monic-irreducible-poly (R⟦carrier := char-subring R⟧) ?f'
  using char-iso p-def zf.domain-axioms
  by (intro monic-irreducible-poly-hom[OF assms(2)]) auto
moreover have order R = card (char-subring R) ^ degree ?f'
  using assms(3) unfolding char-def by simp
ultimately obtain x where x-def: eval ?f' x = 0 x ∈ carrier R
  using find-root[OF char-ring-is-subfield[OF char-pos]] by blast
let ?φ = (λg. the-elem ((λg'. eval (map (char-iso R) g') x) ' g))
interpret r: ring-hom-ring ?F R ?φ
  using assms(2,3)
  unfolding monic-irreducible-poly-def monic-poly-def p-def
  by (intro eval-on-root-is-iso x-def, auto)
have a: ?φ ∈ ring-hom ?F R
  using r.homh by auto

have field (RuptZFact p (carrier (ZFact p)) f)
  using assms(2)
  unfolding monic-irreducible-poly-def monic-poly-def
  by (subst zfp.rupture-is-field-iff-pirreducible, simp-all)
hence b: inj-on ?φ (carrier ?F)
  using non-trivial-field-hom-is-inj[OF a - field-axioms] by simp

have card (?φ ' carrier ?F) = order ?F
  using card-image[OF b] unfolding Coset.order-def by simp
also have ... = card (carrier (ZFact p)) ^ degree f
  using assms(2) zf.monic-poly-min-degree[OF assms(2)]
  unfolding monic-irreducible-poly-def monic-poly-def
  by (intro zf.rupture-order[OF zf.carrier-is-subfield]) auto
also have ... = char R ^ degree f
  unfolding p-def by (subst card-zfact-carr[OF char-pos], simp)
also have ... = card (carrier R)
  using assms(3) unfolding Coset.order-def by simp
finally have card (?φ ' carrier ?F) = card (carrier R) by simp
moreover have ?φ ' carrier ?F ⊆ carrier R
  by (intro image-subsetI, simp)
ultimately have ?φ ' carrier ?F = carrier R
  by (intro card-seteq finite-carrier, auto)
hence bij-betw ?φ (carrier ?F) (carrier R)

```

```

using b bij-betw-imageI by auto

thus ?thesis
  unfolding ring-iso-def using a b by auto
qed

theorem uniqueness:
  assumes finite-field F1
  assumes finite-field F2
  assumes order F1 = order F2
  shows F1 ≃ F2
proof -
  obtain n where o1: order F1 = char F1 ^ n n > 0
    using finite-field.finite-field-order[OF assms(1)] by auto
  obtain m where o2: order F2 = char F2 ^ m m > 0
    using finite-field.finite-field-order[OF assms(2)] by auto

  interpret f1: finite-field F1 using assms(1) by simp
  interpret f2: finite-field F2 using assms(2) by simp

  have char-pos: char F1 > 0 char F2 > 0
    using f1.finite-carrier f1.finite-carr-imp-char-ge-0
    using f2.finite-carrier f2.finite-carr-imp-char-ge-0 by auto
  hence char-prime:
    Factorial-Ring.prime (char F1)
    Factorial-Ring.prime (char F2)
    using f1.characteristic-is-prime f2.characteristic-is-prime
    by auto

  have char F1 ^ n = char F2 ^ m
    using o1 o2 assms(3) by simp
  hence eq: n = m char F1 = char F2
    using char-prime char-pos o1(2) o2(2) prime-power-inj' by auto

  obtain p where p-def: p = char F1 p = char F2
    using eq by simp

  have p-prime: Factorial-Ring.prime p
    unfolding p-def(1)
    using f1.characteristic-is-prime char-pos by simp

  interpret zf: finite-field ZFact (int p)
    using zfact-prime-is-finite-field p-prime o1(2)
    using prime-nat-int-transfer by blast

  interpret zfp: polynomial-ring ZFact p carrier (ZFact p)
    unfolding polynomial-ring-def polynomial-ring-axioms-def
    using zf.field-axioms zf.carrier-is-subfield by simp

```

```

obtain  $f$  where  $f$ -def:
  monic-irreducible-poly (ZFact (int  $p$ ))  $f$  degree  $f = n$ 
  using  $zf$ .exist-irred o1(2) by auto

let  $?F_0 = \text{Rupt}_{(ZFact\ p)} (\text{carrier } (ZFact\ p))\ f$ 

obtain  $\varphi_1$  where  $\varphi_1$ -def:  $\varphi_1 \in \text{ring-iso } ?F_0\ F_1$ 
  using  $f1$ .find-iso-from-zfact  $f$ -def o1
  unfolding  $p$ -def by auto

obtain  $\varphi_2$  where  $\varphi_2$ -def:  $\varphi_2 \in \text{ring-iso } ?F_0\ F_2$ 
  using  $f2$ .find-iso-from-zfact  $f$ -def o2
  unfolding  $p$ -def(2) eq(1) by auto

have  $?F_0 \simeq F_1$  using  $\varphi_1$ -def is-ring-iso-def by auto
moreover have  $?F_0 \simeq F_2$  using  $\varphi_2$ -def is-ring-iso-def by auto
moreover have field  $?F_0$ 
  using  $f$ -def(1)  $zf$ .monic-poly-carr monic-irreducible-poly-def
  by (subst  $zfp$ .rupture-is-field-iff-pirreducible) auto
hence ring  $?F_0$  using field.is-ring by auto
ultimately show  $?thesis$ 
  using ring-iso-trans ring-iso-sym by blast
qed

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

```

```

lemma  $pcoprimeI$ :
  assumes  $\bigwedge r. r \in \text{carrier } (poly\text{-ring } R) \implies r\ pdivides\ R\ p \implies r$ 
   $pdivides_R\ q \implies \text{degree } r = 0$ 
  shows  $pcoprime_R\ p\ q$ 
  using  $assms$  unfolding  $pcoprime$ -def by auto

```

```

context field
begin

```

```

interpretation  $r$ :polynomial-ring  $R$  (carrier  $R$ )

```

**unfolding** *polynomial-ring-def polynomial-ring-axioms-def*  
**using** *carrier-is-subfield field-axioms* **by force**

**lemma** *pcoprime-one*:  $pcoprime_R p \mathbf{1}_{poly\text{-ring } R}$   
**proof** (*rule pcoprimeI*)  
**fix**  $r$   
**assume**  $r\text{-carr}$ :  $r \in carrier (poly\text{-ring } R)$   
**moreover assume**  $r$  *pdivides*  $R \mathbf{1}_{poly\text{-ring } R}$   
**moreover have**  $\mathbf{1}_{poly\text{-ring } R} \neq []$  **by** (*simp add:univ-poly-one*)  
**ultimately have**  $degree\ r \leq degree \mathbf{1}_{poly\text{-ring } R}$   
**by** (*intro pdivides-imp-degree-le[OF carrier-is-subring] r-carr*) *auto*  
**also have**  $\dots = 0$  **by** (*simp add:univ-poly-one*)  
**finally show**  $degree\ r = 0$  **by auto**  
**qed**

**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** (*rule pcoprimeI*)  
**fix**  $r$   
**assume**  $r\text{-carr}$ :  $r \in carrier (poly\text{-ring } R)$   
**assume**  $r$  *pdivides*  $R x$   
**hence**  $r$  *pdivides*  $R (x \otimes_{poly\text{-ring } R} y)$   
**using** *assms(1,2) r-carr r.p.divides-prod-r* **unfolding** *pdivides-def*  
**by simp**  
**moreover assume**  $r$  *pdivides*  $R z$   
**ultimately show**  $degree\ r = 0$  **using** *assms(4) r-carr* **unfolding**  
*pcoprime-def* **by simp**  
**qed**

**lemma** *pcoprime-sym*:  
**shows**  $pcoprime\ x\ y = pcoprime\ y\ x$   
**unfolding** *pcoprime-def* **by auto**

**lemma** *pcoprime-left-assoc-cong-aux*:  
**assumes**  $x1 \in carrier (poly\text{-ring } R)$   $x2 \in carrier (poly\text{-ring } R)$   
**assumes**  $x2 \sim_{poly\text{-ring } R} x1$   
**assumes**  $y \in carrier (poly\text{-ring } R)$   
**assumes**  $pcoprime\ x1\ y$   
**shows**  $pcoprime\ x2\ y$   
**using** *assms r.p.divides-cong-r[OF - assms(3)]* **unfolding** *pcoprime-def*  
*pdivides-def* **by simp**

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

**assumes**  $y \in \text{carrier } (\text{poly-ring } R)$   
**shows**  $\text{pcoprime } x1 \ y = \text{pcoprime } x2 \ y$   
**using** *assms p coprime-left-assoc-cong-aux r.p.associated-sym* **by** *metis*

**lemma** *p coprime-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**  $\text{pcoprime } y \ x1 = \text{pcoprime } y \ x2$   
**using** *assms p coprime-sym p coprime-left-assoc-cong* **by** *metis*

**lemma** *p coprime-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** –  
**have**  $d \ \text{pdivides } f \longleftrightarrow d \ \text{pdivides } (f \ \text{pmod } g)$  **if**  $d \in \text{carrier } (\text{poly-ring } R)$  **d pdivides g for d**  
**proof** –  
**have**  $d \ \text{pdivides } f \longleftrightarrow d \ \text{pdivides } (g \otimes_{r.P} (f \ \text{pdiv } g) \oplus_{r.P} (f \ \text{pmod } g))$   
**using** *pdiv-pmod[OF carrier-is-subfield assms]* **by** *simp*  
**also have**  $\dots \longleftrightarrow d \ \text{pdivides } ((f \ \text{pmod } g))$   
**using** *that assms long-division-closed[OF carrier-is-subfield]* *r.p.divides-prod-r*  
**unfolding** *pdivides-def* **by** (*intro r.p.div-sum-iff*) *simp-all*  
**finally show** *?thesis* **by** *simp*  
**qed**  
**hence**  $d \ \text{pdivides } f \wedge d \ \text{pdivides } g \longleftrightarrow d \ \text{pdivides } g \wedge d \ \text{pdivides } (f \ \text{pmod } g)$   
**if**  $d \in \text{carrier } (\text{poly-ring } R)$  **for d**  
**using** *that* **by** *auto*  
**thus** *?thesis*  
**unfolding** *p coprime-def* **by** *auto*  
**qed**

**lemma** *p coprime-zero-iff*:  
**assumes**  $f \in \text{carrier } (\text{poly-ring } R)$   
**shows**  $\text{pcoprime } f \ [] \longleftrightarrow \text{length } f = 1$   
**proof** –  
**consider** *(i) length f = 0 | (ii) length f = 1 | (iii) length f > 1*  
**by** *linarith*  
**thus** *?thesis*  
**proof** (*cases*)  
**case** *i*  
**hence**  $f = []$  **by** *simp*  
**moreover have**  $X \ \text{pdivides } []$  **using** *r.pdivides-zero r.var-closed(1)*  
**by** *blast*  
**moreover have**  $\text{degree } X = 1$  **using** *degree-var* **by** *simp*  
**ultimately have**  $\neg \text{pcoprime } f \ []$  **using** *r.var-closed(1)* **unfolding**

```

pcoprime-def by auto
  then show ?thesis using i by auto
next
case ii
hence  $f \neq 0$  degree  $f = 0$  by auto
hence degree  $d = 0$  if  $d$  pdivides  $f$   $d \in \text{carrier } (\text{poly-ring } R)$  for  $d$ 
  using that(1) pdivides-imp-degree-le[OF carrier-is-subring that(2)
assms] by simp
hence pcoprime  $f$  by unfolding pcoprime-def by auto
then show ?thesis using ii by simp
next
case iii
have  $f$  pdivides  $f$  using assms unfolding pdivides-def by simp
moreover have  $f$  pdivides  $0$  using assms r.pdivides-zero by blast
moreover have degree  $f > 0$  using iii by simp
  ultimately have  $\neg$ pcoprime  $f$  by using assms unfolding pcoprime-def by auto
  then show ?thesis using iii by auto
qed
qed
end

```

```

context finite-field
begin

```

```

interpretation r:polynomial-ring R (carrier R)
  unfolding polynomial-ring-def polynomial-ring-axioms-def
  using carrier-is-subfield field-axioms by force

```

```

lemma exists-irreducible-proper-factor:

```

```

  assumes monic-poly  $R$   $f$  degree  $f > 0$   $\neg$ monic-irreducible-poly  $R$   $f$ 
  shows  $\exists g.$  monic-irreducible-poly  $R$   $g \wedge g$  pdivides $_R$   $f \wedge$  degree  $g <$ 
degree  $f$ 

```

```

proof -

```

```

  define  $S$  where  $S = \{d.$  monic-irreducible-poly  $R$   $d \wedge 0 <$  pmult  $d$ 
 $f\}$ 

```

```

  have  $f$ -carr:  $f \in \text{carrier } (\text{poly-ring } R)$   $f \neq \mathbf{0}_{\text{poly-ring } R}$ 
  using assms(1) unfolding monic-poly-def univ-poly-zero by auto

```

```

  have  $S \neq \{\}$ 

```

```

  proof (rule ccontr)

```

```

    assume  $S$ -empty:  $\neg(S \neq \{\})$ 

```

```

    have  $f = (\bigotimes_{\text{poly-ring } R} d \in S. d [\wedge]_{\text{poly-ring } R} \text{pmult } d f)$ 

```

```

      unfolding  $S$ -def by (intro factor-monic-poly assms(1))

```

```

    also have  $\dots = \mathbf{1}_{\text{poly-ring } R}$  using  $S$ -empty by simp

```

```

    finally have  $f = \mathbf{1}_{\text{poly-ring } R}$  by simp

```

```

    hence degree  $f = 0$  using degree-one by simp
  qed

```

**thus** *False* **using** *assms(2)* **by** *simp*  
**qed**  
**then obtain**  $g$  **where**  $g\text{-irred}$ : *monic-irreducible-poly*  $R$   $g$  **and**  $0 <$   
 $\text{pmult } g \ f$   
**unfolding**  $S\text{-def}$  **by** *auto*  
  
**hence**  $1 \leq \text{pmult } g \ f$  **by** *simp*  
  
**hence**  $g\text{-div}$ :  $g$  *pdivides*  $f$  **using** *multiplicity-ge-1-iff-pdivides*  $f\text{-carr}$   
 $g\text{-irred}$  **by** *blast*  
  
**then obtain**  $h$  **where**  $f\text{-def}$ :  $f = g \otimes_{\text{poly-ring } R} h$  **and**  $h\text{-carr}$ :  $h \in$   
*carrier* (*poly-ring*  $R$ )  
**unfolding** *pdivides-def* **by** *auto*  
  
**have**  $g\text{-nz}$ :  $g \neq \mathbf{0}_{\text{poly-ring } R}$  **and**  $h\text{-nz}$ :  $h \neq \mathbf{0}_{\text{poly-ring } R}$   
**and**  $g\text{-carr}$ :  $g \in \text{carrier}$  (*poly-ring*  $R$ )  
**using**  $f\text{-carr}(2)$   $h\text{-carr}$   $g\text{-irred}$  **unfolding**  $f\text{-def}$  *monic-irreducible-poly-def*  
*monic-poly-def*  
**by** *auto*  
  
**have**  $\text{degree } f = \text{degree } g + \text{degree } h$   
**using**  $g\text{-nz}$   $h\text{-nz}$   $g\text{-carr}$   $h\text{-carr}$  **unfolding**  $f\text{-def}$  **by** (*intro de-*  
*gree-mult[OF r.K-subring]*) *auto*  
**moreover have**  $\text{degree } h > 0$   
**proof** (*rule ccontr*)  
**assume**  $\neg(\text{degree } h > 0)$   
**hence**  $\text{degree } h = 0$  **by** *simp*  
**hence**  $h \in \text{Units}$  (*poly-ring*  $R$ )  
**using**  $h\text{-carr}$   $h\text{-nz}$  **by** (*simp add: carrier-is-subfield univ-poly-units'*  
*univ-poly-zero*)  
**hence**  $f \sim_{\text{poly-ring } R} g$   
**unfolding**  $f\text{-def}$  **using**  $g\text{-carr}$  *r.p.associatedI2'* **by** *force*  
**hence**  $f \sim_{\text{mult-of } (\text{poly-ring } R)} g$   
**using**  $f\text{-carr}$   $g\text{-nz}$   $g\text{-carr}$  **by** (*simp add: r.p.assoc-iff-assoc-mult*)  
**hence**  $f = g$   
**using** *monic-poly-not-assoc*  $\text{assms}(1)$   $g\text{-irred}$  **unfolding** *monic-irreducible-poly-def*  
**by** *simp*  
**hence** *monic-irreducible-poly*  $R$   $f$   
**using**  $g\text{-irred}$  **by** *simp*  
**thus** *False*  
**using**  $\text{assms}(3)$  **by** *auto*  
**qed**  
**ultimately have**  $\text{degree } g < \text{degree } f$  **by** *simp*  
**thus** *?thesis* **using**  $g\text{-irred}$   $g\text{-div}$  **by** *auto*  
**qed**  
  
**theorem** *rabin-irreducibility-condition*:  
**assumes** *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** *monic-irreducible-poly*  $R f \longleftrightarrow$   
 $(f \text{ pdivides } \text{gauss-poly } R (\text{order } R \hat{\text{degree}} f) \wedge (\forall n \in N. \text{pcoprime } (\text{gauss-poly } R (\text{order } R \hat{n}) f))$   
 $(\text{is } ?L \longleftrightarrow ?R1 \wedge ?R2)$   
**proof** –  
**have**  $f\text{-carr}: f \in \text{carrier } (\text{poly-ring } R)$   
**using** *assms(1)* **unfolding** *monic-poly-def* **by** *blast*  
  
**have**  $?R1$  **if**  $?L$   
**using** *div-gauss-poly-iff* [**where**  $n = \text{degree } f$ ] **that** *assms(2)* **by** *simp*  
**moreover** **have** *False* **if**  $\text{cthat}: \neg \text{pcoprime } (\text{gauss-poly } R (\text{order } R \hat{n}))$   
 $f ?L n \in N$  **for**  $n$   
**proof** –  
**obtain**  $d$  **where**  $d\text{-def}$ :  
 $d \text{ pdivides } f$   
 $d \text{ pdivides } (\text{gauss-poly } R (\text{order } R \hat{n}))$   $\text{degree } d > 0$   $d \in \text{carrier}$   
 $(\text{poly-ring } R)$   
**using** *cthat(1)* **unfolding** *pcoprime-def* **by** *auto*  
  
**obtain**  $p$  **where**  $p\text{-def}$ :  
 $n = \text{degree } f \text{ div } p$  *Factorial-Ring.prime*  $p$   $p \text{ dvd } \text{degree } f$   
**using** *cthat(3)* **unfolding**  $N\text{-def}$  **by** *auto*  
  
**have**  $n\text{-gt-0}: n > 0$   
**using**  $p\text{-def}$  *assms(2)* **by** (*metis dvd-div-eq-0-iff gr0I*)  
  
**have**  $d \notin \text{Units } (\text{poly-ring } R)$   
**using**  $d\text{-def}(3,4)$  *univ-poly-units'* [*OF carrier-is-subfield*] **by** *simp*  
**hence**  $f \text{ pdivides } d$   
**using** *cthat(2)*  $d\text{-def}(1,4)$  **unfolding** *monic-irreducible-poly-def*  
*ring-irreducible-def*  
 $\text{Divisibility.irreducible-def}$   $\text{properfactor-def}$   $\text{pdivides-def}$   $f\text{-carr}$  **by**  
*auto*  
**hence**  $f \text{ pdivides } (\text{gauss-poly } R (\text{order } R \hat{n}))$   
**using**  $d\text{-def}(2,4)$   $f\text{-carr}$  *r.p.divides-trans* **unfolding**  $\text{pdivides-def}$   
**by** *metis*  
**hence**  $\text{degree } f \text{ dvd } n$   
**using**  $n\text{-gt-0}$  *div-gauss-poly-iff* [*OF - cthat(2)*] **by** *auto*  
**thus** *False*  
**using**  $p\text{-def}$  **by** (*metis assms(2) div-less-dividend n-gt-0 nat-dvd-not-less*  
*prime-gt-1-nat*)  
**qed**  
**moreover** **have** *False* **if**  $\text{not-}l: \neg ?L$  **and**  $r1: ?R1$  **and**  $r2: ?R2$   
**proof** –  
**obtain**  $g$  **where**  $g\text{-def}$ :  $g \text{ pdivides } f$   $\text{degree } g < \text{degree } f$  *monic-irreducible-poly*  
 $R g$   
**using**  $r1$   $\text{not-}l$  *exists-irreducible-proper-factor* *assms(1,2)* **by** *auto*

```

have g-carr:  $g \in \text{carrier } (\text{poly-ring } R)$  and g-nz:  $g \neq \mathbf{0}_{\text{poly-ring } R}$ 
using g-def(3) unfolding monic-irreducible-poly-def monic-poly-def
by (auto simp:univ-poly-zero)

have g pdivides gauss-poly R (order R^n)
using g-carr r1 g-def(1) unfolding pdivides-def using r.p.divides-trans
by blast

hence degree g dvd degree f
using div-gauss-poly-iff[OF assms(2) g-def(3)] by auto

then obtain t where deg-f-def: degree f = t * degree g
by fastforce
hence  $t > 1$  using g-def(2) by simp
then obtain p where p-prime: Factorial-Ring.prime p p dvd t
by (metis order-less-irrefl prime-factor-nat)
hence p-div-deg-f: p dvd degree f
unfolding deg-f-def by simp
define n where  $n = \text{degree } f \text{ div } p$ 
have n-in-N: n ∈ N
unfolding N-def n-def using p-prime(1) p-div-deg-f by auto

have deg-g-dvd-n: degree g dvd n
using p-prime(2) unfolding n-def deg-f-def by auto

have n-gt-0: n > 0
using p-div-deg-f assms(2) p-prime(1) unfolding n-def
by (metis dvd-div-eq-0-iff gr0I)

have deg-g-gt-0: degree g > 0
using monic-poly-min-degree[OF g-def(3)] by simp

have 0:g pdivides gauss-poly R (order R^n)
using deg-g-dvd-n div-gauss-poly-iff[OF n-gt-0 g-def(3)] by simp

have pcoprime (gauss-poly R (order R^n)) f
using n-in-N r2 by simp
thus False
using 0 g-def(1) g-carr deg-g-gt-0 unfolding pcoprime-def by
simp
qed
ultimately show ?thesis
by auto
qed

```

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**  $N \equiv \{\text{degree } f \text{ div } p \mid p . \text{Factorial-Ring.prime } p \wedge p \text{ dvd } \text{degree } f\}$   
**shows**  $\text{pirreducible } (\text{carrier } R) f \longleftrightarrow$   
 $(f \text{ pdivides } \text{gauss-poly } R (\text{order } R \widehat{\text{degree } f}) \wedge (\forall n \in N. \text{pcoprime } (\text{gauss-poly } R (\text{order } R \widehat{n}) f))$   
 $(\text{is } ?L \longleftrightarrow ?R1 \wedge ?R2)$   
**proof** –  
**define**  $\alpha$  **where**  $\alpha = [\text{inv } (\text{hd } f)]$   
**let**  $?g = (\lambda x. \text{gauss-poly } R (\text{order } R \widehat{x}))$   
**let**  $?h = \alpha \otimes_{\text{poly-ring } R} f$   
  
**have**  $f\text{-nz}: f \neq \mathbf{0}_{\text{poly-ring } R}$  **unfolding**  $\text{univ-poly-zero}$  **using**  $\text{assms}(2)$   
**by**  $\text{auto}$   
  
**hence**  $\text{hd } f \in \text{carrier } R - \{\mathbf{0}\}$  **using**  $\text{assms}(1)$   $\text{lead-coeff-carr}$  **by**  
 $\text{simp}$   
**hence**  $\text{inv } (\text{hd } f) \in \text{carrier } R - \{\mathbf{0}\}$  **using**  $\text{field-Units}$  **by**  $\text{auto}$   
**hence**  $\alpha\text{-unit}: \alpha \in \text{Units } (\text{poly-ring } R)$   
**unfolding**  $\alpha\text{-def}$  **using**  $\text{univ-poly-carrier-units}$  **by**  $\text{simp}$   
  
**have**  $\alpha\text{-nz}: \alpha \neq \mathbf{0}_{\text{poly-ring } R}$  **unfolding**  $\text{univ-poly-zero}$   $\alpha\text{-def}$  **by**  
 $\text{simp}$   
**have**  $\text{hd } ?h = \text{hd } \alpha \otimes \text{hd } f$   
**using**  $\alpha\text{-nz } f\text{-nz } \text{assms}(1)$   $\alpha\text{-unit}$  **by**  $(\text{intro } \text{lead-coeff-mult}) \text{ auto}$   
**also have**  $\dots = \text{inv } (\text{hd } f) \otimes \text{hd } f$  **unfolding**  $\alpha\text{-def}$  **by**  $\text{simp}$   
**also have**  $\dots = \mathbf{1}$  **using**  $\text{lead-coeff-carr } f\text{-nz } \text{assms}(1)$  **by**  $(\text{simp add:}$   
 $\text{field-Units})$   
**finally have**  $\text{hd } ?h = \mathbf{1}$  **by**  $\text{simp}$   
**moreover have**  $?h \neq []$   
**using**  $\alpha\text{-nz } f\text{-nz } \text{univ-poly-zero}$  **by**  $(\text{metis } \alpha\text{-unit } \text{assms}(1) \text{ r.p.Units-closed}$   
 $\text{r.p.integral})$   
**ultimately have**  $h\text{-monic}: \text{monic-poly } R ?h$   
**using**  $\text{r.p.Units-closed}[OF \alpha\text{-unit}] \text{ assms}(1)$  **unfolding**  $\text{monic-poly-def}$   
**by**  $\text{auto}$   
  
**have**  $\text{degree } ?h = \text{degree } \alpha + \text{degree } f$   
**using**  $\text{assms}(1)$   $f\text{-nz } \alpha\text{-unit } \alpha\text{-nz}$  **by**  $(\text{intro } \text{degree-mult}[OF \text{carrier-is-subring}]) \text{ auto}$   
**also have**  $\dots = \text{degree } f$  **unfolding**  $\alpha\text{-def}$  **by**  $\text{simp}$   
**finally have**  $\text{deg-f}: \text{degree } f = \text{degree } ?h$  **by**  $\text{simp}$   
  
**have**  $hf\text{-cong}: ?h \sim_{r.P} f$   
**using**  $\text{assms}(1)$   $\alpha\text{-unit}$  **by**  $(\text{simp add: } \text{r.p.Units-closed } \text{r.p.associatedI2}$   
 $\text{r.p.m-comm})$   
**hence**  $0: f \text{ pdivides } ?g (\text{degree } f) \longleftrightarrow ?h \text{ pdivides } ?g (\text{degree } f)$   
**unfolding**  $\text{pdivides-def}$  **using**  $\text{r.p.divides-cong-l } \text{r.p.associated-sym}$   
**using**  $\text{r.p.Units-closed}[OF \alpha\text{-unit}] \text{ assms}(1)$   $\text{gauss-poly-carr}$  **by**  
 $\text{blast}$

```

have 1: pcoprime (?g n) f  $\longleftrightarrow$  pcoprime (?g n) ?h for n
using hf-cong r.p.associated-sym r.p.Units-closed[OF  $\alpha$ -unit] assms(1)
by (intro p coprime-right-assoc-cong gauss-poly-carr) auto

have ?L  $\longleftrightarrow$  pirreducible (carrier R) ( $\alpha \otimes_{\text{poly-ring } R} f$ )
using  $\alpha$ -unit  $\alpha$ -nz assms(1) f-nz r.p.integral unfolding ring-irreducible-def
by (intro arg-cong2[where f=( $\wedge$ )] r.p.irreducible-prod-unit assms)
auto
also have ...  $\longleftrightarrow$  monic-irreducible-poly R ( $\alpha \otimes_{\text{poly-ring } R} f$ )
using h-monic unfolding monic-irreducible-poly-def by auto
also have ...  $\longleftrightarrow$  ?h pdivides ?g (degree f)  $\wedge$  ( $\forall n \in N.$  pcoprime
(?g n) ?h)
using assms(2) unfolding N-def deg-f by (intro rabin-irreducibility-condition
h-monic) auto
also have ...  $\longleftrightarrow$  f pdivides ?g (degree f)  $\wedge$  ( $\forall n \in N.$  pcoprime (?g
n) f)
using 0 1 by simp
finally show ?thesis by simp
qed

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  $\Rightarrow$  'a  $\Rightarrow$  'a
  idx-udivide :: 'a  $\Rightarrow$  'a
  idx-mult :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a
  idx-zero :: 'a
  idx-one :: 'a

record 'a idx-ring-enum = 'a idx-ring +
  idx-size :: nat
  idx-enum :: nat  $\Rightarrow$  'a
  idx-enum-inv :: 'a  $\Rightarrow$  nat

fun idx-pow :: ('a,'b) idx-ring-scheme  $\Rightarrow$  'a  $\Rightarrow$  nat  $\Rightarrow$  '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 ( $\langle 0_{C1} \rangle$ )
notation idx-one ( $\langle 1_{C1} \rangle$ )
notation idx-plus (infixl  $\langle +_{C1} \rangle$  65)
notation idx-mult (infixl  $\langle *_{C1} \rangle$  70)
notation idx-uminus ( $\langle -_{C1} \rightarrow [81] 80 \rangle$ )
notation idx-udivide ( $\langle \cdot^{-1}_{C1} [81] 80 \rangle$ )
notation idx-pow (infixr  $\langle \wedge_{C1} \rangle$  75)
end

definition ring-of :: ('a,'b) idx-ring-scheme  $\Rightarrow$  'a ring
where ring-of A =  $\langle$ 
  carrier = {x. idx-pred A x},
  mult =  $(\lambda$  x y. x  $*_{CA}$  y),
  one =  $1_{CA}$ ,
  zero =  $0_{CA}$ ,
  add =  $(\lambda$  x y. x  $+_{CA}$  y)  $\rangle$ 

definition ringC where
  ringC A = (ring (ring-of A)  $\wedge$  ( $\forall$ x. idx-pred A x  $\longrightarrow$   $-_{CA}$  x =
 $\ominus_{ring-of A}$  x)  $\wedge$ 
  ( $\forall$ x. x  $\in$  Units (ring-of A)  $\longrightarrow$   $x^{-1}_{CA}$  = invring-of A x))

```

**lemma ring-cD-ax:**  
 $x \widehat{ }_C A n = x [\widehat{ }]_{ring-of A} n$   
**by** (induction n) (auto simp:ring-of-def)

**lemma ring-cD:**  
**assumes** ring<sub>C</sub> A  
**shows**  
 $0_{CA} = \mathbf{0}_{ring-of A}$   
 $1_{CA} = \mathbf{1}_{ring-of A}$   
 $\bigwedge x y. x *_CA y = x \otimes_{ring-of A} y$   
 $\bigwedge x y. x +_CA y = x \oplus_{ring-of A} y$   
 $\bigwedge x. x \in carrier (ring-of A) \implies -_CA x = \ominus_{ring-of A} x$   
 $\bigwedge x. x \in Units (ring-of A) \implies x^{-1}_{CA} = inv_{ring-of A} x$   
 $\bigwedge x. x \widehat{ }_C A n = x [\widehat{ }]_{ring-of A} n$   
**using** assms ring-cD-ax **unfolding** ring<sub>C</sub>-def ring-of-def **by** auto

**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<sub>C</sub> A  
**proof** –  
**have**  $x \in carrier (ring-of A) \iff idx\_pred A x$  **for** x **unfolding**  
ring-of-def **by** auto  
**thus** ?thesis **using** assms **unfolding** ring<sub>C</sub>-def **by** auto  
**qed**

**definition cring<sub>C</sub> where** cring<sub>C</sub> A = (ring<sub>C</sub> 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<sub>C</sub> A  
**unfolding** cring<sub>C</sub>-def **by** (intro ring-cI conjI assms cring.axioms(1))

**lemma cring-c-imp-ring: cring<sub>C</sub> A  $\implies$  ring<sub>C</sub> A**  
**unfolding** cring<sub>C</sub>-def **by** simp

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

**definition domain<sub>C</sub> where** domain<sub>C</sub> A = (cring<sub>C</sub> 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**  $\text{domain}_C A$   
**unfolding**  $\text{domain}_C\text{-def}$  **by** (*intro conjI cring-cI assms domain.axioms(1)*)

**lemma**  $\text{domain-c-imp-ring}$ :  $\text{domain}_C A \implies \text{ring}_C A$   
**unfolding**  $\text{cring}_C\text{-def}$   $\text{domain}_C\text{-def}$  **by** *simp*

**lemmas**  $\text{domain-cD} = \text{ring-cD}[OF \text{domain-c-imp-ring}]$

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

**lemma**  $\text{field-cI}$ :  
**assumes**  $\text{field}(\text{ring-of } A)$   
**assumes**  $\bigwedge x. x \in \text{carrier}(\text{ring-of } A) \implies -_C A x = \ominus_{\text{ring-of } A} x$   
**assumes**  $\bigwedge x. x \in \text{Units}(\text{ring-of } A) \implies x^{-1}_C A = \text{inv}_{\text{ring-of } A} x$   
**shows**  $\text{field}_C A$   
**unfolding**  $\text{field}_C\text{-def}$  **by** (*intro conjI domain-cI assms field.axioms(1)*)

**lemma**  $\text{field-c-imp-ring}$ :  $\text{field}_C A \implies \text{ring}_C A$   
**unfolding**  $\text{field}_C\text{-def}$   $\text{cring}_C\text{-def}$   $\text{domain}_C\text{-def}$  **by** *simp*

**lemmas**  $\text{field-cD} = \text{ring-cD}[OF \text{field-c-imp-ring}]$

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

**lemma**  $\text{enum-cI}$ :  
**assumes**  $\text{finite}(\text{carrier}(\text{ring-of } A))$   
**assumes**  $\text{idx-size } A = \text{order}(\text{ring-of } A)$   
**assumes**  $\text{bij-betw}(\text{idx-enum } A) \{..<\text{order}(\text{ring-of } A)\}(\text{carrier}(\text{ring-of } A))$   
**assumes**  $\bigwedge x. x < \text{order}(\text{ring-of } A) \implies \text{idx-enum-inv } A (\text{idx-enum } A x) = x$   
**shows**  $\text{enum}_C A$   
**using** *assms* **unfolding**  $\text{enum}_C\text{-def}$  **by** *auto*

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

```

x) = x
  using assms
proof -
  let ?n = order (ring-of R)
  have a:idx-enum-inv R x = the-inv-into {..?n} (idx-enum R) x
    if x-carr: x ∈ carrier (ring-of R) for x
  proof -
    have idx-enum R ‘ {..order (ring-of R)} = carrier (ring-of R)
      using assms unfolding bij-betw-def enumC-def by simp
    then obtain y where y-carr: y ∈ {..order (ring-of R)} and
x-def: x = idx-enum R y
      using x-carr by auto
    have idx-enum-inv R x = y using assms y-carr unfolding x-def
enumC-def by simp
    also have ... = the-inv-into {..?n} (idx-enum R) x
      using assms unfolding bij-betw-def enumC-def unfolding x-def
      by (intro the-inv-into-f-f[symmetric] y-carr) auto
    finally show ?thesis by simp
qed

  have bij-betw (the-inv-into {..?n} (idx-enum R)) (carrier (ring-of
R)) {..?n}
    using assms unfolding enumC-def by (intro bij-betw-the-inv-into)
auto
  thus bij-betw (idx-enum-inv R) (carrier (ring-of R)) {..order (ring-of
R)}
    by (subst bij-betw-cong[OF a]) auto
  show idx-enum R (idx-enum-inv R x) = x if x ∈ carrier (ring-of R)
for x
    using that assms unfolding a[OF that] enumC-def bij-betw-def by
(intro f-the-inv-into-f) auto
qed (use assms enumC-def in auto)

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 ≠ 0CE 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) 0CE) @ 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) 0CE)) (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 ≠ 0CE) ∧ 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 (⟨XC1⟩)
where poly-var E = [idx-one E, idx-zero E]

```

```

lemma poly-var: poly-var R = Xring-of R
unfolding var-def poly-var-def by (simp add:ring-of-def)

```

```

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

```

```

lemma ring-of-poly:
  assumes ringC A
  shows ring-of (poly A) = poly-ring (ring-of A)
proof (intro ring.equality)
  interpret ring ring-of A using assms unfolding ringC-def by auto

```

```

have b: 0ring-of A = 0CA unfolding ring-of-def by simp
have c: (⊗ring-of A) = (*CA) unfolding ring-of-def by simp
have d: (⊕ring-of A) = (+CA) unfolding ring-of-def by simp

```

```

have o-normalize A x = normalize x for x
  using b by (induction x) simp-all

```

**hence**  $o\text{-poly-add } A \ x \ y = \text{poly-add } x \ y$  **if**  $\text{length } y \leq \text{length } x$  **for**  $x \ y$   
**using** that **by** (*subst o-poly-add.simps, subst poly-add.simps*) (*simp add: b d*)

**hence**  $a:o\text{-poly-add } A \ x \ y = \text{poly-add } x \ y$  **for**  $x \ y$   
**by** (*subst o-poly-add.simps, subst poly-add.simps*) *simp*

**hence**  $x \oplus_{\text{ring-of } (poly \ A)} \ y = x \oplus_{\text{poly-ring } (ring-of \ A)} \ y$  **for**  $x \ y$   
**by** (*simp add: univ-poly-def poly-def ring-of-def*)

**thus**  $(\oplus_{\text{ring-of } (poly \ A)}) = (\oplus_{\text{poly-ring } (ring-of \ A)})$  **by** (*intro ext*)

**show**  $\text{carrier } (ring-of \ (poly \ A)) = \text{carrier } (poly-ring \ (ring-of \ A))$   
**by** (*auto simp add: ring-of-def poly-def univ-poly-def polynomial-def list-all-iff*)

**have**  $o\text{-poly-mult } A \ x \ y = \text{poly-mult } x \ y$  **for**  $x \ y$

**proof** (*induction x*)

**case** *Nil* **then show** *?case* **by** *simp*

**next**

**case** (*Cons a x*) **then show** *?case*

**by** (*subst o-poly-mult.simps, subst poly-mult.simps*)

(*simp add: a b c del: poly-add.simps o-poly-add.simps*)

**qed**

**hence**  $x \otimes_{\text{ring-of } (poly \ A)} \ y = x \otimes_{\text{poly-ring } (ring-of \ A)} \ y$  **for**  $x \ y$

**by** (*simp add: univ-poly-def poly-def ring-of-def*)

**thus**  $(\otimes_{\text{ring-of } (poly \ A)}) = (\otimes_{\text{poly-ring } (ring-of \ A)})$  **by** (*intro ext*)

**qed** (*simp-all add: ring-of-def poly-def univ-poly-def*)

**lemma** *poly-eval*:

**assumes**  $\text{ring}_C \ R$

**assumes**  $\text{fsc:fs} \in \text{carrier } (ring-of \ (poly \ R))$  **and**  $\text{xc:x} \in \text{carrier } (ring-of \ R)$

**shows**  $\text{poly-eval } R \ \text{fs} \ x = \text{ring.eval } (ring-of \ R) \ \text{fs} \ x$

**proof** –

**interpret**  $\text{ring } ring-of \ R$  **using** *assms* **unfolding**  $\text{ring}_C\text{-def}$  **by** *auto*

**have**  $\text{fs-carr:fs} \in \text{carrier } (poly-ring \ (ring-of \ R))$  **using**  $\text{ring-of-poly}[OF \ \text{assms}(1)] \ \text{fsc}$  **by** *auto*

**hence**  $\text{set } \text{fs} \subseteq \text{carrier } (ring-of \ R)$  **by** (*simp add: polynomial-incl univ-poly-carrier*)

**thus** *?thesis*

**proof** (*induction rule: rev-induct*)

**case** *Nil* **thus** *?case* **by** *simp* (*simp add: ring-of-def*)

**next**

**case** (*snoc ft fh*)

**have**  $\text{poly-eval } R \ (\text{fh} \ @ \ [\text{ft}]) \ x = \text{poly-eval } R \ \text{fh} \ x \ *_{C \ R} \ x \ +_{C \ R} \ \text{ft}$

**by simp**  
**also have** ... = *eval fh x \*<sub>C R</sub> x +<sub>C R</sub> ft* **using snoc by** (*subst snoc*)  
*auto*  
**also have** ... = *eval fh x ⊗<sub>ring-of R</sub> x ⊕<sub>ring-of R</sub> ft* **by** (*simp*  
*add:ring-of-def*)  
**also have** ... = *eval (fh@[ft]) x* **using snoc by** (*intro eval-append-aux[symmetric]*  
*xc*) *auto*  
**finally show ?case by auto**  
**qed**  
**qed**

**lemma poly-domain:**  
**assumes** *domain<sub>C</sub> A*  
**shows** *domain<sub>C</sub> (poly A)*  
**proof** –  
**interpret** *domain ring-of A* **using** *assms* **unfolding** *domain<sub>C</sub>-def*  
**by auto**

**have** *a: ⊖<sub>ring-of A</sub> x = -<sub>C A</sub> x* **if** *x ∈ carrier (ring-of A)* **for** *x*  
**using that by** (*intro domain-cD[symmetric] assms*)  
**have** *ring<sub>C</sub> A*  
**using** *assms* **unfolding** *domain<sub>C</sub>-def cring<sub>C</sub>-def* **by auto**  
**hence** *b: ring-of (poly A) = poly-ring (ring-of A)*  
**by** (*subst ring-of-poly*) *auto*

**have** *c: domain (ring-of (poly A))*  
**unfolding b by** (*rule univ-poly-is-domain[OF carrier-is-subring]*)

**interpret** *d: domain poly-ring (ring-of A)*  
**using c** **unfolding b by simp**

**have** *-<sub>C poly A</sub> x = ⊖<sub>ring-of (poly A)</sub> x* **if** *x ∈ carrier (ring-of (poly A))* **for** *x*

**proof** –  
**have** *⊖<sub>ring-of (poly A)</sub> x = map (a-inv (ring-of A)) x*  
**using that** **unfolding b by** (*subst univ-poly-a-inv-def'[OF carrier-is-subring]*) *auto*  
**also have** ... = *map (λr. -<sub>C A</sub> r) x*  
**using that** **unfolding b** *univ-poly-carrier[symmetric] polynomial-def*  
**by** (*intro map-cong refl a*) *auto*  
**also have** ... = *-<sub>C poly A</sub> x*  
**unfolding poly-def by simp**  
**finally show ?thesis by simp**

**qed**  
**moreover have** *x<sup>-1</sup><sub>C poly A</sub> = inv<sub>ring-of (poly A)</sub> x* **if** *x ∈ Units (ring-of (poly A))* **for** *x*

**proof** –  
**have** *x ∈ {[k] | k. k ∈ carrier (ring-of A) - {0<sub>ring-of A</sub>}}*

**using** that *univ-poly-carrier-units-incl* **unfolding** *b* **by** *auto*  
**then obtain** *k* **where** *x-eq*:  $k \in \text{carrier } (\text{ring-of } A) - \{\mathbf{0}_{\text{ring-of } A}\}$   
 $x = [k]$  **by** *auto*  
**have**  $\text{inv}_{\text{ring-of } (\text{poly } A)} x \in \text{Units } (\text{poly-ring } (\text{ring-of } A))$   
**using** that **unfolding** *b* **by** *simp*  
**hence**  $\text{inv}_{\text{ring-of } (\text{poly } A)} x \in \{[k] \mid k. k \in \text{carrier } (\text{ring-of } A) - \{\mathbf{0}_{\text{ring-of } A}\}\}$   
**using** that *univ-poly-carrier-units-incl* **unfolding** *b* **by** *auto*  
**then obtain** *v* **where** *x-inv-eq*:  $v \in \text{carrier } (\text{ring-of } A) - \{\mathbf{0}_{\text{ring-of } A}\}$   
 $\text{inv}_{\text{ring-of } (\text{poly } A)} x = [v]$  **by** *auto*  
  
**have**  $\text{poly-mult } [k] [v] = [k] \otimes_{\text{ring-of } (\text{poly } A)} [v]$  **unfolding** *b*  
*univ-poly-mult* **by** *simp*  
**also have**  $\dots = x \otimes_{\text{ring-of } (\text{poly } A)} \text{inv}_{\text{ring-of } (\text{poly } A)} x$  **using**  
*x-inv-eq* *x-eq* **by** *auto*  
**also have**  $\dots = \mathbf{1}_{\text{ring-of } (\text{poly } A)}$  **using** that **unfolding** *b* **by** *simp*  
**also have**  $\dots = [\mathbf{1}_{\text{ring-of } A}]$  **unfolding** *b* *univ-poly-one* **by** (*simp*  
*add:ring-of-def*)  
**finally have**  $\text{poly-mult } [k] [v] = [\mathbf{1}_{\text{ring-of } A}]$  **by** *simp*  
**hence**  $k \otimes_{\text{ring-of } A} v \oplus_{\text{ring-of } A} \mathbf{0}_{\text{ring-of } A} = \mathbf{1}_{\text{ring-of } A}$   
**by** (*simp* *add:if-distribR* *if-distrib*) (*simp* *cong:if-cong*, *metis*)  
**hence**  $e: k \otimes_{\text{ring-of } A} v = \mathbf{1}_{\text{ring-of } A}$  **using** *x-eq(1)* *x-inv-eq(1)*  
**by** *simp*  
**hence**  $f: v \otimes_{\text{ring-of } A} k = \mathbf{1}_{\text{ring-of } A}$  **using** *x-eq(1)* *x-inv-eq(1)*  
*m-comm* **by** *simp*  
**have**  $g: v = \text{inv}_{\text{ring-of } A} k$   
**using** *e* *x-eq(1)* *x-inv-eq(1)* **by** (*intro* *comm-inv-char[symmetric]*)  
*auto*  
**hence**  $h: k \in \text{Units } (\text{ring-of } A)$  **unfolding** *Units-def* **using** *e* *f*  
*x-eq(1)* *x-inv-eq(1)* **by** *blast*  
  
**have**  $x^{-1} {}_C \text{poly } A = [k]^{-1} {}_C \text{poly } A$  **unfolding** *x-eq* **by** *simp*  
**also have**  $\dots = [k^{-1} {}_C A]$  **unfolding** *poly-def* **by** *simp*  
**also have**  $\dots = [v]$   
**unfolding** *g* **by** (*intro* *domain-cD[OF assms(1)]* *arg-cong2* [*where*  
*f=(#)]* *h refl*)  
**also have**  $\dots = \text{inv}_{\text{ring-of } (\text{poly } A)} x$  **unfolding** *x-inv-eq* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**  
**ultimately show** *?thesis* **using** *c* **by** (*intro* *domain-cI*)  
**qed**

**function** *long-division<sub>C</sub>* :: (*'a*,*'b*) *idx-ring-scheme*  $\Rightarrow$  *'a list*  $\Rightarrow$  *'a list*  
 $\Rightarrow$  *'a list*  $\times$  *'a list*  
**where** *long-division<sub>C</sub>*  $F f g =$  (  
*if* (*length* *g* = 0  $\vee$  *length* *f* < *length* *g*)  
*then* ([], *f*)  
*else* (

```

    let k = length f - length g;
        α = -C F (hd f *C F (hd g) -1C F);
        h = [α] *C poly F XC F ^C poly F k;
        f' = f +C poly F (h *C poly F g);
        f'' = take (length f - 1) f'
    in apfst (λx. x +C poly F -C poly F h) (long-divisionC F f'' g))
  by pat-completeness auto

```

**lemma** *pmod-termination-helper*:

```

  g ≠ [] ⇒ ¬length f < length g ⇒ min x (length f - 1) < length f
  by (metis diff-less length-greater-0-conv list.size(3) min.strict-coboundedI2
    zero-less-one)

```

**termination by** (*relation measure* (λ(-, f, -). length f)) (*use pmod-termination-helper in auto*)

**declare** *long-division<sub>C</sub>.simps*[*simp del*]

**lemma** *long-division-c-length*:

```

  assumes length g > 0
  shows length (snd (long-divisionC R f g)) < length g
  proof (induction length f arbitrary:f rule:nat-less-induct)
  case 1
  have 0: length (snd (long-divisionC R x g)) < length g
    if length x < length f for x using 1 that by blast
  show length (snd (long-divisionC R f g)) < length g
  proof (cases length f < length g)
  case True then show ?thesis by (subst long-divisionC.simps) simp
  next
  case False
  hence length f > 0 using assms by auto
  thus ?thesis using assms by (subst long-divisionC.simps)
    (auto intro!:0 simp: min.commute min.strict-coboundedII Let-def)
  qed
  qed

```

**context** *field*

**begin**

```

interpretation r:polynomial-ring R (carrier R)
  unfolding polynomial-ring-def polynomial-ring-axioms-def
  using carrier-is-subfield field-axioms by force

```

**lemma** *poly-length-from-coeff*:

```

  assumes p ∈ carrier (poly-ring R)
  assumes ∧i. i ≥ k ⇒ coeff p i = 0
  shows length p ≤ k

```

**proof** (*rule ccontr*)  
**assume**  $a: \neg \text{length } p \leq k$   
**hence**  $p\text{-nz}: p \neq []$  **by** *auto*  
**have**  $k < \text{length } p$  **using**  $a$  **by** *simp*  
**hence**  $k \leq \text{length } p - 1$  **by** *simp*  
**hence**  $0 = \text{coeff } p (\text{degree } p)$  **by** (*intro assms(2)[symmetric]*)  
**also have**  $\dots = \text{lead-coeff } p$  **by** (*intro lead-coeff-simp[OF p-nz]*)  
**finally have**  $0 = \text{lead-coeff } p$  **by** *simp*  
**thus** *False*  
**using**  $p\text{-nz}$  *assms(1)* **unfolding** *univ-poly-def polynomial-def* **by**  
*simp*  
**qed**

**lemma** *poly-add-cancel-len*:

**assumes**  $f \in \text{carrier } (\text{poly-ring } R) - \{0_{\text{poly-ring } R}\}$   
**assumes**  $g \in \text{carrier } (\text{poly-ring } R) - \{0_{\text{poly-ring } R}\}$   
**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** –  
**have**  $f\text{-ne}: f \neq []$  **using** *assms(1)* **unfolding** *univ-poly-zero* **by** *simp*  
**have**  $g\text{-ne}: g \neq []$  **using** *assms(2)* **unfolding** *univ-poly-zero* **by** *simp*  
  
**have**  $\text{coeff } f i = \ominus \text{coeff } g i$  **if**  $i \geq \text{degree } f$  **for**  $i$   
**proof** (*cases i = degree f*)  
**case** *True*  
**have**  $\text{coeff } f i = \text{hd } f$  **unfolding** *True* **by** (*subst lead-coeff-simp[OF f-ne]*) *simp*  
**also have**  $\dots = \ominus \text{hd } g$  **using** *assms(3)* **by** *simp*  
**also have**  $\dots = \ominus \text{coeff } g i$  **unfolding** *True* *assms(4)* **by** (*subst lead-coeff-simp[OF g-ne]*) *simp*  
**finally show** *?thesis* **by** *simp*  
**next**  
**case** *False*  
**hence**  $i > \text{degree } f$   $i > \text{degree } g$  **using** *assms(4)* **that** **by** *auto*  
**thus**  $\text{coeff } f i = \ominus \text{coeff } g i$  **using** *coeff-degree* **by** *simp*  
**qed**  
**hence**  $\text{coeff } (f \oplus_{\text{poly-ring } R} g) i = 0$  **if**  $i \geq \text{degree } f$  **for**  $i$   
**using** *assms(1,2)* **that** **by** (*subst r.coeff-add*) (*auto intro:l-neg simp:r.coeff-range*)  
  
**hence**  $\text{length } (f \oplus_{\text{poly-ring } R} g) \leq \text{length } f - 1$   
**using** *assms(1,2)* **by** (*intro poly-length-from-coeff*) *auto*  
**also have**  $\dots < \text{length } f$  **using**  $f\text{-ne}$  **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

**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** –  
**have**  $h \text{ pdivides } (h \otimes_{\text{poly-ring } R} (f \text{ pdiv } h)) \otimes_{\text{poly-ring } R} g$   
**using** *assms long-division-closed[OF carrier-is-subfield]*  
**by** (*simp add: dividesI' pdivides-def r.p.m-assoc*)  
**hence**  $0:(h \otimes_{\text{poly-ring } R} (f \text{ pdiv } h)) \otimes_{\text{poly-ring } R} g \text{ pmod } h = \mathbf{0}_{\text{poly-ring } R}$   
**using** *pmod-zero-iff-pdivides[OF carrier-is-subfield] assms long-division-closed[OF carrier-is-subfield] univ-poly-zero*  
**by** (*metis (no-types, opaque-lifting) r.p.m-closed*)  
  
**have**  $?L = (h \otimes_{\text{poly-ring } R} (f \text{ pdiv } h) \oplus_{\text{poly-ring } R} (f \text{ pmod } h)) \otimes_{\text{poly-ring } R} g \text{ pmod } h$   
**using** *assms by (intro arg-cong2[where f=( $\otimes_{\text{poly-ring } R}$ )] arg-cong2[where f=(pmod)] pdiv-pmod[OF carrier-is-subfield]) auto*  
**also have**  $\dots = ((h \otimes_{\text{poly-ring } R} (f \text{ pdiv } h)) \otimes_{\text{poly-ring } R} g \oplus_{\text{poly-ring } R} (f \text{ pmod } h) \otimes_{\text{poly-ring } R} g) \text{ pmod } h$   
**using** *assms long-division-closed[OF carrier-is-subfield]*  
**by** (*intro r.p.l-distr arg-cong2[where f=(pmod)] auto*)  
**also have**  $\dots = ((h \otimes_{\text{poly-ring } R} (f \text{ pdiv } h)) \otimes_{\text{poly-ring } R} g) \text{ pmod } h \oplus_{\text{poly-ring } R} ((f \text{ pmod } h) \otimes_{\text{poly-ring } R} g) \text{ pmod } h$   
**using** *assms long-division-closed[OF carrier-is-subfield]*  
**by** (*intro long-division-add[OF carrier-is-subfield] auto*)  
**also have**  $\dots = ?R$   
**using** *assms long-division-closed[OF carrier-is-subfield] unfolding*  
*0 by auto*  
**finally show** *?thesis*  
**by** *simp*  
**qed**

**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** –  
**have**  $?L = (g \otimes_{\text{poly-ring } R} f) \text{ pmod } h$  **using** *assms by algebra*  
**also have**  $\dots = ((g \text{ pmod } h) \otimes_{\text{poly-ring } R} f) \text{ pmod } h$  **by** (*intro pmod-mult-left assms*)  
**also have**  $\dots = ?R$  **using** *assms long-division-closed[OF carrier-is-subfield]*  
**by** *algebra*  
**finally show** *?thesis by simp*  
**qed**

**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** –  
**have**  $(f \otimes_{\text{poly-ring } R} g) \text{ pmod } h = ((f \text{ pmod } h) \otimes_{\text{poly-ring } R} g) \text{ pmod } h$   
**by** (*intro pmod-mult-left assms*)  
**also have**  $\dots = ?R$   
**using** *assms long-division-closed[OF carrier-is-subfield]* **by** (*intro pmod-mult-right*) *auto*  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *field-Unit-minus-closed*:  
**assumes**  $x \in \text{Units } R$   
**shows**  $\ominus x \in \text{Units } R$   
**using** *assms mult-of.Units-eq* **by** *auto*

**end**

**lemma** *long-division-c*:  
**assumes** *field<sub>C</sub> 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** –  
**let**  $?P = \text{poly-ring } (\text{ring-of } R)$   
**let**  $?result = (\lambda f r. f = \text{snd } r \oplus_{\text{poly-ring } (\text{ring-of } R)} (\text{fst } r \otimes_{\text{poly-ring } (\text{ring-of } R)} g))$   
**(g))**

**define**  $r$  **where**  $r = \text{long-division}_C R f g$

**interpret** *field ring-of R* **using** *assms(1)* **unfolding** *field<sub>C</sub>-def* **by** *auto*

**interpret** *d-poly-ring: domain poly-ring (ring-of R)*  
**by** (*rule univ-poly-is-domain[OF carrier-is-subring]*)

**have** *ring-c: ring<sub>C</sub> R* **using** *assms(1)* **unfolding** *field<sub>C</sub>-def domain<sub>C</sub>-def cring<sub>C</sub>-def* **by** *auto*

**have** *d-poly: domain<sub>C</sub> (poly R)* **using** *assms (1)* **unfolding** *field<sub>C</sub>-def* **by** (*intro poly-domain*) *auto*

**have**  $r = \text{long-division}_C R f g \implies ?result f r \wedge \{\text{fst } r, \text{snd } r\} \subseteq \text{carrier } (\text{poly-ring } (\text{ring-of } R))$

**using** *assms*(2)  
**proof** (*induction length f arbitrary: f r rule:nat-less-induct*)  
**case** 1

**have** *ind*:  $x = \text{snd } q \oplus_{?P} \text{fst } q \otimes_{?P} g \{ \text{fst } q, \text{snd } q \} \subseteq \text{carrier}$   
 (*poly-ring (ring-of R)*)  
**if**  $\text{length } x < \text{length } f \quad q = \text{long-division}_C R x g \quad x \in \text{carrier}$   
 (*poly-ring (ring-of R)*)  
**for**  $x \quad q$  **using** 1(1) **that by auto**

**show** *?case*  
**proof** (*cases length g = 0  $\vee$  length f < length g*)  
**case** *True*  
**hence**  $r = (\mathbf{0}_{\text{poly-ring (ring-of R)}}, f)$   
**unfolding** 1(2) *univ-poly-zero* **by** (*subst long-division\_C.simps*)  
*simp*  
**then show** *?thesis* **using** *assms*(3) 1(3) **by** *simp*  
**next**  
**case** *False*  
**hence**  $\text{length } g > 0 \quad \text{length } f \geq \text{length } g$  **by auto**  
**hence**  $f \neq [] \quad g \neq []$  **by auto**  
**hence**  $f\text{-carr}: f \in \text{carrier } ?P - \{\mathbf{0}_{?P}\}$  **and**  $g\text{-carr}: g \in \text{carrier}$   
 $?P - \{\mathbf{0}_{?P}\}$   
**using** 1(3) *assms*(3) *univ-poly-zero* **by auto**

**define**  $k$  **where**  $k = \text{length } f - \text{length } g$   
**define**  $\alpha$  **where**  $\alpha = -_C R (\text{hd } f *_C R (\text{hd } g)^{-1}_C R)$   
**define**  $h$  **where**  $h = [\alpha] *_C \text{poly } R \widehat{X}_C R \widehat{\ }_C \text{poly } R^k$   
**define**  $f'$  **where**  $f' = f +_C \text{poly } R (h *_C \text{poly } R g)$   
**define**  $f''$  **where**  $f'' = \text{take } (\text{length } f - 1) f'$   
**obtain**  $s \quad t$  **where**  $st\text{-def}: (s, t) = \text{long-division}_C R f'' g$  **by** (*metis surj-pair*)

**have**  $r = \text{apfst } (\lambda x. x +_C \text{poly } R -_C \text{poly } R h) (\text{long-division}_C R f'' g)$   
**using** *False* **unfolding** 1(2)  
**by** (*subst long-division\_C.simps*) (*simp add:Let-def f''-def f'-def h-def  $\alpha$ -def k-def*)

**hence**  $r\text{-def}: r = (s +_C \text{poly } R -_C \text{poly } R h, t)$   
**unfolding** *st-def[symmetric]* **by** *simp*

**have** *monic-poly (ring-of R) (X<sub>ring-of R</sub> [  $\widehat{\ }$  ]<sub>poly-ring (ring-of R)</sub> )<sup>k)</sup>*

**by** (*intro monic-poly-pow monic-poly-var*)  
**hence** [*simp*]:  $\text{lead-coeff } (X_{\text{ring-of } R} [\widehat{\ } ]_{\text{poly-ring (ring-of } R)}^k) =$   
 $\mathbf{1}_{\text{ring-of } R}$   
**unfolding** *monic-poly-def* **by** *simp*

**have**  $hd\text{-}f\text{-}unit$ :  $hd\ f \in Units\ (ring\text{-}of\ R)$  **and**  $hd\text{-}g\text{-}unit$ :  $hd\ g \in Units\ (ring\text{-}of\ R)$   
**using**  $f\text{-}carr\ g\text{-}carr\ lead\text{-}coeff\text{-}carr\ field\text{-}Units$  **by** *auto*  
**hence**  $hd\text{-}f\text{-}carr$ :  $hd\ f \in carrier\ (ring\text{-}of\ R)$  **and**  $hd\text{-}g\text{-}carr$ :  $hd\ g \in carrier\ (ring\text{-}of\ R)$   
**by** *auto*

**have**  $k\text{-}def'$ :  $k = degree\ f - degree\ g$  **using** *False* **unfolding**  $k\text{-}def$  **by** *auto*  
**have**  $\alpha\text{-}def'$ :  $\alpha = \ominus_{ring\text{-}of\ R}\ (hd\ f \otimes_{ring\text{-}of\ R}\ inv_{ring\text{-}of\ R}\ hd\ g)$   
**unfolding**  $\alpha\text{-}def$  **using**  $hd\text{-}g\text{-}unit\ hd\text{-}f\text{-}carr\ field\text{-}cD[OF\ assms(1)]$   
**by** *simp*

**have**  $\alpha\text{-}unit$ :  $\alpha \in Units\ (ring\text{-}of\ R)$  **unfolding**  $\alpha\text{-}def'$  **using**  $hd\text{-}f\text{-}unit\ hd\text{-}g\text{-}unit$   
**by** (*intro field-Unit-minus-closed*) *simp*  
**hence**  $\alpha\text{-}carr$ :  $\alpha \in carrier\ (ring\text{-}of\ R) - \{\mathbf{0}_{ring\text{-}of\ R}\}$  **unfolding**  $field\text{-}Units$  **by** *simp*  
**hence**  $\alpha\text{-}poly\text{-}carr$ :  $[\alpha] \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R)) - \{\mathbf{0}_{poly\text{-}ring\ (ring\text{-}of\ R)}\}$   
**by** (*simp add: univ-poly-carrier[symmetric] univ-poly-zero polynomial-def*)

**have**  $h\text{-}def'$ :  $h = [\alpha] \otimes_{?P}\ X_{ring\text{-}of\ R}\ [\wedge]_{?P}\ k$   
**unfolding**  $h\text{-}def\ poly\text{-}var\ domain\text{-}cD[OF\ d\text{-}poly]$  **by** (*simp add: ring-of-poly[OF ring-c]*)  
**have**  $f'\text{-}def'$ :  $f' = f \oplus_{?P}\ (h \otimes_{?P}\ g)$   
**unfolding**  $f'\text{-}def\ domain\text{-}cD[OF\ d\text{-}poly]$  **by** (*simp add: ring-of-poly[OF ring-c]*)

**have**  $h\text{-}carr$ :  $h \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R)) - \{\mathbf{0}_{poly\text{-}ring\ (ring\text{-}of\ R)}\}$   
**using**  $d\text{-}poly\text{-}ring.\text{mult-of.m-closed}\ \alpha\text{-}poly\text{-}carr\ var\text{-}pow\text{-}carr[OF\ carrier\text{-}is\text{-}subring]$   
**unfolding**  $h\text{-}def'$  **by** *auto*

**have**  $degree\ f = k + degree\ g$  **using** *False* **unfolding**  $k\text{-}def$  **by** *linarith*  
**also have**  $\dots = degree\ [\alpha] + degree\ (X_{ring\text{-}of\ R}\ [\wedge]_{?P}\ k) + degree\ g$   
**unfolding**  $var\text{-}pow\text{-}degree[OF\ carrier\text{-}is\text{-}subring]$  **by** *simp*  
**also have**  $\dots = degree\ h + degree\ g$  **unfolding**  $h\text{-}def'$   
**by** (*intro arg-cong2[where f=(+)] degree-mult[symmetric] carrier-is-subring  $\alpha\text{-}poly\text{-}carr\ var\text{-}pow\text{-}carr\ refl$* )  
**also have**  $\dots = degree\ (h \otimes_{poly\text{-}ring\ (ring\text{-}of\ R)}\ g)$   
**by** (*intro degree-mult[symmetric] carrier-is-subring h-carr g-carr*)  
**finally have**  $deg\text{-}f$ :  $degree\ f = degree\ (h \otimes_{poly\text{-}ring\ (ring\text{-}of\ R)}\ g)$   
**by** *simp*

```

have f'-carr: f' ∈ carrier (poly-ring (ring-of R))
  using f-carr h-carr g-carr unfolding f'-def' by auto

have hd f = ⊖ring-of R (α ⊗ring-of R lead-coeff g)
  using hd-g-unit hd-f-carr hd-g-carr α-unit α-carr unfolding
α-def'
  by (simp add: m-assoc l-minus)
also have ... = ⊖ring-of R (hd h ⊗ring-of R hd g)
  using hd-f-carr α-carr α-poly-carr var-pow-carr[OF carrier-is-subring]
unfolding h-def'
  by (subst lead-coeff-mult) (simp-all add:algebra-simps)
also have ... = ⊖ring-of R hd (h ⊗poly-ring (ring-of R) g)
  using h-carr g-carr by (subst lead-coeff-mult) auto
finally have hd f = ⊖ring-of R hd (h ⊗poly-ring (ring-of R) g)
  by simp
  hence len-f': length f' < length f using deg-f h-carr g-carr
d-poly-ring.integral
  unfolding f'-def' by (intro poly-add-cancel-len f-carr) auto
  hence f''-def': f'' = f' unfolding f''-def by simp

have {fst (s,t),snd (s,t)} ⊆ carrier (poly-ring (ring-of R))
  using len-f' f''-def' f'-carr by (intro ind(2)[where x=f'])
st-def) auto
  hence s-carr: s ∈ carrier ?P and t-carr: t ∈ carrier ?P by auto

have r-def': r = (s ⊖poly-ring (ring-of R) h, t)
  using h-carr domain-cD[OF d-poly] unfolding r-def a-minus-def
  using ring-of-poly[OF ring-c,symmetric] by simp

have r-carr: {fst r, snd r} ⊆ carrier (poly-ring (ring-of R))
  using s-carr t-carr h-carr unfolding r-def' by auto
have f = f'' ⊖?P h ⊗?P g
  using h-carr g-carr f-carr unfolding f''-def' f'-def' by simp
algebra
  also have ... = (snd (s,t) ⊕?P fst (s,t) ⊗?P g) ⊖?P h ⊗?P g
  using f'-carr f''-def' len-f'
  by (intro arg-cong2[where f=λx y. x ⊖?P y] ind(1) st-def)
auto
  also have ... = t ⊕?P (s ⊖?P h) ⊗?P g
  using s-carr t-carr h-carr g-carr by simp algebra
  also have ... = snd r ⊕poly-ring (ring-of R) fst r ⊗poly-ring (ring-of R)
g
  unfolding r-def' by simp
  finally have f = snd r ⊕poly-ring (ring-of R) fst r ⊗poly-ring (ring-of R)
g by simp
  thus ?thesis using r-carr by auto
qed
qed

```

**hence** *result*:  $?result\ f\ r\ \{fst\ r,\ snd\ r\} \subseteq carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$   
**using** *r-def* **by** *auto*  
**show** *?thesis*  
**proof** (*cases*  $g = []$ )  
**case** *True* **then show** *?thesis* **by** (*simp add:long-division<sub>C</sub>.simps pmod-def pdiv-def*)  
**next**  
**case** *False*  
**hence**  $snd\ r = [] \vee degree\ (snd\ r) < degree\ g$   
**using** *long-division-c-length* **unfolding** *r-def*  
**by** (*metis One-nat-def Suc-pred length-greater-0-conv not-less-eq*)  
**moreover have**  $f = g \otimes_{?P} (fst\ r) \oplus_{poly\text{-}ring\ (ring\text{-}of\ R)} (snd\ r)$   
**using** *result(1,2) assms(2,3)* **by** *simp algebra*  
**ultimately have** *long-divides*  $f\ g\ (fst\ r,\ snd\ r)$   
**using** *result(2)* **unfolding** *long-divides-def* **by** (*auto simp:mem-Times-iff*)  
**hence**  $(fst\ r,\ snd\ r) = (pdiv\ f\ g,\ pmod\ f\ g)$   
**by** (*intro long-divisionI[OF carrier-is-subfield] False assms*)  
**then show** *?thesis* **unfolding** *r-def* **by** *simp*  
**qed**  
**qed**

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

**lemma** *pdiv-c*:  
**assumes** *field<sub>C</sub>*  $R$   
**assumes**  $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$   
**assumes**  $g \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$   
**shows**  $pdiv_C\ R\ f\ g = ring.pdiv\ (ring\text{-}of\ R)\ f\ g$   
**unfolding** *pdiv<sub>C</sub>-def long-division-c[OF assms]* **by** *simp*

**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<sub>C</sub>*  $R$   
**assumes**  $f \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$   
**assumes**  $g \in carrier\ (poly\text{-}ring\ (ring\text{-}of\ R))$   
**shows**  $pmod_C\ R\ f\ g = ring.pmod\ (ring\text{-}of\ R)\ f\ g$   
**unfolding** *pmod<sub>C</sub>-def long-division-c[OF assms]* **by** *simp*

**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*  $F\ f\ g = ($   
 $\text{if } f = [] \vee g = [] \text{ then}$

```

      ((1C poly F, 1C poly F).f +C poly F g)
    else (
      let (p,q) = long-divisionC 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)))
    by pat-completeness auto

```

**termination**

```

apply (relation measure (λ(-, -, f). length f))
subgoal by simp
by (metis case-prod-conv in-measure length-greater-0-conv long-division-c-length
prod.sel(2))

```

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

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

**shows**  $x \text{ pdivides } x$

**proof** –

**interpret**  $d:\text{domain poly-ring } R$  **by** (rule univ-poly-is-domain[OF carrier-is-subring])

**show** ?thesis

**using** *assms* **unfolding** pdivides-def

**by** (intro dividesI[where  $c=1_{\text{poly-ring } R}$ ]) simp-all

qed

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

**lemma** ext-euclidean:

**assumes**  $\text{field}_C R$

**defines**  $P \equiv \text{poly-ring } (\text{ring-of } R)$

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

**assumes**  $g \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$

**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** –

**let** ?P=  $\text{poly-ring } (\text{ring-of } R)$

**interpret**  $\text{field ring-of } R$  **using** *assms*(1) **unfolding**  $\text{field}_C\text{-def}$  **by** auto

**interpret**  $d\text{-poly-ring: domain poly-ring } (\text{ring-of } R)$

**by** (rule univ-poly-is-domain[OF carrier-is-subring])

**have**  $\text{ring-c: ring}_C R$  **using** *assms*(1) **unfolding**  $\text{field}_C\text{-def}$   $\text{domain}_C\text{-def}$   $\text{cring}_C\text{-def}$  **by** auto

**have**  $d\text{-poly: domain}_C (\text{poly } R)$  **using** *assms* (1) **unfolding**  $\text{field}_C\text{-def}$

by (intro poly-domain) auto

have pdiv-zero:  $x \text{ pdivides}_{\text{ring-of } R} \mathbf{0}_{?P}$  if  $x \in \text{carrier } ?P$  for  $x$   
 using that unfolding univ-poly-zero by (intro pdivides-zero[OF carrier-is-subring])

have  $\text{snd } r = f \otimes_{?P} (\text{fst } (\text{fst } r)) \oplus_{?P} g \otimes_{?P} (\text{snd } (\text{fst } r)) \wedge$   
 $\text{snd } r \text{ pdivides}_{\text{ring-of } R} f \wedge \text{snd } r \text{ pdivides}_{\text{ring-of } R} g \wedge$   
 $\{\text{snd } r, \text{fst } (\text{fst } r), \text{snd } (\text{fst } r)\} \subseteq \text{carrier } ?P \wedge$   
 $(\text{snd } r = [] \longrightarrow f = [] \wedge g = [])$   
 if  $r = \text{ext-euclidean } R \ f \ g \ \{f, g\} \subseteq \text{carrier } ?P$   
 using that

proof (induction length g arbitrary: f g r rule:nat-less-induct)

case 1

have ind:

$\text{snd } s = x \otimes_{?P} \text{fst } (\text{fst } s) \oplus_{?P} y \otimes_{?P} \text{snd } (\text{fst } s)$   
 $\text{snd } s \text{ pdivides}_{\text{ring-of } R} x \text{ snd } s \text{ pdivides}_{\text{ring-of } R} y$   
 $\{\text{snd } s, \text{fst } (\text{fst } s), \text{snd } (\text{fst } s)\} \subseteq \text{carrier } ?P$   
 $(\text{snd } s = [] \longrightarrow x = [] \wedge y = [])$

if  $\text{length } y < \text{length } g \ s = \text{ext-euclidean } R \ x \ y \ \{x, y\} \subseteq \text{carrier } ?P$

for  $x \ y \ s$  using that 1(1) by metis+

show ?case

proof (cases  $f = [] \vee g = []$ )

case True

hence r-def:  $r = ((\mathbf{1}_{?P}, \mathbf{1}_{?P}), f \oplus_{?P} g)$  unfolding 1(2)

by (simp add: ext-euclidean.simps domain-cD[OF d-poly] ring-of-poly[OF ring-c])

consider  $f = \mathbf{0}_{?P} \mid g = \mathbf{0}_{?P}$

using True unfolding univ-poly-zero by auto

hence  $\text{snd } r \text{ pdivides}_{\text{ring-of } R} f \wedge \text{snd } r \text{ pdivides}_{\text{ring-of } R} g$

using 1(3) pdiv-zero pdivides-self unfolding r-def by cases

auto

moreover have  $\text{snd } r = f \otimes_{?P} \text{fst } (\text{fst } r) \oplus_{?P} g \otimes_{?P} \text{snd } (\text{fst } r)$

r)

using 1(3) unfolding r-def by simp

moreover have  $\{\text{snd } r, \text{fst } (\text{fst } r), \text{snd } (\text{fst } r)\} \subseteq \text{carrier } ?P$

using 1(3) unfolding r-def by auto

moreover have  $\text{snd } r = [] \longrightarrow f = [] \wedge g = []$

using 1(3) True unfolding r-def by (auto simp: univ-poly-zero)

ultimately show ?thesis by (intro conjI) metis+

next

case False

obtain  $p \ q$  where pq-def:  $(p, q) = \text{long-division}_C \ R \ f \ g$

by (metis surj-pair)

obtain  $u \ v \ s$  where uvs-def:  $((u, v), s) = \text{ext-euclidean } R \ g \ q$

by (metis surj-pair)

**have**  $(p,q) = (pdiv\ f\ g,\ pmod\ f\ g)$   
**using**  $1(\mathcal{B})$  **unfolding**  $pq\text{-def}$  **by**  $(intro\ long\text{-division}\text{-}c[OF\ assms(1)])\ auto$   
**hence**  $p\text{-def}: p = pdiv\ f\ g$  **and**  $q\text{-def}: q = pmod\ f\ g$  **by**  $auto$   
**have**  $p\text{-carr}: p \in carrier\ ?P$  **and**  $q\text{-carr}: q \in carrier\ ?P$   
**using**  $1(\mathcal{B})\ long\text{-division}\text{-}closed[OF\ carrier\text{-is}\text{-}subfield]$  **unfolding**  $p\text{-def}\ q\text{-def}$  **by**  $auto$

**have**  $length\ g > 0$  **using**  $False$  **by**  $auto$   
**hence**  $len\text{-}q: length\ q < length\ g$  **using**  $long\text{-division}\text{-}c\text{-}length\ pq\text{-def}$  **by**  $(metis\ snd\text{-}conv)$   
**have**  $s\text{-eq}: s = g \otimes_{?P} u \oplus_{?P} q \otimes_{?P} v$   
**and**  $s\text{-div}\text{-}g: s\ pdivides_{ring\text{-of}\ R} g$   
**and**  $s\text{-div}\text{-}q: s\ pdivides_{ring\text{-of}\ R} q$   
**and**  $suv\text{-}carr: \{s,u,v\} \subseteq carrier\ ?P$   
**and**  $s\text{-zero}\text{-}iff: s = [] \longrightarrow g = [] \wedge q = []$   
**using**  $ind[OF\ len\text{-}q\ uvs\text{-}def\ -]\ q\text{-carr}\ 1(\mathcal{B})$  **by**  $auto$

**have**  $r = ((v,u +_C\ poly\ R\ (-_C\ poly\ R\ (p *_C\ poly\ R\ v))),s)$  **unfolding**  $1(2)$  **using**  $False$   
**by**  $(subst\ ext\text{-}euclidean.\text{simps})\ (simp\ add: pq\text{-def}[symmetric])\ uvs\text{-}def[symmetric]$   
**also** **have**  $\dots = ((v, u \ominus_{?P} (p \otimes_{?P} v)), s)$  **using**  $p\text{-carr}\ suv\text{-}carr\ domain\text{-}cD[OF\ d\text{-}poly]$   
**unfolding**  $a\text{-minus}\text{-}def\ ring\text{-of}\text{-}poly[OF\ ring\text{-}c]$  **by**  $(intro\ arg\text{-}cong2[where\ f=Pair]\ refl)\ simp$   
**finally** **have**  $r\text{-def}: r = ((v, u \ominus_{?P} (p \otimes_{?P} v)), s)$  **by**  $simp$

**have**  $snd\ r = g \otimes_{?P} u \oplus_{?P} q \otimes_{?P} v$  **unfolding**  $r\text{-def}\ s\text{-eq}$  **by**  $simp$   
**also** **have**  $\dots = g \otimes_{?P} u \oplus_{?P} (f \ominus_{?P} g \otimes_{?P} p) \otimes_{?P} v$   
**using**  $1(\mathcal{B})\ p\text{-carr}\ q\text{-carr}\ suv\text{-}carr$   
**by**  $(subst\ pdiv\text{-}pmod[OF\ carrier\text{-is}\text{-}subfield,\ of\ f\ g])\ (simp\text{-}all\ add:p\text{-def}[symmetric]\ q\text{-def}[symmetric],\ algebra)$   
**also** **have**  $\dots = f \otimes_{?P} v \oplus_{?P} g \otimes_{?P} (u \ominus_{?P} ((p \otimes_{?P} v)))$   
**using**  $1(\mathcal{B})\ p\text{-carr}\ q\text{-carr}\ suv\text{-}carr$  **by**  $simp\ algebra$   
**finally** **have**  $r1: snd\ r = f \otimes_{?P} fst\ (fst\ r) \oplus_{?P} g \otimes_{?P} snd\ (fst\ r)$   
**unfolding**  $r\text{-def}$  **by**  $simp$   
**have**  $pmod\ f\ s = pmod\ (g \otimes_{?P} p \oplus_{?P} q)\ s$  **using**  $1(\mathcal{B})$   
**by**  $(subst\ pdiv\text{-}pmod[OF\ carrier\text{-is}\text{-}subfield,\ of\ f\ g])\ (simp\text{-}all\ add:p\text{-def}[symmetric]\ q\text{-def}[symmetric])$   
**also** **have**  $\dots = pmod\ (g \otimes_{?P} p)\ s \oplus_{?P} pmod\ q\ s$   
**using**  $1(\mathcal{B})\ p\text{-carr}\ q\text{-carr}\ suv\text{-}carr$   
**by**  $(subst\ long\text{-division}\text{-}add[OF\ carrier\text{-is}\text{-}subfield])\ simp\text{-}all$   
**also** **have**  $\dots = pmod\ (pmod\ g\ s \otimes_{?P} p)\ s \oplus_{?P} []$   
**using**  $1(\mathcal{B})\ p\text{-carr}\ q\text{-carr}\ suv\text{-}carr\ s\text{-div}\text{-}q$   
**by**  $(intro\ arg\text{-}cong2[where\ f=(\oplus_{?P})]\ pmod\text{-}mult\text{-}left)\ (simp\text{-}all\ add: pmod\text{-}zero\text{-}iff\text{-}pdivides[OF\ carrier\text{-is}\text{-}subfield])$

```

    also have ... = pmod (0 ?P ⊗ ?P p) s ⊕ ?P 0 ?P unfolding
univ-poly-zero
    using 1(3) p-carr q-carr suv-carr s-div-g by (intro arg-cong2[where
f=(⊕ ?P)])
        arg-cong2[where f=(⊗ ?P)] arg-cong2[where f=pmod]
        (simp-all add: pmod-zero-iff-pdivides[OF carrier-is-subfield])
    also have ... = pmod 0 ?P s
    using p-carr suv-carr long-division-closed[OF carrier-is-subfield]
by simp
    also have ... = [] unfolding univ-poly-zero
    using suv-carr long-division-zero(2)[OF carrier-is-subfield] by
simp
    finally have pmod f s = [] by simp
    hence r2: snd r pdividesring-of R f using suv-carr 1(3) unfold-
ing r-def
    by (subst pmod-zero-iff-pdivides[OF carrier-is-subfield,symmetric])
simp-all
    have r3: snd r pdividesring-of R g unfolding r-def using s-div-g
by auto
    have r4: {snd r, fst (fst r), snd (fst r)} ⊆ carrier ?P
    using suv-carr p-carr unfolding r-def by simp-all
    have r5: f = [] ∧ g = [] if snd r = []
    proof -
    have r5-a: g = [] ∧ q = [] using that s-zero-iff unfolding r-def
by simp
    hence pmod f [] = [] unfolding q-def by auto
    hence f = [] using pmod-def by simp
    thus ?thesis using r5-a by auto
    qed
    show ?thesis using r1 r2 r3 r4 r5 by (intro conjI) metis+
    qed
    thus ?T1 ?T2 ?T3 ?T4 ?T5 using assms by auto
    qed
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 = (λx. x < n),
  idx-uminus = (λx. (n-x) mod n),
  idx-plus = (λx y. (x+y) mod n),

```

```

    idx-udivide = (λx. nat (fst (bezout-coefficients (int x) (int n)) mod
(int n))),
    idx-mult = (λx 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  $zfact\text{-}iso\ n\ 0 = \mathbf{0}_{ZFact\ (int\ n)}$ 

```

**proof** –

```

  let  $?I = Idl_{\mathcal{Z}}\ \{int\ n\}$ 
  have ideal-I: ideal  $?I\ \mathcal{Z}$ 
    by (simp add: int.genideal-ideal)

```

```

  interpret i:ideal  $?I\ \mathcal{Z}$  using ideal-I by simp
  interpret s:ring-hom-ring  $\mathcal{Z}\ ZFact\ (int\ n)\ (+>_{\mathcal{Z}})\ ?I$ 
  using i.rcos-ring-hom-ring ZFact-def by auto

```

```

  show ?thesis
    by (simp add:zfact-iso-def ZFact-def)

```

**qed**

**lemma** *zfact-prime-is-field*:

```

  assumes Factorial-Ring.prime ( $p :: nat$ )
  shows field ( $ZFact\ (int\ p)$ )
  using zfact-prime-is-finite-field[OF assms] finite-field-def by auto

```

**definition** *zfact-iso-inv* ::  $nat \Rightarrow int\ set \Rightarrow nat\ \mathbf{where}$

```

zfact-iso-inv  $p = the\text{-}inv\text{-}into\ \{..\lt p\}\ (zfact\text{-}iso\ p)$ 

```

**lemma** *zfact-iso-inv-0*:

```

  assumes n-ge-0:  $n > 0$ 
  shows  $zfact\text{-}iso\text{-}inv\ n\ \mathbf{0}_{ZFact\ (int\ n)} = 0$ 
  unfolding zfact-iso-inv-def zfact-iso-0[OF n-ge-0, symmetric] using
n-ge-0
  by (rule the-inv-into-f-f[OF zfact-iso-inj], simp add:mod-ring-def)

```

**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  $x = I\ +>_{\mathcal{Z}}\ (int\ (zfact\text{-}iso\text{-}inv\ n\ x))$ 
proof –
  have  $x \in zfact\text{-}iso\ n\ \{..\lt n\}$ 
    using assms zfact-iso-ran by simp

```

**hence**  $zfact\text{-}iso\ n\ (zfact\text{-}iso\text{-}inv\ n\ x) = x$   
**unfolding**  $zfact\text{-}iso\text{-}inv\text{-}def$  **by**  $(intro\ f\text{-}the\text{-}inv\text{-}into\text{-}f\ zfact\text{-}iso\text{-}inj)$   
*auto*  
**thus** *?thesis* **unfolding**  $zfact\text{-}iso\text{-}def\ I\text{-}def$  **by** *blast*  
**qed**

**lemma**  $zfact\text{-}iso\text{-}inv\text{-}bij$ :  
**assumes**  $n > 0$   
**shows**  $bij\text{-}betw\ (zfact\text{-}iso\text{-}inv\ n)\ (carrier\ (ZFact\ (int\ n)))\ (carrier\ (ring\text{-}of\ (mod\text{-}ring\ n)))$   
**proof** –  
**have**  $bij\text{-}betw\ (the\text{-}inv\text{-}into\ \{..\lt n\}\ (zfact\text{-}iso\ n))\ (carrier\ (ZFact\ (int\ n)))\ \{..\lt n\}$   
**by**  $(intro\ bij\text{-}betw\text{-}the\text{-}inv\text{-}into\ zfact\text{-}iso\text{-}bij[OF\ assms])$   
**thus** *?thesis*  
**unfolding**  $zfact\text{-}iso\text{-}inv\text{-}def\ mod\text{-}ring\text{-}def\ ring\text{-}of\text{-}def\ lessThan\text{-}def$   
*by simp*  
**qed**

**lemma**  $zfact\text{-}iso\text{-}inv\text{-}is\text{-}ring\text{-}iso$ :  
**fixes**  $n :: nat$   
**assumes**  $n\text{-}ge\ 1: n > 1$   
**shows**  $zfact\text{-}iso\text{-}inv\ n \in ring\text{-}iso\ (ZFact\ (int\ n))\ (ring\text{-}of\ (mod\text{-}ring\ n))\ (is\ ?f \in \text{-})$   
**proof**  $(rule\ ring\text{-}iso\text{-}memI)$   
**interpret**  $r:cring\ (ZFact\ (int\ n))$   
**using**  $ZFact\text{-}is\text{-}cring$  **by** *simp*

**define**  $I$  **where**  $I = Idl_{\mathcal{Z}}\ \{int\ n\}$

**have**  $n\text{-}ge\ 0: n > 0$  **using**  $n\text{-}ge\ 1$  **by** *simp*

**interpret**  $i:ideal\ I\ \mathcal{Z}$   
**unfolding**  $I\text{-}def$  **using**  $int.\text{genideal}\text{-}ideal$  **by** *simp*

**interpret**  $s:ring\text{-}hom\text{-}ring\ \mathcal{Z}\ ZFact\ (int\ n)\ (+>_{\mathcal{Z}})\ I$   
**using**  $i.\text{rcos}\text{-}ring\text{-}hom\text{-}ring\ ZFact\text{-}def\ I\text{-}def$  **by** *auto*

**show**  $zfact\text{-}iso\text{-}inv\ n\ x \in carrier\ (ring\text{-}of\ (mod\text{-}ring\ n))$  **if**  $x \in carrier\ (ZFact\ (int\ n))$  **for**  $x$

**proof** –  
**have**  $zfact\text{-}iso\text{-}inv\ n\ x \in \{..\lt n\}$   
**unfolding**  $zfact\text{-}iso\text{-}inv\text{-}def$  **using**  $that\ zfact\text{-}iso\text{-}ran[OF\ n\text{-}ge\ 0]$   
**by**  $(intro\ the\text{-}inv\text{-}into\text{-}into\ zfact\text{-}iso\text{-}inj\ n\text{-}ge\ 0)\ auto$   
**thus**  $zfact\text{-}iso\text{-}inv\ n\ x \in carrier\ (ring\text{-}of\ (mod\text{-}ring\ n))$   
**by**  $(simp\ add:ring\text{-}of\text{-}def\ mod\text{-}ring\text{-}def)$

**qed**

**show**  $?f\ (x \otimes_{ZFact\ (int\ n)} y) = ?f\ x \otimes_{ring\text{-}of\ (mod\text{-}ring\ n)} ?f\ y$

**if**  $x\text{-carr}: x \in \text{carrier } (\text{ZFact } (\text{int } n))$  **and**  $y\text{-carr}: y \in \text{carrier } (\text{ZFact } (\text{int } n))$  **for**  $x\ y$   
**proof** –  
**define**  $x'$  **where**  $x' = \text{zfact-iso-inv } n\ x$   
**define**  $y'$  **where**  $y' = \text{zfact-iso-inv } n\ y$   
**have**  $x \otimes_{\text{ZFact } (\text{int } n)} y = (I \text{ +> }_{\mathcal{Z}} (\text{int } x^{\wedge})) \otimes_{\text{ZFact } (\text{int } n)} (I \text{ +> }_{\mathcal{Z}} (\text{int } y^{\wedge}))$   
**unfolding**  $x'\text{-def } y'\text{-def}$   
**using**  $x\text{-carr } y\text{-carr } \text{zfact-coset}[OF\ n\text{-ge-}0] I\text{-def}$  **by**  $\text{simp}$   
**also have**  $\dots = (I \text{ +> }_{\mathcal{Z}} (\text{int } x' * \text{int } y'))$   
**by**  $\text{simp}$   
**also have**  $\dots = (I \text{ +> }_{\mathcal{Z}} (\text{int } ((x' * y') \bmod n)))$   
**unfolding**  $I\text{-def } \text{zmod-int}$  **by**  $(\text{rule } \text{int-cosetI}[OF\ n\text{-ge-}0], \text{simp})$   
**also have**  $\dots = (I \text{ +> }_{\mathcal{Z}} (x' \otimes_{\text{ring-of } (\text{mod-ring } n)} y'))$   
**unfolding**  $\text{ring-of-def } \text{mod-ring-def}$  **by**  $\text{simp}$   
**also have**  $\dots = \text{zfact-iso } n\ (x' \otimes_{\text{ring-of } (\text{mod-ring } n)} y')$   
**unfolding**  $\text{zfact-iso-def } I\text{-def}$  **by**  $\text{simp}$   
**finally have**  $a: x \otimes_{\text{ZFact } (\text{int } n)} y = \text{zfact-iso } n\ (x' \otimes_{\text{ring-of } (\text{mod-ring } n)} y')$   
**by**  $\text{simp}$   
**have**  $b: x' \otimes_{\text{ring-of } (\text{mod-ring } n)} y' \in \{..<n\}$   
**using**  $\text{mod-ring-def } n\text{-ge-}0$  **by**  $(\text{auto } \text{simp:ring-of-def})$   
**have**  $?f (\text{zfact-iso } n\ (x' \otimes_{\text{ring-of } (\text{mod-ring } n)} y')) = x' \otimes_{\text{ring-of } (\text{mod-ring } n)} y'$   
**unfolding**  $\text{zfact-iso-inv-def}$   
**by**  $(\text{rule } \text{the-inv-into-f-f}[OF\ \text{zfact-iso-inj}[OF\ n\text{-ge-}0] b])$   
**thus**  
 $\text{zfact-iso-inv } n\ (x \otimes_{\text{ZFact } (\text{int } n)} y) =$   
 $\text{zfact-iso-inv } n\ x \otimes_{\text{ring-of } (\text{mod-ring } n)} \text{zfact-iso-inv } n\ y$   
**using**  $a\ x'\text{-def } y'\text{-def}$  **by**  $\text{simp}$   
**qed**

**show**  $\text{zfact-iso-inv } n\ (x \oplus_{\text{ZFact } (\text{int } n)} y) =$   
 $\text{zfact-iso-inv } n\ x \oplus_{\text{ring-of } (\text{mod-ring } n)} \text{zfact-iso-inv } n\ y$   
**if**  $x\text{-carr}: x \in \text{carrier } (\text{ZFact } (\text{int } n))$  **and**  $y\text{-carr}: y \in \text{carrier } (\text{ZFact } (\text{int } n))$  **for**  $x\ y$   
**proof** –  
**define**  $x'$  **where**  $x' = \text{zfact-iso-inv } n\ x$   
**define**  $y'$  **where**  $y' = \text{zfact-iso-inv } n\ y$   
**have**  $x \oplus_{\text{ZFact } (\text{int } n)} y = (I \text{ +> }_{\mathcal{Z}} (\text{int } x^{\wedge})) \oplus_{\text{ZFact } (\text{int } n)} (I \text{ +> }_{\mathcal{Z}} (\text{int } y^{\wedge}))$   
**unfolding**  $x'\text{-def } y'\text{-def}$   
**using**  $x\text{-carr } y\text{-carr } \text{zfact-coset}[OF\ n\text{-ge-}0] I\text{-def}$  **by**  $\text{simp}$   
**also have**  $\dots = (I \text{ +> }_{\mathcal{Z}} (\text{int } x' + \text{int } y'))$   
**by**  $\text{simp}$   
**also have**  $\dots = (I \text{ +> }_{\mathcal{Z}} (\text{int } ((x' + y') \bmod n)))$   
**unfolding**  $I\text{-def } \text{zmod-int}$  **by**  $(\text{rule } \text{int-cosetI}[OF\ n\text{-ge-}0], \text{simp})$   
**also have**  $\dots = (I \text{ +> }_{\mathcal{Z}} (x' \oplus_{\text{ring-of } (\text{mod-ring } n)} y'))$

```

    unfolding mod-ring-def ring-of-def by simp
    also have ... = zfact-iso n (x' ⊕ring-of (mod-ring n) y∧)
    unfolding zfact-iso-def I-def by simp
    finally have a:x ⊕ZFact (int n) y = zfact-iso n (x' ⊕ring-of (mod-ring n)
y∧)
    by simp
    have b:x' ⊕ring-of (mod-ring n) y' ∈ {..n}
    using mod-ring-def n-ge-0 by (auto simp:ring-of-def)
    have ?f (zfact-iso n (x' ⊕ring-of (mod-ring n) y∧)) = x' ⊕ring-of (mod-ring n)
y'
    unfolding zfact-iso-inv-def
    by (rule the-inv-into-f-f[OF zfact-iso-inj[OF n-ge-0] b])
    thus ?f (x ⊕ZFact (int n) y) = ?f x ⊕ring-of (mod-ring n) ?f y
    using a x'-def y'-def by simp
qed

have 1ZFact (int n) = zfact-iso n (1ring-of (mod-ring n))
  by (simp add:zfact-iso-def ZFact-def I-def[symmetric] ring-of-def
mod-ring-def)

thus zfact-iso-inv n 1ZFact (int n) = 1ring-of (mod-ring n)
  unfolding zfact-iso-inv-def mod-ring-def ring-of-def
  using the-inv-into-f-f[OF zfact-iso-inj] n-ge-1 by simp

show bij-betw (zfact-iso-inv n) (carrier (ZFact (int n))) (carrier
(ring-of (mod-ring n)))
  by (intro zfact-iso-inv-bij n-ge-0)
qed

lemma mod-ring-finite:
  finite (carrier (ring-of (mod-ring n)))
  by (simp add:mod-ring-def ring-of-def)

lemma mod-ring-carr:
  x ∈ carrier (ring-of (mod-ring n)) ↔ x < n
  by (simp add:mod-ring-def ring-of-def)

lemma mod-ring-is-cring:
  assumes n-ge-1: n > 1
  shows cring (ring-of (mod-ring n))
proof -
  have n-ge-0: n > 0 using n-ge-1 by simp

  interpret cring ZFact (int n)
    using ZFact-is-cring by simp

  have cring ((ring-of (mod-ring n)) (| zero := zfact-iso-inv n 0ZFact (int n)
|))

```

**by** (rule ring-iso-imp-imp-cring[OF zfact-iso-inv-is-ring-iso[OF n-ge-1]])  
**moreover have**  
ring-of (mod-ring n)  $\parallel$  zero := zfact-iso-inv n  $\mathbf{0}_{ZFact (int n)}$   $\parallel$  =  
ring-of (mod-ring n)  
**using** zfact-iso-inv-0[OF n-ge-0] **by** (simp add:mod-ring-def ring-of-def)  
**ultimately show** ?thesis **by** simp  
**qed**

**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** –  
**have** r:ring (ZFact (int n))  
**using** ZFact-is-cring cring.axioms(1) **by** blast

**interpret** s: ring (ring-of (mod-ring n))  
**using** mod-ring-is-cring cring.axioms(1) n-ge-1 **by** blast  
**have** n-ge-0:  $n > 0$  **using** n-ge-1 **by** linarith

**have** inv-into (carrier (ZFact (int n))) (zfact-iso-inv n)  
 $\in$  ring-iso (ring-of (mod-ring n)) (ZFact (int n))  
**using** ring-iso-set-sym[OF r zfact-iso-inv-is-ring-iso[OF n-ge-1]]  
**by** simp  
**moreover have** inv-into (carrier (ZFact (int n))) (zfact-iso-inv n)  
 $x =$  zfact-iso n x  
**if**  $x \in$  carrier (ring-of (mod-ring n)) **for** x  
**proof** –  
**have**  $x \in \{..<n\}$  **using** that **by** (simp add:mod-ring-def ring-of-def)  
**thus** inv-into (carrier (ZFact (int n))) (zfact-iso-inv n)  $x =$  zfact-iso  
n x  
**using** zfact-iso-inv-bij[OF n-ge-0] zfact-iso-bij[OF n-ge-0] **un-**  
**folding** zfact-iso-inv-def  
**by** (intro inv-into-f-eq bij-betw-apply[OF zfact-iso-inv-bij[OF  
n-ge-0]] the-inv-into-f-f)  
(auto intro:bij-betw-imp-inj-on simp:bij-betwE)  
**qed**

**ultimately show** ?thesis **using** s.ring-iso-restrict **by** blast  
**qed**

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** –  
**have** p-ge-0:  $p > 0$  **using** assms prime-gt-0-nat **by** blast  
**have** p-ge-1:  $p > 1$  **using** assms prime-gt-1-nat **by** blast  
  
**interpret** field ZFact (int p)

```

using zfact-prime-is-field[OF assms] by simp

have field ((ring-of (mod-ring p)) (| zero := zfact-iso-inv p 0ZFact (int p)
|))
by (rule ring-iso-imp-imp-field[OF zfact-iso-inv-is-ring-iso[OF p-ge-1]])

moreover have
  (ring-of (mod-ring p)) (| zero := zfact-iso-inv p 0ZFact (int p) |) =
ring-of (mod-ring p)
using zfact-iso-inv-0[OF p-ge-0] by (simp add:mod-ring-def ring-of-def)
ultimately show ?thesis by simp
qed

```

```

lemma mod-ring-is-ring-c:
  assumes n > 1
  shows cringC (mod-ring n)
proof (intro cring-cI mod-ring-is-cring assms)
  fix x
  assume a:x ∈ carrier (ring-of (mod-ring n))
  hence x-le-n: x < n unfolding mod-ring-def ring-of-def by simp

  interpret cring (ring-of (mod-ring n)) by (intro mod-ring-is-cring
assms)

```

```

  show  $-_C$  mod-ring n x =  $\ominus$ ring-of (mod-ring n) x using x-le-n
  by (intro minus-equality[symmetric] a) (simp-all add:ring-of-def
mod-ring-def mod-simps)

```

```

next
fix x
assume a:x ∈ Units (ring-of (mod-ring n))

```

```

let ?l = fst (bezout-coefficients (int x) (int n))
let ?r = snd (bezout-coefficients (int x) (int n))

```

```

interpret cring ring-of (mod-ring n) by (intro mod-ring-is-cring
assms)

```

```

obtain y where x  $\otimes$ ring-of (mod-ring n) y = 1ring-of (mod-ring n)
using a by (meson Units-r-inv-ex)
hence x * y mod n = 1 by (simp-all add:mod-ring-def ring-of-def)
hence gcd x n = 1 by (metis dvd-triv-left gcd.assoc gcd-1-nat gcd-nat.absorb-iff1
gcd-red-nat)
hence 0:gcd (int x) (int n) = 1 unfolding gcd-int-int-eq by simp

```

```

have int x * ?l mod int n = (?l * int x + ?r * int n) mod int n
using assms by (simp add:mod-simps algebra-simps)
also have ... = (gcd (int x) (int n)) mod int n
by (intro arg-cong2[where f=(mod)] refl bezout-coefficients) simp
also have ... = 1 unfolding 0 using assms by simp

```

```

finally have  $\text{int } x * ?l \text{ mod int } n = 1$  by simp
hence  $\text{int } x * \text{nat } (\text{fst } (\text{bezout-coefficients } (\text{int } x) (\text{int } n)) \text{ mod int } n) \text{ mod } n = 1$ 
using assms by (simp add:mod-simps)
hence  $x * \text{nat } (\text{fst } (\text{bezout-coefficients } (\text{int } x) (\text{int } n)) \text{ mod int } n) \text{ mod } n = 1$ 
by (metis nat-mod-as-int nat-one-as-int of-nat-mult)
hence  $x \otimes_{\text{ring-of } (\text{mod-ring } n)} x^{-1} \text{ mod-ring } n = \mathbf{1}_{\text{ring-of } (\text{mod-ring } n)}$ 
using assms unfolding mod-ring-def ring-of-def by simp
moreover have  $\text{nat } (\text{fst } (\text{bezout-coefficients } (\text{int } x) (\text{int } n)) \text{ mod int } n) < n$ 
using assms by (subst nat-less-iff) auto
hence  $x^{-1} \text{ mod-ring } n \in \text{carrier } (\text{ring-of } (\text{mod-ring } n))$ 
using assms unfolding mod-ring-def ring-of-def by simp
moreover have  $x \in \text{carrier } (\text{ring-of } (\text{mod-ring } n))$  using a by auto
ultimately show  $x^{-1} \text{ mod-ring } n = \text{inv}_{\text{ring-of } (\text{mod-ring } n)} x$ 
by (intro comm-inv-char[symmetric])
qed

```

```

lemma mod-ring-is-field-c:
assumes Factorial-Ring.prime p
shows  $\text{field}_C (\text{mod-ring } p)$ 
unfolding field_C-def domain_C-def
by (intro conjI mod-ring-is-ring-c mod-ring-is-field assms prime-gt-1-nat domain.axioms(1) field.axioms(1))

```

```

lemma mod-ring-is-enum-c:
shows  $\text{enum}_C (\text{mod-ring } n)$ 
by (intro enum-cI (simp-all add:mod-ring-def ring-of-def Coset.order-def lessThan-def))

```

**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  $\text{pcoprime}_C :: ('a, 'b) \text{idx-ring-scheme} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow \text{bool}$ 
  where  $\text{pcoprime}_C R f g = (\text{length } (\text{snd } (\text{ext-euclidean } R f g)) = 1)$ 

declare  $\text{pcoprime}_C.\text{simps}[simp \text{ del}]$ 

```

```

lemma pcoprime-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  $\text{pcoprime}_C R f g \longleftrightarrow \text{pcoprime}_{\text{ring-of } R} f g$  (is ?L = ?R)
proof (cases  $f = [] \wedge g = []$ )
  case True
  interpret field ring-of R
    using assms(1) unfolding fieldC-def by simp
  interpret d-poly-ring: domain poly-ring (ring-of R)
    by (rule univ-poly-is-domain[OF carrier-is-subring])

  have ?L = False using True by (simp add: pcoprimeC.simps ext-euclidean.simps
poly-def)
  also have ...  $\longleftrightarrow (\text{length } \mathbf{0}_{\text{poly-ring } (\text{ring-of } R)} = 1)$  by (simp
add:univ-poly-zero)
  also have ...  $\longleftrightarrow \text{pcoprime}_{\text{ring-of } R} \mathbf{0}_{\text{poly-ring } (\text{ring-of } R)}$   $\square$ 
    by (subst pcoprime-zero-iff) (simp-all)
  also have ...  $\longleftrightarrow ?R$  using True by (simp add: univ-poly-zero)
  finally show ?thesis by simp
next
  case False

  let ?P = poly-ring (ring-of R)
  interpret field ring-of R
    using assms(1) unfolding fieldC-def by simp
  interpret d-poly-ring: domain poly-ring (ring-of R)
    by (rule univ-poly-is-domain[OF carrier-is-subring])

  obtain s u v where su-def: ((u,v),s) = ext-euclidean R f g by
(metis surj-pair)

  have s-eq: s = f  $\otimes_{?P} u \oplus_{?P} g \otimes_{?P} v$  (is ?T1)
  and s-div-f: s pdividesring-of R f and s-div-g: s pdividesring-of R g
(is ?T3)
  and su-carr: {s, u, v}  $\subseteq$  carrier ?P
  and s-nz: s  $\neq$  []
  using False su-def[symmetric] ext-euclidean[OF assms(1,2,3)] by
auto

  have ?L  $\longleftrightarrow \text{length } s = 1$  using su-def[symmetric] by (simp
add:pcoprimeC.simps)
  also have ...  $\longleftrightarrow ?R$ 
  unfolding pcoprime-def
proof (intro iffI impI ballI)
  fix r assume len-s: length s = 1
  assume r-carr: r  $\in$  carrier ?P
  and r pdividesring-of R f  $\wedge$  r pdividesring-of R g
  hence r-div: pmod f r =  $\mathbf{0}_{?P}$  pmod g r =  $\mathbf{0}_{?P}$  unfolding

```

```

univ-poly-zero
  using assms(2,3) pmod-zero-iff-pdivides[OF carrier-is-subfield]
by auto

  have pmod s r = pmod (f ⊗?P u) r ⊕?P pmod (g ⊗?P v) r
  using r-carr suv-carr assms unfolding s-eq
  by (intro long-division-add[OF carrier-is-subfield]) auto
  also have ... = pmod (pmod f r ⊗?P u) r ⊕?P pmod (pmod g r
⊗?P v) r
  using r-carr suv-carr assms by (intro arg-cong2[where f=(⊕?P)]
pmod-mult-left) auto
  also have ... = pmod 0?P r ⊕?P pmod 0?P r
  using suv-carr unfolding r-div by simp
  also have ... = [] using r-carr unfolding univ-poly-zero
  by (simp add: long-division-zero[OF carrier-is-subfield] univ-poly-add)
  finally have pmod s r = [] by simp
  hence r pdividesring-of R s
  using r-carr suv-carr pmod-zero-iff-pdivides[OF carrier-is-subfield]
by auto
  hence degree r ≤ degree s
  using s-nz r-carr suv-carr by (intro pdivides-imp-degree-le[OF
carrier-is-subring]) auto
  thus degree r = 0 using len-s by simp
next
  assume ∀ r ∈ carrier ?P. r pdividesring-of R f ∧ r pdividesring-of R
g ⟶ degree r = 0
  hence degree s = 0 using s-div-f s-div-g suv-carr by simp
  thus length s = 1 using s-nz
  by (metis diff-is-0-eq diffs0-imp-equal length-0-conv less-one
linorder-le-less-linear)
qed
  finally show ?thesis by simp
qed

```

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
1Cpoly F)
    in pmodC F (r *Cpoly 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 ∈ carrier (poly-ring (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 (induction n rule:nat-less-induct)
  case (1 n)

  let  $?P = \text{poly-ring } (\text{ring-of } R)$ 
  interpret field ring-of R
    using  $\text{assms}(1)$  unfolding fieldC-def by simp
  interpret d-poly-ring: domain poly-ring (ring-of R)
    by (rule univ-poly-is-domain[OF carrier-is-subring])

  have ring-c: ringC R using  $\text{assms}(1)$  unfolding fieldC-def domainC-def cringC-def by auto
  have d-poly: domainC (poly R) using  $\text{assms}(1)$  unfolding fieldC-def by (intro poly-domain) auto

  have ind: pmod-powC R f m g = pmod (f [\wedge]_{?P} m) g if  $m < n$  for  $m$ 
    using 1 that by auto

  define  $r$  where  $r = (\text{if } n \geq 2 \text{ then } \text{pmod-pow}_C R f (n \text{ div } 2) g \wedge_{?P} \widehat{?P} \text{poly } R \ 2 \text{ else } 1_C \text{poly } R)$ 

  have  $\text{pmod } r g = \text{pmod } (f [\wedge]_{?P} (n - (n \text{ mod } 2))) g \wedge r \in \text{carrier } ?P$ 
  proof (cases n ≥ 2)
    case True
      hence  $r = \text{pmod-pow}_C R f (n \text{ div } 2) g [\wedge]_{?P} (2 :: \text{nat})$ 
      unfolding r-def domain-cD[OF d-poly] by (simp add:ring-of-poly[OF ring-c])
      also have  $\dots = \text{pmod } (f [\wedge]_{?P} (n \text{ div } 2)) g [\wedge]_{?P} (2 :: \text{nat})$ 
        using True by (intro arg-cong2[where f=([\wedge]_{?P}) refl ind) auto
      finally have  $r\text{-alt}: r = \text{pmod } (f [\wedge]_{?P} (n \text{ div } 2)) g [\wedge]_{?P} (2 :: \text{nat})$ 
        by simp

      have  $\text{pmod } r g = \text{pmod } (\text{pmod } (f [\wedge]_{?P} (n \text{ div } 2)) g \otimes_{?P} \text{pmod } (f [\wedge]_{?P} (n \text{ div } 2)) g) g$ 
        unfolding r-alt using  $\text{assms}(2,3)$  long-division-closed[OF carrier-is-subfield]
        by (simp add:numeral-eq-Suc) algebra
      also have  $\dots = \text{pmod } (f [\wedge]_{?P} (n \text{ div } 2) \otimes_{?P} f [\wedge]_{?P} (n \text{ div } 2)) g$ 
        using  $\text{assms}(2,3)$  by (intro pmod-mult-both[symmetric]) auto
      also have  $\dots = \text{pmod } (f [\wedge]_{?P} ((n \text{ div } 2) + (n \text{ div } 2))) g$ 
        using  $\text{assms}(2,3)$  by (subst d-poly-ring.nat-pow-mult) auto
      also have  $\dots = \text{pmod } (f [\wedge]_{?P} (n - (n \text{ mod } 2))) g$ 
        by (intro arg-cong2[where f=pmod] refl arg-cong2[where f=([\wedge]_{?P})])
    presburger
    finally have  $\text{pmod } r g = \text{pmod } (f [\wedge]_{?P} (n - (n \text{ mod } 2))) g$ 

```

```

    by simp
    moreover have  $r \in \text{carrier } ?P$ 
    using assms(2,3) long-division-closed[OF carrier-is-subfield] un-
folding r-alt by auto
    ultimately show ?thesis by auto
  next
  case False
  hence  $r = 1_{?P}$ 
    unfolding r-def using domain-cD[OF d-poly] ring-of-poly[OF
ring-c] by simp
    also have  $\dots = f \ [\ ]_{?P} (0 :: \text{nat})$  by simp
    also have  $\dots = f \ [\ ]_{?P} (n - (n \bmod 2))$ 
      using False by (intro arg-cong2[where f=(\ [\ ]_{?P}) refl) auto
    finally have  $r = f \ [\ ]_{?P} (n - (n \bmod 2))$  by simp
    then show ?thesis using assms(2) by simp
  qed

```

hence *r-exp*:  $\text{pmod } r \ g = \text{pmod } (f \ [\ ]_{?P} (n - (n \bmod 2))) \ g$  and  
*r-carr*:  $r \in \text{carrier } ?P$   
 by *auto*

```

  have  $\text{pmod-pow}_C \ R \ f \ n \ g = \text{pmod}_C \ R \ (r \ *_C \text{poly } R \ (f \ \widehat{\ }_C \text{poly } R \ (n$ 
 $\text{mod } 2))) \ g$ 
    by (subst pmod-pow_C.simps) (simp add:r-def[symmetric])
  also have  $\dots = \text{pmod}_C \ R \ (r \ \otimes_{?P} (f \ [\ ]_{?P} (n \bmod 2))) \ g$ 
    unfolding domain-cD[OF d-poly] by (simp add:ring-of-poly[OF
ring-c])
  also have  $\dots = \text{pmod } (r \ \otimes_{?P} (f \ [\ ]_{?P} (n \bmod 2))) \ g$ 
    using r-carr assms(2,3) by (intro pmod-c[OF assms(1)]) auto
  also have  $\dots = \text{pmod } (\text{pmod } r \ g \ \otimes_{?P} (f \ [\ ]_{?P} (n \bmod 2))) \ g$ 
    using r-carr assms(2,3) by (intro pmod-mult-left) auto
  also have  $\dots = \text{pmod } (f \ [\ ]_{?P} (n - (n \bmod 2)) \ \otimes_{?P} (f \ [\ ]_{?P} (n$ 
 $\text{mod } 2))) \ g$ 
    using assms(2,3) unfolding r-exp by (intro pmod-mult-left[symmetric])
  auto
  also have  $\dots = \text{pmod } (f \ [\ ]_{?P} ((n - (n \bmod 2)) + (n \bmod 2))) \ g$ 
    using assms(2,3) by (intro arg-cong2[where f=pmod] refl d-poly-ring.nat-pow-mult)
  auto
  also have  $\dots = \text{pmod } (f \ [\ ]_{?P} n) \ g$  by simp
  finally show  $\text{pmod-pow}_C \ R \ f \ n \ g = \text{pmod } (f \ [\ ]_{?P} n) \ g$  by simp
  qed

```

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

**definition** *pcoprime-with-gauss-poly* ::  $(\ 'a, 'b) \ \text{id}x\text{-ring-scheme} \Rightarrow \ 'a \ \text{list} \Rightarrow \ \text{nat} \Rightarrow \ \text{bool}$

```

  where pcoprime-with-gauss-poly  $F \ p \ n =$ 
    (pcoprimeC  $F \ p \ (\text{pmod-pow}_C \ F \ X_{CF} \ n \ p \ +_C \text{poly } F \ (-_C \text{poly } F$ 
 $\text{pmod}_C \ F \ X_{CF} \ p)))$ 

```

**definition** *divides-gauss-poly* :: ('a,'b) *idx-ring-scheme*  $\Rightarrow$  'a *list*  $\Rightarrow$  *nat*  $\Rightarrow$  *bool*  
**where** *divides-gauss-poly* *F p n* =  
 (*pmod-pow*<sub>C</sub> *F X<sub>C</sub>F n p* +<sub>C</sub>*poly F* (-<sub>C</sub>*poly F pmod*<sub>C</sub> *F X<sub>C</sub>F p*) =  
 [])

**lemma** *mod-gauss-poly*:

**assumes** *field*<sub>C</sub> *R*  
**assumes** *f*  $\in$  *carrier* (*poly-ring* (*ring-of R*))  
**shows** *pmod-pow*<sub>C</sub> *R X<sub>C</sub>R n f* +<sub>C</sub>*poly R* (-<sub>C</sub>*poly R pmod*<sub>C</sub> *R X<sub>C</sub>R f*) =  
*ring.pmod* (*ring-of R*) (*gauss-poly* (*ring-of R*) *n*) *f* (**is** ?*L* = ?*R*)

**proof** –

**interpret** *field* *ring-of R*  
**using** *assms*(1) **unfolding** *field*<sub>C</sub>-*def* **by** *simp*  
**interpret** *d-poly-ring*: *domain* *poly-ring* (*ring-of R*)  
**by** (*rule* *univ-poly-is-domain*[*OF carrier-is-subring*])

**have** *ring-c*: *ring*<sub>C</sub> *R* **using** *assms*(1) **unfolding** *field*<sub>C</sub>-*def* *domain*<sub>C</sub>-*def* *cring*<sub>C</sub>-*def* **by** *auto*  
**have** *d-poly*: *domain*<sub>C</sub> (*poly R*) **using** *assms* (1) **unfolding** *field*<sub>C</sub>-*def*  
**by** (*intro* *poly-domain*) *auto*  
**let** ?*P* = *poly-ring* (*ring-of R*)

**have** ?*L* = *pmod-pow*<sub>C</sub> *R X<sub>ring-of R</sub> n f*  $\oplus$  ?*P* -<sub>C</sub>*poly R pmod*<sub>C</sub> *R X<sub>ring-of R</sub> f*  
**by** (*simp* *add*: *poly-var domain-cD*[*OF d-poly*] *ring-of-poly*[*OF ring-c*])  
**also have** ... = *pmod* (*X<sub>ring-of R</sub>*[ $\lceil$ ?*P* *n*] *f*  $\oplus$  ?*P* -<sub>C</sub>*poly R pmod*<sub>C</sub> *X<sub>ring-of R</sub> f*)  
**using** *assms* *var-carr*[*OF carrier-is-subring*] **by** (*intro* *refl arg-cong2*[**where** *f*=( $\oplus$ ?*P*)]  
*pmod-pow-c arg-cong*[**where** *f*= $\lambda x. (-_C poly R x)$ ] *pmod-c*) *auto*  
**also have** ... = *pmod* (*X<sub>ring-of R</sub>*[ $\lceil$ ?*P* *n*] *f*  $\ominus$  ?*P* *pmod* *X<sub>ring-of R</sub> f*)  
**unfolding** *a-minus-def* **using** *assms*(1,2) *var-carr*[*OF carrier-is-subring*]  
*ring-of-poly*[*OF ring-c*] *long-division-closed*[*OF carrier-is-subfield*]  
**by** (*subst* *domain-cD*[*OF d-poly*]) *auto*  
**also have** ... = *pmod* (*X<sub>ring-of R</sub>*[ $\lceil$ ?*P* *n*] *f*  $\oplus$  ?*P* *pmod* ( $\ominus$  ?*P* *X<sub>ring-of R</sub> f*))  
**using** *assms*(2) *var-carr*[*OF carrier-is-subring*]  
**unfolding** *a-minus-def* **by** (*subst* *long-division-a-inv*[*OF carrier-is-subfield*])  
*auto*  
**also have** ... = *pmod* (*gauss-poly* (*ring-of R*) *n*) *f*  
**using** *assms*(2) *var-carr*[*OF carrier-is-subring*] *var-pow-carr*[*OF carrier-is-subring*]  
**unfolding** *gauss-poly-def a-minus-def* **by** (*subst* *long-division-add*[*OF carrier-is-subfield*]) *auto*

**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *pcoprime-with-gauss-poly*:

**assumes** *field<sub>C</sub> 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** –

**interpret** *field ring-of R*  
**using** *assms(1)* **unfolding** *field<sub>C</sub>-def* **by** *simp*

**have**  $?L \longleftrightarrow \text{pcoprime}_C \ R \ f \ (\text{pmod } (\text{gauss-poly } (\text{ring-of } R) \ n) \ f)$   
**unfolding** *pcoprime-with-gauss-poly-def* **using** *assms* **by** (*subst mod-gauss-poly*) *auto*  
**also have**  $\dots = \text{pcoprime}_{\text{ring-of } R} \ f \ (\text{pmod } (\text{gauss-poly } (\text{ring-of } R) \ n) \ f)$   
**using** *assms gauss-poly-carr long-division-closed[OF carrier-is-subfield]*  
**by** (*intro pcoprime-c*) *auto*  
**also have**  $\dots = \text{pcoprime}_{\text{ring-of } R} (\text{gauss-poly } (\text{ring-of } R) \ n) \ f$   
**by** (*intro pcoprime-step[symmetric] gauss-poly-carr assms*)  
**finally show** *?thesis* **by** *simp*

**qed**

**lemma** *divides-gauss-poly*:

**assumes** *field<sub>C</sub> 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** –

**interpret** *field ring-of R*  
**using** *assms(1)* **unfolding** *field<sub>C</sub>-def* **by** *simp*  
**have**  $?L \longleftrightarrow (\text{pmod } (\text{gauss-poly } (\text{ring-of } R) \ n) \ f = [])$   
**unfolding** *divides-gauss-poly-def* **using** *assms* **by** (*subst mod-gauss-poly*)

*auto*

**also have**  $\dots \longleftrightarrow ?R$   
**using** *assms gauss-poly-carr* **by** (*intro pmod-zero-iff-pdivides[OF carrier-is-subfield]*) *auto*  
**finally show** *?thesis*

**by** *simp*

**qed**

**fun** *rabin-test-powers* ::  $(\ 'a, \ 'b) \ \text{id}x\text{-ring-enum-scheme} \Rightarrow \text{nat} \Rightarrow \text{nat list}$

**where** *rabin-test-powers*  $F \ n =$   
 $\text{map } (\lambda p. \ \text{id}x\text{-size } F \ \widehat{(n \ \text{div } p)}) \ (\text{filter } (\lambda p. \ \text{prime } p \wedge p \ \text{dvd } n))$

[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 Fdegree 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)
  using field-R enum-cD[OF enum-R] unfolding fieldC-def
  by (simp add:finite-field-def finite-field-axioms-def)
```

```
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  $\wedge$  p dvd
n}
  (is ?L = ?R)
```

```
proof -
  let ?f = ( $\lambda$ x. order (ring-of R)(n div x))

  have 0:p  $\in$  {2..n} if Factorial-Ring.prime p p dvd n for p
    using assms that by (simp add: dvd-imp-le prime-ge-2-nat)

  have ?L = ?f ‘ {p  $\in$  {2..n}. Factorial-Ring.prime p  $\wedge$  p dvd n}
    using enum-cD[OF enum-R] by auto
  also have ... = ?f ‘ {p. Factorial-Ring.prime p  $\wedge$  p dvd n}
    using 0 by (intro image-cong Collect-cong) auto
  also have ... = ?R
    by auto
  finally show ?thesis by simp
qed
```

```
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 (cases degree f = 0)
  case True
  thus ?thesis unfolding rabin-test.simps using monic-poly-min-degree
by fastforce
next
  case False
  define N where N = {degree f div p | p . Factorial-Ring.prime p ∧
p dvd degree f}

  have f-carr: f ∈ carrier (poly-ring (ring-of R))
    using assms(1) unfolding monic-poly-def by auto

  have deg-f-gt-0: degree f > 0
    using False by auto
  have rt-powers: set (rabin-test-powers R (degree f)) = (λx. order
(ring-of R) ^ x) ‘ N
    unfolding rabin-test-powers[OF deg-f-gt-0] N-def by auto

  have ?L ↔ divides-gauss-poly R f (idx-size R ^ degree f) ∧
(∀ n ∈ set (rabin-test-powers R (degree f)). (pcoprime-with-gauss-poly
R f n))
    using False by (simp add: list-all-def rabin-test.simps del:rabin-test-powers.simps)
  also have ... ↔ f pdividesring-of R (gauss-poly (ring-of R) (order
(ring-of R) ^ degree f))
    ∧ (∀ n ∈ N. pcoprimering-of R (gauss-poly (ring-of R) (order (ring-of
R) ^ n)) f)
    unfolding divides-gauss-poly[OF field-R f-carr] pcoprime-with-gauss-poly[OF
field-R f-carr]
    rt-powers enum-cD[OF enum-R] by simp
  also have ... ↔ ?R
    using False unfolding N-def by (intro rabin-irreducibility-condition[symmetric]
assms(1)) auto
  finally show ?thesis by simp
qed

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**  $\text{nth-digit } x \ k \ b = 0$   
**using** *assms unfolding nth-digit-def by auto*

**lemma** *nth-digit-bounded'*:  
**assumes**  $b > 0$   
**shows**  $\text{nth-digit } v \ x \ b < b$   
**using** *assms by (simp add: nth-digit-def)*

**lemma** *digit-gen-sum-repr'*:  
**assumes**  $n < b^c$   
**shows**  $n = (\sum_{k < c} \text{nth-digit } n \ k \ b * b^k)$   
**proof** –  
**consider**  $(a) \ b = 0 \ c = 0 \mid (b) \ b = 0 \ c > 0 \mid (c) \ b = 1 \mid (d) \ b > 1$   
**by** *linarith*  
**thus** *?thesis*  
**proof** (*cases*)  
**case** *a* **thus** *?thesis using assms by simp*  
**next**  
**case** *b* **thus** *?thesis using assms by (simp add: zero-power)*  
**next**  
**case** *c* **thus** *?thesis using assms by (simp add: nth-digit-def)*  
**next**  
**case** *d* **thus** *?thesis by (intro digit-gen-sum-repr assms d)*  
**qed**  
**qed**

**lemma**  
**assumes**  $\bigwedge x. x \in A \implies f (g x) = x$   
**shows**  $\bigwedge y. y \in g^{-1} A \implies g (f y) = y$   
**proof** –  
**show**  $g (f y) = y$  **if**  $0: y \in g^{-1} A$  **for**  $y$   
**proof** –  
**obtain**  $x$  **where**  $x\text{-dom}: x \in A$  **and**  $y\text{-def}: y = g x$  **using**  $0$  **by**  
*auto*  
**hence**  $g (f y) = g (f (g x))$  **by** *simp*  
**also** **have**  $\dots = g x$  **by** (*intro arg-cong[where f=g] assms(1)*  
*x-dom*)  
**also** **have**  $\dots = y$  **unfolding**  $y\text{-def}$  **by** *simp*  
**finally** **show** *?thesis by simp*  
**qed**  
**qed**

**lemma** *nth-digit-bij*:  
 $\text{bij-betw } (\lambda v. (\lambda x \in \{..<n\}. \text{nth-digit } v \ x \ b)) \ \{..<b^n\} \ (\{..<n\} \rightarrow_E \{..<b\})$   
**(is** *bij-betw ?f ?A ?B*)  
**proof** –  
**have** *inj-f: inj-on ?f ?A*

**using** *digit-gen-sum-repr'* **by** (*intro inj-on-inverseI*[**where**  $g=(\lambda x. (\sum k < n. x k * b^{\wedge}k))$ ]) *auto*  
**consider** (a)  $b = 0 \ n = 0$  | (b)  $b = 0 \ n > 0$  | (c)  $b > 0$  **by** *linarith*  
**hence** *nth-digit*  $x \ i \ b \in \{..<b\}$  **if**  $i < n$   $x < b^{\wedge}n$  **for**  $i \ x$   
**proof** (*cases*)  
**case** a **then show** *?thesis* **using** *that* **by** *auto*  
**next**  
**case** b **thus** *?thesis* **using** *that* **by** (*simp add:zero-power*)  
**next**  
**case** c **thus** *?thesis* **using** *that* **by** (*simp add:nth-digit-def*)  
**qed**  
**hence**  $?f \ x \in ?B$  **if**  $x \in ?A$  **for**  $x$  **using** *that* **unfolding** *restrict-PiE-iff*  
**by** *auto*  
**hence**  $?f \ ' \ ?A = ?B$   
**using** *card-image*[*OF inj-f*] **by** (*intro card-seteq finite-PiE image-subsetI*) (*auto simp:card-PiE*)  
**thus** *?thesis* **using** *inj-f* **unfolding** *bij-betw-def* **by** *auto*  
**qed**

**lemma** *nth-digit-sum*:

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

**proof** –

**define**  $n$  **where**  $n = (\sum i < l. f \ i * b^{\wedge}i)$

**have** *restrict*  $f \ \{..<l\} \in \{..<l\} \rightarrow_E \{..<b\}$  **using** *assms(1)* **by** *auto*  
**then obtain**  $m$  **where**  $a:(\lambda x \in \{..<l\}. \text{nth-digit} \ m \ x \ b) = \text{restrict} \ f \ \{..<l\}$  **and**  $b:m \in \{..<b^{\wedge}l\}$   
**using** *bij-betw-imp-surj-on*[*OF nth-digit-bij*][**where**  $n=l$  **and**  $b=b$ ]  
**by** (*metis (no-types, lifting) image-iff*)

**have**  $m = (\sum i < l. \text{nth-digit} \ m \ i \ b * b^{\wedge}i)$   
**using**  $b$  **by** (*intro digit-gen-sum-repr'*) *auto*  
**also have**  $\dots = (\sum i < l. f \ i * b^{\wedge}i)$   
**using**  $a$  **by** (*intro sum.cong arg-cong2*)[**where**  $f=(*)$ ] *refl*) (*metis restrict-apply'*)  
**also have**  $\dots = n$  **unfolding** *n-def* **by** *simp*  
**finally have**  $c:n = m$  **by** *simp*  
**show**  $(\sum i < l. f \ i * b^{\wedge}i) < b^{\wedge}l$  **unfolding** *n-def*[*symmetric*]  $c$  **using**  $b$  **by** *auto*  
**show** *nth-digit*  $(\sum i < l. f \ i * b^{\wedge}i) \ k \ b = f \ k$  **if**  $k < l$  **for**  $k$   
**proof** –  
**have** *nth-digit*  $(\sum i < l. f \ i * b^{\wedge}i) \ k \ b = \text{nth-digit} \ m \ k \ b$  **unfolding** *n-def*[*symmetric*]  $c$  **by** *simp*  
**also have**  $\dots = f \ k$  **using**  $a$  **that** **by** (*metis lessThan-iff restrict-apply'*)  
**finally show** *?thesis* **by** *simp*

qed  
qed

lemma *bij-betw-reindex*:

assumes *bij-betw*  $f$   $I$   $J$   
shows *bij-betw*  $(\lambda x. \lambda i \in I. x (f i)) (J \rightarrow_E S) (I \rightarrow_E S)$   
proof (rule *bij-betwI*[**where**  $g = (\lambda x. \lambda i \in J. x (the-inv-into I f i))$ ])  
have  $0: \textit{bij-betw} (the-inv-into I f) J I$   
using *assms bij-betw-the-inv-into* **by** *auto*

show  $(\lambda x. \lambda i \in I. x (f i)) \in (J \rightarrow_E S) \rightarrow I \rightarrow_E S$   
using *bij-betw-apply*[*OF assms*] **by** *auto*  
show  $(\lambda x. \lambda i \in J. x (the-inv-into I f i)) \in (I \rightarrow_E S) \rightarrow J \rightarrow_E S$   
using *bij-betw-apply*[*OF 0*] **by** *auto*  
show  $(\lambda j \in J. (\lambda i \in I. x (f i)) (the-inv-into I f j)) = x$  **if**  $x \in J \rightarrow_E S$   
**for**  $x$   
proof –  
have  $(\lambda i \in I. x (f i)) (the-inv-into I f j) = x j$  **if**  $j \in J$  **for**  $j$   
using  $0$  *assms f-the-inv-into-f-bij-betw bij-betw-apply* **that** **by** *fastforce*  
thus *?thesis* **using** *PiE-arb*[*OF that*] **by** *auto*  
qed  
show  $(\lambda i \in I. (\lambda j \in J. y (the-inv-into I f j)) (f i)) = y$  **if**  $y \in I \rightarrow_E S$   
**for**  $y$   
proof –  
have  $(\lambda j \in J. y (the-inv-into I f j)) (f i) = y i$  **if**  $i \in I$  **for**  $i$   
using *assms 0 that the-inv-into-f-f*[*OF bij-betw-imp-inj-on*[*OF assms*]] *bij-betw-apply* **by** *force*  
thus *?thesis* **using** *PiE-arb*[*OF that*] **by** *auto*  
qed  
qed

lemma *lift-bij-betw*:

assumes *bij-betw*  $f$   $S$   $T$   
shows *bij-betw*  $(\lambda x. \lambda i \in I. f (x i)) (I \rightarrow_E S) (I \rightarrow_E T)$   
proof –  
let  $?g = the-inv-into S f$

have *bij-g*: *bij-betw*  $?g$   $T$   $S$  **using** *bij-betw-the-inv-into*[*OF assms*]  
**by** *simp*  
have  $0: ?g(f x) = x$  **if**  $x \in S$  **for**  $x$  **by** (*intro the-inv-into-f-f that bij-betw-imp-inj-on*[*OF assms*])  
have  $1: f(?g x) = x$  **if**  $x \in T$  **for**  $x$  **by** (*intro f-the-inv-into-f-bij-betw*[*OF assms*] *that*)

have  $(\lambda i \in I. f (x i)) \in I \rightarrow_E T$  **if**  $x \in (I \rightarrow_E S)$  **for**  $x$   
using *bij-betw-apply*[*OF assms*] **that** **by** (*auto simp: Pi-def*)  
**moreover** have  $(\lambda i \in I. ?g (x i)) \in I \rightarrow_E S$  **if**  $x \in (I \rightarrow_E T)$  **for**  $x$   
using *bij-betw-apply*[*OF bij-g*] **that** **by** (*auto simp: Pi-def*)

```

moreover have  $(\lambda i \in I. ?g ((\lambda i \in I. f (x i)) i)) = x$  if  $x \in (I \rightarrow_E S)$ 
for  $x$ 
proof –
  have  $(\lambda i \in I. ?g ((\lambda i \in I. f (x i)) i)) i = x i$  for  $i$ 
    using PiE-mem[OF that] using PiE-arb[OF that] by (cases i ∈ I) (simp add:0)+
  thus ?thesis by auto
qed
moreover have  $(\lambda i \in I. f ((\lambda i \in I. ?g (x i)) i)) = x$  if  $x \in (I \rightarrow_E T)$ 
for  $x$ 
proof –
  have  $(\lambda i \in I. f ((\lambda i \in I. ?g (x i)) i)) i = x i$  for  $i$ 
    using PiE-mem[OF that] using PiE-arb[OF that] by (cases i ∈ I) (simp add:1)+
  thus ?thesis by auto
qed
ultimately show ?thesis
  by (intro bij-betwI[where g=(λx. λi ∈ I. ?g (x i))]) simp-all
qed

```

**lemma** *lists-bij*:

```

  bij-betw  $(\lambda x. \text{map } x [0..<d]) (\{..<d\} \rightarrow_E S) \{x. \text{set } x \subseteq S \wedge \text{length } x = d\}$ 
proof (intro bij-betwI[where g=(λx. λi ∈ {..<d}. x ! i)] funcsetI CollectI, goal-cases)
  case  $(1 x)$ 
  hence  $x \subseteq \{0..<d\} \subseteq S$  by (intro image-subsetI) auto
  thus ?case by simp
next
  case  $(2 x)$  thus ?case by auto
next
  case  $(3 x)$ 
  have restrict  $((!) (\text{map } x [0..<d]) \{..<d\}) j = x j$  for  $j$ 
    using PiE-arb[OF 3] by (cases j ∈ {..<d}) auto
  thus ?case by auto
next
  case  $(4 y)$ 
  have map  $(\text{restrict } ((!) y) \{..<d\}) [0..<d] = \text{map } (((!) y) [0..<d])$ 
by (intro map-cong) auto
  also have  $\dots = y$  using 4 map-nth by blast
  finally show ?case by auto
qed

```

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

```

proof –
  have bij-betw-aux:  $x + s * y < s * t$  if  $x < s y < t$  for  $x y :: \text{nat}$ 
proof –
  have  $x + s * y < s + s * y$  using that by simp

```

```

    also have ... = s * (y+1) by simp
    also have ... ≤ s * t using that by (intro mult-left-mono) auto
    finally show ?thesis by simp
qed

show ?thesis
proof (cases s > 0 ∧ t > 0)
  case True
  then show ?thesis using less-mult-imp-div-less bij-betw-aux
    by (intro bij-betwI[where g=(λx. fst x + s * snd x)]) (auto
simp:mult.commute)
  next
  case False then show ?thesis by (auto simp:bij-betw-def)
qed
qed

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 ≤ b and x > 0 x ≤ 1
  shows x powr b ≤ x powr a
proof -
  have x powr b = (1/x) powr (-b) using assms by (simp add:
powr-divide powr-minus-divide)
  also have ... ≤ (1/x) powr (-a) using assms by (intro powr-mono)
  auto
  also have ... = x powr a using assms by (simp add: powr-divide
powr-minus-divide)
  finally show ?thesis by simp
qed

lemma integral-bind-pmf:
  fixes f :: - ⇒ real
  assumes bounded (f ^ set-pmf (bind-pmf p q))
  shows (∫ x. f x ∂bind-pmf p q) = (∫ x. ∫ y. f y ∂q x ∂p) (is ?L =
?R)
proof -
  obtain M where a:|f x| ≤ M if x ∈ set-pmf (bind-pmf p q) for x
  using assms(1) unfolding bounded-iff by auto
  define clamp where clamp x = (if |x| > M then 0 else x) for x

  obtain x where x ∈ set-pmf (bind-pmf p q) using set-pmf-not-empty

```

by *fast*  
 hence  $M\text{-ge-}0: M \geq 0$  using *a* by *fastforce*

have  $a: \bigwedge x y. x \in \text{set-pmf } p \implies y \in \text{set-pmf } (q \ x) \implies \neg |f \ y| > M$   
 using *a* by *fastforce*  
 hence  $(\int x. f \ x \ \partial \text{bind-pmf } p \ q) = (\int x. \text{clamp } (f \ x) \ \partial \text{bind-pmf } p \ q)$   
 unfolding *clamp-def* by *(intro integral-cong-AE AE-pmfI) auto*  
 also have  $\dots = (\int x. \int y. \text{clamp } (f \ y) \ \partial q \ x \ \partial p)$  unfolding *measure-pmf-bind*  
 by *(subst integral-bind[where K=count-space UNIV and B'=1 and B=M])*  
*(simp-all add:measure-subprob clamp-def M-ge-0)*  
 also have  $\dots = ?R$  unfolding *clamp-def* using *a* by *(intro integral-cong-AE AE-pmfI) simp-all*  
 finally show *?thesis* by *simp*  
 qed

lemma *measure-bind-pmf*:  
 $\text{measure } (\text{bind-pmf } m \ f) \ s = (\int x. \text{measure } (f \ x) \ s \ \partial m)$  (is  $?L = ?R$ )  
 proof –  
 have  $?L = (\int x. \text{indicator } s \ x \ \partial \text{bind-pmf } m \ f)$  by *simp*  
 also have  $\dots = (\int x. (\int y. \text{indicator } s \ y \ \partial f \ x) \ \partial m)$   
 by *(intro integral-bind-pmf) (auto intro!:boundedI)*  
 also have  $\dots = ?R$  by *simp*  
 finally show *?thesis* by *simp*  
 qed

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  $\text{poly-enum } R \ l \ n =$   
 $\text{dropWhile } ((=) \ 0 \ C \ R) \ (\text{map } (\lambda p. \ \text{idx-enum } R \ (\text{nth-digit } n \ (l-1-p) \ (\text{idx-size } R))) \ [0..<l])$

lemma *replicate-drop-while-cancel*:

assumes  $k = \text{length } (\text{takeWhile } ((=) \ x) \ y)$

shows  $\text{replicate } k \ x \ @ \ \text{dropWhile } ((=) \ x) \ y = y$  (is  $?L = ?R$ )

proof –

**have** *replicate*  $k$   $x = \text{takeWhile } ((=) x) y$   
**using** *assms* **by** (*metis* (*full-types*) *replicate-length-same set-takeWhileD*)  
**thus** *?thesis* **by** *simp*  
**qed**

**lemma** *arg-cong3*:  
**assumes**  $x = u$   $y = v$   $z = w$   
**shows**  $f x y z = f u v w$   
**using** *assms* **by** *simp*

**lemma** *list-all-dropwhile*:  $\text{list-all } p \text{ } xs \implies \text{list-all } p \text{ } (\text{dropWhile } q \text{ } xs)$   
**by** (*induction* *xs*) *auto*

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

**proof** –

**let**  $?b = \text{idx-size } R$   
**let**  $?S0 = \{..<l\} \rightarrow_E \{..<\text{order } (\text{ring-of } R)\}$   
**let**  $?S1 = \{..<l\} \rightarrow_E \{x. \text{idx-pred } R x\}$   
**let**  $?S2 = \{xs. \text{list-all } (\text{idx-pred } R) \text{ } xs \wedge \text{length } xs = l\}$   
**let**  $?S3 = \{xs. (xs = [] \vee \text{hd } xs \neq 0_{CR}) \wedge \text{list-all } (\text{idx-pred } R) \text{ } xs \wedge \text{length } xs \leq l\}$   
**let**  $?S4 = \{xs. xs \in \text{carrier } (\text{poly-ring } (\text{ring-of } R)) \wedge \text{length } xs \leq l\}$

**interpret** *ring*  $\text{ring-of } R$  **using** *assms*(2) **unfolding**  $\text{ring}_C\text{-def}$  **by** *simp*

**have**  $0 < \text{order } (\text{ring-of } R)$  **using**  $\text{enum-cD}(1)[OF \text{ } \text{assms}(1)]$  *order-gt-0-iff-finite* **by** *metis*  
**also** **have**  $... = ?b$  **using**  $\text{enum-cD}[OF \text{ } \text{assms}(1)]$  **by** *auto*  
**finally** **have**  $b\text{-gt-0}: ?b > 0$  **by** *simp*

**note**  $\text{bij0} = \text{lift-bij-betw}[OF \text{ } \text{enum-cD}(3)[OF \text{ } \text{assms}(1)], \text{ where } I = \{..<l\}$   
**note**  $\text{bij1} = \text{lists-bij}[\text{where } d=l \text{ and } S = \{x. \text{idx-pred } R x\}]$

**have**  $\text{bij-betw } (\text{dropWhile } ((=) 0_{CR})) ?S2 ?S3$   
**proof** (*rule*  $\text{bij-betwI}[\text{where } g = \lambda xs. \text{replicate } (l - \text{length } xs) 0_{CR}$   
@ *xs*])

**have**  $\text{dropWhile } ((=) 0_{CR}) \text{ } xs \in ?S3$  **if**  $xs \in ?S2$  **for**  $xs$   
**proof** –  
**have**  $\text{dropWhile } ((=) 0_{CR}) \text{ } xs = [] \vee \text{hd } (\text{dropWhile } ((=) 0_{CR}) \text{ } xs) \neq 0_{CR}$   
**using** *hd-dropWhile* **by** (*metis* (*full-types*))  
**moreover** **have**  $\text{length } (\text{dropWhile } ((=) 0_{CR}) \text{ } xs) \leq l$   
**by** (*metis* (*mono-tags*, *lifting*) *mem-Collect-eq length-dropWhile-le*  
*that*)

**ultimately** **show** *?thesis* **using** *that* **by** (*auto simp: list-all-dropwhile*)

**qed**  
**thus**  $\text{dropWhile } ((=) 0_{CR}) \in ?S2 \rightarrow ?S3$  **by** *auto*  
**have**  $\text{replicate } (l - \text{length } xs) 0_{CR} @ xs \in ?S2$  **if**  $xs \in ?S3$  **for**  $xs$   
**proof** –  
**have**  $\text{id}\text{-pred } R 0_{CR}$  **using** *add.one-closed* **by** (*simp add:ring-of-def*)  
**moreover** **have**  $\text{length } (\text{replicate } (l - \text{length } xs) 0_{CR} @ xs) = l$   
**using** *that* **by** *auto*  
**ultimately** **show** *?thesis* **using** *that* **by** (*auto simp:list-all-iff*)  
**qed**  
**thus**  $(\lambda xs. \text{replicate } (l - \text{length } xs) 0_{CR} @ xs) \in ?S3 \rightarrow ?S2$  **by**  
*auto*

**show**  $\text{replicate } (l - \text{length } (\text{dropWhile } ((=) 0_{CR}) x)) 0_{CR} @$   
 $\text{dropWhile } ((=) 0_{CR}) x = x$   
**if**  $x \in ?S2$  **for**  $x$   
**proof** –  
**have**  $\text{length } (\text{takeWhile } ((=) 0_{CR}) x) + \text{length } (\text{dropWhile } ((=)$   
 $0_{CR}) x) = \text{length } x$   
**unfolding** *length-append[symmetric]* **by** *simp*  
**thus** *?thesis* **using** *that* **by** (*intro replicate-drop-while-cancel*)  
*auto*

**qed**  
**show**  $\text{dropWhile } ((=) 0_{CR}) (\text{replicate } (l - \text{length } y) 0_{CR} @ y) =$   
 $y$   
**if**  $y \in ?S3$  **for**  $y$   
**proof** –  
**have**  $\text{dropWhile } ((=) 0_{CR}) (\text{replicate } (l - \text{length } y) 0_{CR} @ y)$   
 $= \text{dropWhile } ((=) 0_{CR}) y$   
**by** (*intro dropWhile-append2*) *simp*  
**also** **have**  $\dots = y$  **using** *that* **by** (*intro iffD2[OF dropWhile-eq-self-iff]*)  
*auto*

**finally** **show** *?thesis* **by** *simp*  
**qed**  
**qed**  
**moreover** **have**  $?S3 = ?S4$   
**unfolding** *ring-of-poly[OF assms(2),symmetric]* **by** (*simp add:ring-of-def*  
*poly-def*)  
**ultimately** **have**  $\text{bij2}: \text{bij-betw } (\text{dropWhile } ((=) 0_{CR})) ?S2 ?S4$  **by**  
*simp*

**have**  $\text{bij3}: \text{bij-betw } (\lambda x. l-1-x) \{..<l\} \{..<l\}$   
**by** (*intro bij-betwI[where g= $\lambda x. l-1-x$ ]*) *auto*  
**note**  $\text{bij4} = \text{bij-betw-reindex}[OF \text{bij3}, \text{where } S = \{..<\text{order } (\text{ring-of}$   
 $R)\}]$   
**have**  $\text{bij5}: \text{bij-betw } (\lambda n. (\lambda p \in \{..<l\}. \text{nth-digit } n \ p \ ?b)) \{..<?b\} ?S0$   
**using** *nth-digit-bij[where n=l]* *enum-cD[OF assms(1)]* **by** *simp*  
**have**  $\text{bij6}: \text{bij-betw } (\lambda n. (\lambda p \in \{..<l\}. \text{nth-digit } n \ (l-1-p) \ ?b)) \{..<?b\}$   
 $?S0$   
**by** (*intro iffD2[OF arg-cong3[where f=bij-betw]*) *bij-betw-trans[OF*

*bij5 bij4*]]] *force+*

**have** *carrier* (*ring-of R*) = {*x*. *idx-pred R x*} **unfolding** *ring-of-def*  
**by** *auto*  
**hence** *bij7*: *bij-betw* ( $\lambda n. (\lambda p \in \{..<l\}. \text{idx-enum } R \text{ (nth-digit } n \text{ (} l-1-p \text{) } ?b)) \{..<?b \wedge l\} ?S1$ )  
**by** (*intro iffD2*[*OF arg-cong3*[**where** *f=bij-betw*] *bij-betw-trans*[*OF bij6 bij0*]]) *fastforce+*

**have** *bij8*: *bij-betw* ( $\lambda n. \text{map } (\lambda p. \text{idx-enum } R \text{ (nth-digit } n \text{ (} l-1-p \text{) } ?b)) [0..<l] \{..<?b \wedge l\} ?S2$ )  
**by** (*intro iffD2*[*OF arg-cong3*[**where** *f=bij-betw*] *bij-betw-trans*[*OF bij7 bij1*]])  
*(auto simp: comp-def list-all-iff atLeast0LessThan[symmetric])*

**thus** *bij-betw* (*poly-enum R l*) { $..<\text{idx-size } R \wedge l$ } ?*S4*  
**using** *bij-betw-trans*[*OF bij8 bij2*] **unfolding** *poly-enum-def comp-def*  
**by** *simp*  
**qed**

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

**where** *poly-enum-inv R l f* =  
*(let f' = replicate (l - length f) 0<sub>C</sub>R @ f in*  
*( $\sum i < l. \text{idx-enum-inv } R \text{ (f' ! (l - 1 - i)) * idx-size } R \wedge i$ ))*

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

**lemma** *poly-enum-inv*:

**assumes** *enum<sub>C</sub> R ring<sub>C</sub> R*  
**assumes** *x*  $\in$  {*xs*. *xs*  $\in$  *carrier* (*poly-ring* (*ring-of R*))  $\wedge$  *length xs*  $\leq$  *l*}  
**shows** *the-inv-into* { $..<\text{idx-size } R \wedge l$ } (*poly-enum R l*) *x* = *poly-enum-inv R l x*

**proof** –

**define** *f* **where** *f* = *replicate* (*l* - *length x*) 0<sub>C</sub>R @ *x*  
**let** ?*b* = *idx-size R*  
**let** ?*d* = *dropWhile* ((=) 0<sub>C</sub>R)

**have** *len-f*: *length f* = *l* **using** *assms(3)* **unfolding** *f-def* **by** *auto*  
**note** *enum-c* = *enum-cD*[*OF assms(1)*]

**interpret** *ring* *ring-of R* **using** *assms(2)* **unfolding** *ring<sub>C</sub>-def* **by**  
*simp*

**have** 0: *idx-enum-inv R y* < ?*b* **if** *y*  $\in$  *carrier* (*ring-of R*) **for** *y*  
**using** *bij-betw-imp-surj-on*[*OF enum-c(4)*] *enum-c(2)* **that** **by** *auto*  
**have** 1: (*x* = []  $\vee$  *lead-coeff x*  $\neq$  0<sub>C</sub>R)  $\wedge$  *list-all* (*idx-pred R*) *x*  $\wedge$

$\text{length } x \leq l$   
**using**  $\text{assms}(3)$  **unfolding**  $\text{ring-of-poly}[OF \text{ assms}(2), \text{symmetric}]$   
**by**  $(\text{simp add:ring-of-def poly-def})$   
**moreover have**  $0_{\text{ring-of } R} \in \text{carrier } (\text{ring-of } R)$  **by**  $\text{simp}$   
**hence**  $\text{idx-pred } R \ 0_C R$  **unfolding**  $\text{ring-of-def}$  **by**  $\text{simp}$   
**ultimately have**  $2$ :  $\text{set } f \subseteq \text{carrier } (\text{ring-of } R)$   
**unfolding**  $f\text{-def}$  **by**  $(\text{auto simp add:ring-of-def list-all-iff})$

**have**  $\text{poly-enum } R \ l (\text{poly-enum-inv } R \ l \ x) = \text{poly-enum } R \ l \ (\sum_{i < l} \text{idx-enum-inv } R \ (f \ ! \ (l-1-i)) * ?b \ ^i)$   
**unfolding**  $\text{poly-enum-inv-def } f\text{-def}[\text{symmetric}]$  **by**  $\text{simp}$   
**also have**  $\dots = ?d \ (\text{map } (\lambda p. \ \text{idx-enum } R \ (\text{idx-enum-inv } R \ (f \ ! \ (l-1 - (l-1-p)))))) \ [0..<l])$   
**unfolding**  $\text{poly-enum-def}$  **using**  $2 \ \text{len-f}$  **by**  $(\text{intro arg-cong}[\text{where } f=?d])$   
 $\text{arg-cong}[\text{where } f=\text{idx-enum } R] \ \text{map-cong refl nth-digit-sum } 0)$   
 $\text{auto}$   
**also have**  $\dots = ?d \ (\text{map } (\lambda p. \ (f \ ! \ (l-1 - (l-1-p)))) \ [0..<l])$   
**using**  $2 \ \text{len-f}$  **by**  $(\text{intro arg-cong}[\text{where } f=?d] \ \text{map-cong refl enum-c}) \ \text{auto}$   
**also have**  $\dots = ?d \ (\text{map } (\lambda p. \ (f \ ! \ p)) \ [0..<l])$   
**by**  $(\text{intro arg-cong}[\text{where } f=?d] \ \text{map-cong}) \ \text{auto}$   
**also have**  $\dots = ?d \ f$  **using**  $\text{len-f map-nth}$  **by**  $(\text{intro arg-cong}[\text{where } f=?d]) \ \text{auto}$   
**also have**  $\dots = ?d \ x$  **unfolding**  $f\text{-def}$  **by**  $(\text{intro dropWhile-append2}) \ \text{auto}$   
**also have**  $\dots = x$  **using**  $1$  **by**  $(\text{intro iffD2}[OF \ \text{dropWhile-eq-self-iff}]) \ \text{auto}$   
 $\text{auto}$   
**finally have**  $\text{poly-enum } R \ l \ (\text{poly-enum-inv } R \ l \ x) = x$  **by**  $\text{simp}$   
**moreover have**  $\text{poly-enum-inv } R \ l \ x < \text{idx-size } R \ ^l$   
**unfolding**  $\text{poly-enum-inv-def Let-def } f\text{-def}[\text{symmetric}]$  **using**  $\text{len-f}$   
 $2$   
**by**  $(\text{intro nth-digit-sum}(2) \ 0) \ \text{auto}$   
**ultimately show**  $?thesis$   
**by**  $(\text{intro the-inv-into-f-eq bij-betw-imp-inj-on}[OF \ \text{bij-betw-poly-enum}[OF \ \text{assms}(1,2)]]]) \ \text{auto}$   
**qed**

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

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

$idx\text{-size} = idx\text{-size } R \hat{\ } degree\ f,$   
 $idx\text{-enum} = poly\text{-enum } R\ (degree\ f),$   
 $idx\text{-enum}\text{-inv} = poly\text{-enum}\text{-inv } R\ (degree\ f)\ \Downarrow$

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

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

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

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

**interpretation**  $field\ ring\text{-of } R$   
**using**  $field\text{-}R$  **unfolding**  $field_C\text{-def}$  **by**  $auto$

**interpretation**  $d: domain\ P$   
**by**  $(intro\ univ\text{-poly}\text{-is}\text{-domain}\ carrier\text{-is}\text{-subring})$

**interpretation**  $i: ideal\ I\ P$   
**using**  $f\text{-carr}$  **by**  $(intro\ d.\text{cgenideal}\text{-ideal})\ auto$

**interpretation**  $s: ring\text{-hom}\text{-ring } P\ P\ Quot\ I\ (+>_P)\ I$   
**using**  $i.\text{rcos}\text{-ring}\text{-hom}\text{-ring}$  **by**  $auto$

**interpretation**  $cr: cring\ P\ Quot\ I$   
**by**  $(intro\ i.\text{quotient}\text{-is}\text{-cring } d.\text{cring}\text{-axioms})$

**lemma**  $ring\text{-}c: ring_C\ R$   
**using**  $field\text{-}R$  **unfolding**  $field_C\text{-def}$   $domain_C\text{-def}$   $cring_C\text{-def}$  **by**  $auto$

**lemma**  $d\text{-poly}: domain_C\ (poly\ R)$  **using**  $field\text{-}R$  **unfolding**  $field_C\text{-def}$   
**by**  $(intro\ poly\text{-domain})\ auto$

**lemma**  $ideal\text{-mod}:$   
**assumes**  $y \in carrier\ P$

**shows**  $I +>_P (pmod\ y\ f) = I +>_P\ y$   
**proof** –  
**have**  $f \in I$  **by** (*intro d.cgenideal-self f-carr*)  
**hence**  $(f \otimes_P (pdiv\ y\ f)) \in I$   
**using** *long-division-closed[OF carrier-is-subfield] assms f-carr*  
**by** (*intro i.I-r-closed*) (*simp-all*)  
**hence**  $y \in I +>_P (pmod\ y\ f)$   
**using** *assms f-carr unfolding a-r-coset-def'*  
**by** (*subst pdiv-pmod[OF carrier-is-subfield, where q=f]*) *auto*  
**thus** *?thesis*  
**by** (*intro i.a-repr-independence' assms long-division-closed[OF carrier-is-subfield] f-carr*)  
**qed**

**lemma** *poly-mod-ring-carr-1*:  
 $carrier\ (ring-of\ (poly-mod-ring\ R\ f)) = \{xs.\ xs \in carrier\ P \wedge degree\ xs < degree\ f\}$   
*(is ?L = ?R)*  
**proof** –  
**have**  $?L = \{xs.\ xs \in carrier\ (ring-of\ (poly\ R)) \wedge degree\ xs < degree\ f\}$   
**using** *deg-f unfolding poly-mod-ring-def ring-of-def* **by** *auto*  
**also have**  $\dots = ?R$  **unfolding** *ring-of-poly[OF ring-c]* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

**lemma** *poly-mod-ring-carr*:  
**assumes**  $y \in carrier\ P$   
**shows**  $pmod\ y\ f \in carrier\ (ring-of\ (poly-mod-ring\ R\ f))$   
**proof** –  
**have**  $f \neq []$  **using** *deg-f* **by** *auto*  
**hence**  $pmod\ y\ f = [] \vee degree\ (pmod\ y\ f) < degree\ f$   
**by** (*intro pmod-degree[OF carrier-is-subfield] assms f-carr*)  
**hence**  $degree\ (pmod\ y\ f) < degree\ f$  **using** *deg-f* **by** *auto*  
**moreover have**  $pmod\ y\ f \in carrier\ P$   
**using** *f-carr assms long-division-closed[OF carrier-is-subfield]* **by** *auto*  
**ultimately show** *?thesis* **unfolding** *poly-mod-ring-carr-1* **by** *auto*  
**qed**

**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** –  
**have**  $poly-mod-ring-iso\ R\ f\ x \in carrier\ (P\ Quot\ I)$   
**if**  $x \in carrier\ (ring-of\ (poly-mod-ring\ R\ f))$  **for**  $x$   
**proof** –  
**have**  $I \subseteq carrier\ P$  **by** *auto*  
**moreover have**  $x \in carrier\ P$  **using** *that* **unfolding** *poly-mod-ring-carr-1*

by *auto*  
 ultimately have *poly-mod-ring-iso*  $R f x \in a\text{-rcosets}_P I$   
 using that *f-carr unfolding poly-mod-ring-iso-def* by (intro  
*d.a-rcosetsI*) *auto*  
 thus *?thesis unfolding FactRing-def* by *simp*  
 qed  
 moreover have  $x \in \text{carrier (ring-of (poly-mod-ring } R f))$   
 if  $x \in \text{carrier (} P \text{ Quot } I)$  for  $x$   
 proof –  
 have  $x \in a\text{-rcosets}_P I$  using that *unfolding FactRing-def* by *auto*  
 then obtain  $y$  where *y-def*:  $x = I +>_P y$   $y \in \text{carrier } P$   
 using that *unfolding A-RCOSETS-def'* by *auto*  
 define  $z$  where  $z = \text{pmod } y f$   
 have  $I +>_P z = I +>_P y$  *unfolding z-def* by (intro *ideal-mod*  
*y-def*)  
 hence *poly-mod-ring-iso*  $R f z = x$  *unfolding poly-mod-ring-iso-def*  
*y-def* by *simp*  
 moreover have  $z \in \text{carrier (ring-of (poly-mod-ring } R f))$   
*unfolding z-def* by (intro *poly-mod-ring-carr y-def*)  
 ultimately show *?thesis* by *auto*  
 qed  
 ultimately show *?thesis* by *auto*  
 qed

lemma *poly-mod-ring-iso-inj*:

*inj-on (poly-mod-ring-iso R f) (carrier (ring-of (poly-mod-ring R f)))*  
 proof (rule *inj-onI*)  
 fix  $x y$   
 assume  $x \in \text{carrier (ring-of (poly-mod-ring } R f))$   
 hence  $x : x \in \text{carrier } P$  *degree*  $x < \text{degree } f$  *unfolding poly-mod-ring-carr-1*  
 by *auto*  
 assume  $y \in \text{carrier (ring-of (poly-mod-ring } R f))$   
 hence  $y : y \in \text{carrier } P$  *degree*  $y < \text{degree } f$  *unfolding poly-mod-ring-carr-1*  
 by *auto*

have *degree*  $(x \ominus_P y) \leq \max (\text{degree } x) (\text{degree } (\ominus_P y))$   
*unfolding a-minus-def* by (intro *degree-add*)  
 also have  $\dots = \max (\text{degree } x) (\text{degree } y)$   
*unfolding univ-poly-a-inv-degree[OF carrier-is-subring y(1)]* by  
*simp*  
 also have  $\dots < \text{degree } f$  using *x(2) y(2)* by *simp*  
 finally have *d.degree*  $(x \ominus_P y) < \text{degree } f$  by *simp*

assume *poly-mod-ring-iso*  $R f x = \text{poly-mod-ring-iso } R f y$   
 hence  $I +>_P x = I +>_P y$  *unfolding poly-mod-ring-iso-def* by  
*simp*  
 hence  $x \ominus_P y \in I$  using  $x y$  by (*subst d.quotient-eq-iff-same-a-r-cos[OF*

*i.ideal-axioms*]) *auto*  
**hence**  $f \text{ pdivides}_{\text{ring-of } R} (x \ominus_P y)$   
**using**  $f\text{-carr } x(1) \ y \ d.m\text{-comm}$  **unfolding** *cgenideal-def pdivides-def*  
*factor-def* **by** *auto*  
**hence**  $(x \ominus_P y) = [] \vee \text{degree } (x \ominus_P y) \geq \text{degree } f$   
**using**  $x(1) \ y(1) \ f\text{-carr pdivides-imp-degree-le}$ [*OF carrier-is-subring*]  
**by** (*meson d.minus-closed*)  
**hence**  $(x \ominus_P y) = \mathbf{0}_P$  **unfolding** *univ-poly-zero* **using**  $d$  **by** *simp*  
**thus**  $x = y$  **using**  $x(1) \ y(1)$  **by** *simp*  
**qed**

**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))*  
**using** *poly-mod-ring-iso-ran poly-mod-ring-iso-inj* **unfolding** *bij-betw-def*  
**by** *simp*

**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)))*  
**unfolding** *poly-mod-ring-iso-inv-def* **using** *poly-mod-iso-ring-bij* *bij-betw-the-inv-into*  
**by** *blast*

**lemma** *poly-mod-ring-iso-inv-1*:  
**assumes**  $x \in \text{carrier } (P \text{ Quot } I)$   
**shows**  $\text{poly-mod-ring-iso } R \ f \ (\text{poly-mod-ring-iso-inv } R \ f \ x) = x$   
**unfolding** *poly-mod-ring-iso-inv-def* **using** *assms poly-mod-iso-ring-bij*  
**by** (*intro f-the-inv-into-f-bij-betw*) *auto*

**lemma** *poly-mod-ring-iso-inv-2*:  
**assumes**  $x \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R \ f))$   
**shows**  $\text{poly-mod-ring-iso-inv } R \ f \ (\text{poly-mod-ring-iso } R \ f \ x) = x$   
**unfolding** *poly-mod-ring-iso-inv-def* **using** *assms*  
**by** (*intro the-inv-into-f-f poly-mod-ring-iso-inj*)

**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** –  
**have**  $?L = \text{pmod}_C \ R \ (x \oplus_{\text{ring-of } (\text{poly } R) \ y} f)$   
**unfolding** *poly-mod-ring-def ring-of-def* **using** *domain-cD[OF d-poly]* **by** *simp*  
**also have**  $\dots = ?R$   
**using** *assms* **unfolding** *ring-of-poly[OF ring-c]* **by** (*intro pmod-c[OF field-R] f-carr*) *auto*  
**finally show** *?thesis*  
**by** *simp*

qed

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

proof–

have  $\mathbf{0}_{\text{ring-of } (poly\text{-mod-ring } R f)} = \mathbf{0}_{\text{ring-of } (poly R)}$

using *domain-cD[OF d-poly]* **unfolding** *ring-of-def poly-mod-ring-def*

by *simp*

also have  $\dots = \mathbf{0}_P$  **unfolding** *ring-of-poly[OF ring-c]* by *simp*

finally show *?thesis* by *simp*

qed

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

proof–

have  $\mathbf{1}_{\text{ring-of } (poly\text{-mod-ring } R f)} = \mathbf{1}_{\text{ring-of } (poly R)}$

using *domain-cD[OF d-poly]* **unfolding** *ring-of-def poly-mod-ring-def*

by *simp*

also have  $\dots = \mathbf{1}_P$  **unfolding** *ring-of-poly[OF ring-c]* by *simp*

finally show  $\mathbf{1}_{\text{ring-of } (poly\text{-mod-ring } R f)} = \mathbf{1}_P$  by *simp*

qed

lemma *poly-mod-ring-mult*:

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

assumes  $y \in \text{carrier } P$

shows  $x \otimes_{\text{ring-of } (poly\text{-mod-ring } R f)} y = \text{pmod } (x \otimes_P y) f$  (is ?L = ?R)

proof –

have  $?L = \text{pmod}_C R (x \otimes_{\text{ring-of } (poly R)} y) f$

**unfolding** *poly-mod-ring-def ring-of-def* using *domain-cD[OF d-poly]* by *simp*

also have  $\dots = ?R$

using *assms* **unfolding** *poly-mod-ring-carr-1 ring-of-poly[OF ring-c]*

by (*intro pmod-c[OF field-R] f-carr*) *auto*

finally show *?thesis*

by *simp*

qed

lemma *poly-mod-ring-iso-inv*:

*poly-mod-ring-iso-inv*  $R f \in \text{ring-iso } (P \text{ Quot } I) (\text{ring-of } (poly\text{-mod-ring } R f))$

(is ?f  $\in \text{ring-iso } ?S ?T$ )

proof (*rule ring-iso-memI*)

fix  $x$  assume  $x \in \text{carrier } ?S$

thus ?f  $x \in \text{carrier } ?T$  using *bij-betw-apply[OF poly-mod-iso-ring-bij-2]*

by *auto*

next

fix  $x y$  assume  $x : x \in \text{carrier } ?S$  and  $y : y \in \text{carrier } ?S$

have ?f  $x \in \text{carrier } (\text{ring-of } (poly\text{-mod-ring } R f))$

by (*rule bij-betw-apply[OF poly-mod-iso-ring-bij-2 x]*)

**hence**  $x': ?f x \in \text{carrier } P$  **unfolding** *poly-mod-ring-carr-1* **by** *simp*  
**have**  $?f y \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$   
**by** (*rule bij-betw-apply[OF poly-mod-iso-ring-bij-2 y]*)  
**hence**  $y': ?f y \in \text{carrier } P$  **unfolding** *poly-mod-ring-carr-1* **by** *simp*

**have**  $0: ?f x \otimes_{?T} ?f y = \text{pmod } (?f x \otimes_P ?f y) f$   
**by** (*intro poly-mod-ring-mult x' y'*)  
**also have**  $\dots \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$   
**using**  $x' y'$  **by** (*intro poly-mod-ring-carr*) *auto*  
**finally have**  $xy: ?f x \otimes_{?T} ?f y \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$  **by** *simp*

**have**  $?f (x \otimes_{?S} y) = ?f (\text{poly-mod-ring-iso } R f (?f x) \otimes_{?S} \text{poly-mod-ring-iso } R f (?f y))$   
**using**  $x y$  **by** (*simp add:poly-mod-ring-iso-inv-1*)  
**also have**  $\dots = ?f ((I +>_P (?f x)) \otimes_{?S} (I +>_P (?f y)))$   
**unfolding** *poly-mod-ring-iso-def* **by** *simp*  
**also have**  $\dots = ?f (I +>_P (?f x \otimes_P ?f y))$   
**using**  $x' y'$  **by** *simp*  
**also have**  $\dots = ?f (I +>_P (\text{pmod } (?f x \otimes_P ?f y) f))$   
**using**  $x' y'$  **by** (*subst ideal-mod*) *auto*  
**also have**  $\dots = ?f (I +>_P (?f x \otimes_{?T} ?f y))$   
**unfolding**  $0$  **by** *simp*  
**also have**  $\dots = ?f (\text{poly-mod-ring-iso } R f (?f x \otimes_{?T} ?f y))$   
**unfolding** *poly-mod-ring-iso-def* **by** *simp*  
**also have**  $\dots = ?f x \otimes_{?T} ?f y$   
**using**  $xy$  **by** (*intro poly-mod-ring-iso-inv-2*)  
**finally show**  $?f (x \otimes_{?S} y) = ?f x \otimes_{?T} ?f y$  **by** *simp*

**next**  
**fix**  $x y$  **assume**  $x: x \in \text{carrier } ?S$  **and**  $y: y \in \text{carrier } ?S$   
**have**  $?f x \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$   
**by** (*rule bij-betw-apply[OF poly-mod-iso-ring-bij-2 x]*)  
**hence**  $x': ?f x \in \text{carrier } P$  **unfolding** *poly-mod-ring-carr-1* **by** *simp*  
**have**  $?f y \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$   
**by** (*rule bij-betw-apply[OF poly-mod-iso-ring-bij-2 y]*)  
**hence**  $y': ?f y \in \text{carrier } P$  **unfolding** *poly-mod-ring-carr-1* **by** *simp*

**have**  $0: ?f x \oplus_{?T} ?f y = \text{pmod } (?f x \oplus_P ?f y) f$  **by** (*intro poly-mod-ring-add x' y'*)  
**also have**  $\dots \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$   
**using**  $x' y'$  **by** (*intro poly-mod-ring-carr*) *auto*  
**finally have**  $xy: ?f x \oplus_{?T} ?f y \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$  **by** *simp*

**have**  $?f (x \oplus_{?S} y) = ?f (\text{poly-mod-ring-iso } R f (?f x) \oplus_{?S} \text{poly-mod-ring-iso } R f (?f y))$   
**using**  $x y$  **by** (*simp add:poly-mod-ring-iso-inv-1*)  
**also have**  $\dots = ?f ((I +>_P (?f x)) \oplus_{?S} (I +>_P (?f y)))$   
**unfolding** *poly-mod-ring-iso-def* **by** *simp*

**also have**  $\dots = ?f (I +>_P (?f x \oplus_P ?f y))$   
**using**  $x' y'$  **by** *simp*  
**also have**  $\dots = ?f (I +>_P (pmod (?f x \oplus_P ?f y) f))$   
**using**  $x' y'$  **by** (*subst ideal-mod*) *auto*  
**also have**  $\dots = ?f (I +>_P (?f x \oplus_{?T} ?f y))$   
**unfolding**  $0$  **by** *simp*  
**also have**  $\dots = ?f (poly-mod-ring-iso R f (?f x \oplus_{?T} ?f y))$   
**unfolding** *poly-mod-ring-iso-def* **by** *simp*  
**also have**  $\dots = ?f x \oplus_{?T} ?f y$   
**using**  $xy$  **by** (*intro poly-mod-ring-iso-inv-2*)  
**finally show**  $?f (x \oplus_{?S} y) = ?f x \oplus_{?T} ?f y$  **by** *simp*  
**next**  
**have** *poly-mod-ring-iso R f*  $\mathbf{1}_{ring-of (poly-mod-ring R f)} = (I +>_P \mathbf{1}_P)$   
**unfolding** *poly-mod-ring-one poly-mod-ring-iso-def* **by** *simp*  
**also have**  $\dots = \mathbf{1}_P Quot I$  **using** *s.hom-one* **by** *simp*  
**finally have** *poly-mod-ring-iso R f*  $\mathbf{1}_{ring-of (poly-mod-ring R f)} = \mathbf{1}_P Quot I$  **by** *simp*  
**moreover have** *degree*  $\mathbf{1}_P < degree f$   
**using** *deg-f* **unfolding** *univ-poly-one* **by** *simp*  
**hence**  $\mathbf{1}_{ring-of (poly-mod-ring R f)} \in carrier (ring-of (poly-mod-ring R f))$   
**unfolding** *poly-mod-ring-one poly-mod-ring-carr-1* **by** *simp*  
**ultimately show**  $?f (\mathbf{1}_{?S}) = \mathbf{1}_{?T}$   
**unfolding** *poly-mod-ring-iso-inv-def* **by** (*intro the-inv-into-f-eq poly-mod-ring-iso-inj*)  
**next**  
**show** *bij-betw ?f (carrier ?S) (carrier ?T)* **by** (*rule poly-mod-iso-ring-bij-2*)  
**qed**

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

**proof** –  
**let**  $?f = poly-mod-ring-iso-inv R f$

**have** *poly-mod-ring-iso R f*  $\mathbf{0}_P = \mathbf{0}_P Quot PIdl_P f$   
**unfolding** *poly-mod-ring-iso-def* **by** *simp*  
**moreover have**  $\square \in carrier P$  **using** *univ-poly-zero* [**where**  $K = carrier (ring-of R)$ ] **by** *auto*  
**ultimately have**  $?f \mathbf{0}_P Quot I = \mathbf{0}_P$   
**unfolding** *univ-poly-zero poly-mod-ring-iso-inv-def* **using** *deg-f*  
**by** (*intro the-inv-into-f-eq bij-betw-imp-inj-on[OF poly-mod-iso-ring-bij]*)  
*(simp-all add: add: poly-mod-ring-carr-1)*  
**also have**  $\dots = 0_C poly R$  **using** *ring-of-poly* [*OF ring-c*] *domain-cD* [*OF d-poly*] **by** *auto*

**finally have**  $?f \mathbf{0}_P \text{Quot } I = 0_{C \text{poly } R}$  **by** *simp*  
**thus**  $\text{ring-of } (\text{poly-mod-ring } R \ f) (\text{zero} := ?f \mathbf{0}_P \text{Quot } I) = \text{ring-of}$   
 $(\text{poly-mod-ring } R \ f)$   
**unfolding**  $\text{ring-of-def poly-mod-ring-def}$  **by** *auto*  
**thus**  $\text{cring } (\text{ring-of } (\text{poly-mod-ring } R \ f))$   
**using**  $\text{cr.ring-iso-imp-imp-cring}[OF \ \text{poly-mod-ring-iso-inv}]$  **by** *simp*  
**qed**

**interpretation**  $\text{cr-p: cring } (\text{ring-of } (\text{poly-mod-ring } R \ f))$   
**by**  $(\text{rule cring-poly-mod-ring-1})$

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

**proof** –

**let**  $?P = \text{ring-of } (\text{poly-mod-ring } R \ f)$   
**have**  ${}^{-C} \text{poly-mod-ring } R \ f \ x = \ominus_{\text{ring-of } (\text{poly-mod-ring } R \ f)} \ x$  (**is**  $?L$   
 $= ?R$ )

**if**  $x \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R \ f))$  **for**  $x$

**proof**  $(\text{rule cr-p.minus-equality}[\text{symmetric}, OF \ \text{- that}])$

**have**  ${}^{-C} \text{poly-mod-ring } R \ f \ x = {}^{-C} \text{poly } R \ x$  **unfolding**  $\text{poly-mod-ring-def}$   
**by** *simp*

**also have**  $\dots = \ominus_P \ x$  **using**  $\text{that}$  **unfolding**  $\text{poly-mod-ring-carr-1}$

**by**  $(\text{subst domain-cD}[OF \ d\text{-poly}]) (\text{simp-all add:ring-of-poly}[OF$   
 $\text{ring-c}])$

**finally have**  $0: {}^{-C} \text{poly-mod-ring } R \ f \ x = \ominus_P \ x$  **by** *simp*

**have**  $1: \ominus_P \ x \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R \ f))$

**using**  $\text{that univ-poly-a-inv-degree}[OF \ \text{carrier-is-subring}]$  **unfold-**

**ing**  $\text{poly-mod-ring-carr-1}$

**by** *auto*

**have**  ${}^{-C} \text{poly-mod-ring } R \ f \ x \oplus_{?P} \ x = \text{pmod } (\ominus_P \ x \oplus_P \ x) \ f$

**using**  $\text{that } 1$  **unfolding**  $0 \ \text{poly-mod-ring-carr-1}$  **by**  $(\text{intro poly-mod-ring-add})$

*auto*

**also have**  $\dots = \text{pmod } \mathbf{0}_P \ f$

**using**  $\text{that}$  **unfolding**  $\text{poly-mod-ring-carr-1}$  **by** *simp algebra*

**also have**  $\dots = \square$

**unfolding**  $\text{univ-poly-zero}$  **using**  $\text{carrier-is-subfield f-carr long-division-zero}(2)$

**by** *presburger*

**also have**  $\dots = \mathbf{0}_{?P}$  **by**  $(\text{simp add:poly-mod-ring-def ring-of-def}$   
 $\text{poly-def})$

**finally show**  ${}^{-C} \text{poly-mod-ring } R \ f \ x \oplus_{?P} \ x = \mathbf{0}_{?P}$  **by** *simp*

**show**  ${}^{-C} \text{poly-mod-ring } R \ f \ x \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R$   
 $f))$

**unfolding**  $0$  **by**  $(\text{rule } 1)$

**qed**

**moreover have**  $x \text{ } {}^{-1} C \text{poly-mod-ring } R \ f = \text{inv}_{\text{ring-of } (\text{poly-mod-ring } R \ f)}$

$x$

**if**  $x\text{-unit: } x \in \text{Units } (\text{ring-of } (\text{poly-mod-ring } R \ f))$  **for**  $x$

**proof** (rule *cr-p.comm-inv-char[symmetric]*)  
**show**  $x\text{-carr}: x \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$   
**using that unfolding** *Units-def* **by auto**

**obtain**  $y$  **where**  $y: x \otimes_{\text{ring-of } (\text{poly-mod-ring } R f)} y = \mathbf{1}_{\text{ring-of } (\text{poly-mod-ring } R f)}$   
**and**  $y\text{-carr}: y \in \text{carrier } (\text{ring-of } (\text{poly-mod-ring } R f))$   
**using**  $x\text{-unit}$  **unfolding** *Units-def* **by auto**

**have**  $\text{pmod } (x \otimes_P y) f = x \otimes_{\text{ring-of } (\text{poly-mod-ring } R f)} y$   
**using**  $x\text{-carr } y\text{-carr}$  **by** (*intro poly-mod-ring-mult[symmetric]*)  
(*auto simp: poly-mod-ring-carr-1*)  
**also have**  $\dots = \mathbf{1}_P$   
**unfolding**  $y$  *poly-mod-ring-one* **by** *simp*  
**finally have**  $1: \text{pmod } (x \otimes_P y) f = \mathbf{1}_P$  **by** *simp*

**have**  $\text{pcoprime}_{\text{ring-of } R} (x \otimes_P y) f = \text{pcoprime}_{\text{ring-of } R} f$  (*pmod*  
 $(x \otimes_P y) f$ )  
**using**  $x\text{-carr } y\text{-carr } f\text{-carr}$  **unfolding** *poly-mod-ring-carr-1* **by**  
(*intro pcoprime-step*) *auto*  
**also have**  $\dots = \text{pcoprime}_{\text{ring-of } R} f$   $\mathbf{1}_P$  **unfolding**  $1$  **by** *simp*  
**also have**  $\dots = \text{True}$  **using** *pcoprime-one* **by** *simp*  
**finally have**  $\text{pcoprime}_{\text{ring-of } R} (x \otimes_P y) f$  **by** *simp*  
**hence**  $\text{pcoprime}_{\text{ring-of } R} x f$   
**using**  $x\text{-carr } y\text{-carr } f\text{-carr}$  *pcoprime-left-factor* **unfolding** *poly-mod-ring-carr-1*  
**by** *blast*  
**hence**  $2: \text{length } (\text{snd } (\text{ext-euclidean } R x f)) = 1$   
**using**  $f\text{-carr } x\text{-carr}$  *pcoprime-c[OF field-R]* **unfolding** *poly-mod-ring-carr-1*  
*pcoprime\_C.simps*  
**by auto**

**obtain**  $u v r$  **where**  $\text{wvr-def}: ((u,v),r) = \text{ext-euclidean } R x f$  **by**  
(*metis surj-pair*)

**have**  $x\text{-carr}' : x \in \text{carrier } P$  **using**  $x\text{-carr}$  **unfolding** *poly-mod-ring-carr-1*  
**by auto**  
**have**  $r\text{-eq}: r = x \otimes_P u \oplus_P f \otimes_P v$  **and**  $r\text{v-carr}: \{r, u, v\} \subseteq \text{carrier } P$   
**using**  $\text{wvr-def}[symmetric]$  *ext-euclidean[OF field-R x-carr' f-carr]*  
**by auto**

**have**  $\text{length } r = 1$  **using**  $2$   $\text{wvr-def}[symmetric]$  **by** *simp*  
**hence**  $3: r = [\text{hd } r]$  **by** (*cases r*) *auto*  
**hence**  $r \neq \mathbf{0}_P$  **unfolding** *univ-poly-zero* **by auto**  
**hence**  $\text{hd } r \in \text{carrier } (\text{ring-of } R) - \{\mathbf{0}_{\text{ring-of } R}\}$   
**using**  $r\text{v-carr}$  **by** (*intro lead-coeff-carr*) *auto*  
**hence**  $r\text{-unit}: r \in \text{Units } P$  **using**  $3$  *univ-poly-units[OF carrier-is-subfield]*  
**by auto**  
**hence**  $\text{inv-r-carr}: \text{inv}_P r \in \text{carrier } P$  **by** *simp*

**have**  $0: x^{-1} \text{Cpoly-mod-ring } R f = \text{pmod}_C R (r^{-1} \text{Cpoly } R * \text{Cpoly } R u) f$   
**by** (*simp add:poly-mod-ring-def uvr-def[symmetric]*)  
**also have**  $\dots = \text{pmod}_C R (\text{inv}_P r \otimes_P u) f$   
**using** *r-unit unfolding domain-cD[OF d-poly]*  
**by** (*subst domain-cD[OF d-poly]*) (*simp-all add:ring-of-poly[OF ring-c]*)  
**also have**  $\dots = \text{pmod} (\text{inv}_P r \otimes_P u) f$   
**using** *ruv-carr inv-r-carr* **by** (*intro pmod-c[OF field-R] f-carr*)  
*simp*  
**finally have**  $0: x^{-1} \text{Cpoly-mod-ring } R f = \text{pmod} (\text{inv}_P r \otimes_P u) f$   
**by** *simp*

**show**  $x^{-1} \text{Cpoly-mod-ring } R f \in \text{carrier} (\text{ring-of} (\text{poly-mod-ring } R f))$   
**using** *ruv-carr r-unit unfolding 0* **by** (*intro poly-mod-ring-carr*)  
*simp*

**have**  $4: \text{degree } 1_P < \text{degree } f$  **unfolding** *univ-poly-one* **using** *deg-f*  
**by** *auto*

**have**  $f \text{ divides}_P \text{inv}_P r \otimes_P f \otimes_P v$   
**using** *inv-r-carr ruv-carr f-carr*  
**by** (*intro dividesI[where c=inv\_P r \otimes\_P v]*) (*simp-all, algebra*)  
**hence**  $5: \text{pmod} (\text{inv}_P r \otimes_P f \otimes_P v) f = []$   
**using** *f-carr ruv-carr inv-r-carr*  
**by** (*intro iffD2[OF pmod-zero-iff-pdivides[OF carrier-is-subfield]]*)  
*(auto simp:pdivides-def)*

**have**  $x \otimes_P x^{-1} \text{Cpoly-mod-ring } R f = \text{pmod} (x \otimes_P \text{pmod} (\text{inv}_P r \otimes_P u) f) f$   
**using** *ruv-carr inv-r-carr f-carr unfolding 0*  
**by** (*intro poly-mod-ring-mult x-carr' long-division-closed[OF carrier-is-subfield]*) *simp-all*  
**also have**  $\dots = \text{pmod} (x \otimes_P (\text{inv}_P r \otimes_P u)) f$   
**using** *ruv-carr inv-r-carr f-carr* **by** (*intro pmod-mult-right[symmetric] x-carr'*) *auto*  
**also have**  $\dots = \text{pmod} (\text{inv}_P r \otimes_P (x \otimes_P u)) f$   
**using** *x-carr' ruv-carr inv-r-carr* **by** (*intro arg-cong2[where f=pmod] refl*) (*simp, algebra*)  
**also have**  $\dots = \text{pmod} (\text{inv}_P r \otimes_P (r \ominus_P f \otimes_P v)) f$  **using** *ruv-carr f-carr x-carr'*  
**by** (*intro arg-cong2[where f=pmod] arg-cong2[where f=(\otimes\_P)] refl*) (*simp add:r-eq, algebra*)  
**also have**  $\dots = \text{pmod} (\text{inv}_P r \otimes_P r \ominus_P \text{inv}_P r \otimes_P f \otimes_P v) f$   
**using** *ruv-carr inv-r-carr f-carr* **by** (*intro arg-cong2[where f=pmod] refl*) (*simp, algebra*)  
**also have**  $\dots = \text{pmod } 1_P f \oplus_P \text{pmod} (\ominus_P (\text{inv}_P r \otimes_P f \otimes_P v)) f$   
**using** *ruv-carr inv-r-carr f-carr* **unfolding** *d.Units-l-inv[OF*

*r-unit*] *a-minus-def*  
**by** (*intro long-division-add*[*OF carrier-is-subfield*]) *simp-all*  
**also have** ... =  $\mathbf{1}_P \ominus_P \text{pmod} (\text{inv}_P r \otimes_P f \otimes_P v) f$   
**using** *ruv-carr f-carr inv-r-carr* **unfolding** *a-minus-def*  
**by** (*intro arg-cong2*[**where**  $f = (\oplus_P)$ ] *pmod-const*[*OF carrier-is-subfield*] 4) *simp-all*  
**also have** ... =  $\mathbf{1}_P \ominus_P \mathbf{0}_P$  **unfolding** 5 *univ-poly-zero* **by** *simp*  
**also have** ... =  $\mathbf{1}_{\text{ring-of } (\text{poly-mod-ring } R f)}$  **unfolding** *poly-mod-ring-one*  
**by** *algebra*  
**finally show**  $x \otimes_{\text{ring-of } (\text{poly-mod-ring } R f)} x^{-1} \in \text{poly-mod-ring } R f$   
=  $\mathbf{1}_{\mathcal{P}}$  **by** *simp*  
**qed**  
**ultimately show** *?thesis* **using** *cring-poly-mod-ring-1* **by** (*intro cring-cI*)  
**qed**

**end**

**lemma** *field-c-poly-mod-ring:*

**assumes** *field-R:* *field<sub>C</sub> R*

**assumes** *monic-irreducible-poly* (*ring-of R*) *f*

**shows** *field<sub>C</sub> (poly-mod-ring R f)*

**proof** –

**interpret** *field ring-of R* **using** *field-R* **unfolding** *field<sub>C</sub>-def* **by** *auto*

**have** *f-carr:*  $f \in \text{carrier } (\text{poly-ring } (\text{ring-of } R))$

**using** *assms(2) monic-poly-carr* **unfolding** *monic-irreducible-poly-def*  
**by** *auto*

**have** *deg-f:* *degree f > 0* **using** *monic-poly-min-degree assms(2)* **by** *fastforce*

**have** *f-irred:* *pirreducible<sub>ring-of R</sub> (carrier (ring-of R)) f*

**using** *assms(2)* **unfolding** *monic-irreducible-poly-def* **by** *auto*

**interpret** *r:field poly-ring (ring-of R) Quot (PID<sub>poly-ring (ring-of R)</sub>*  
*f)*

**using** *f-irred f-carr iffD2*[*OF rupture-is-field-iff-pirreducible*[*OF carrier-is-subfield*]]

**unfolding** *rupture-def* **by** *blast*

**have** *field (ring-of (poly-mod-ring R f))*

**using** *r.ring-iso-imp-img-field*[*OF poly-mod-ring-iso-inv*[*OF field-R f-carr deg-f*]]

**using** *cring-poly-mod-ring-1(1)*[*OF field-R f-carr deg-f*] **by** *simp*

**moreover have** *cring<sub>C</sub> (poly-mod-ring R f)*

**by** (*rule cring-c-poly-mod-ring*[*OF field-R f-carr deg-f*])

**ultimately show** *?thesis* **unfolding** *field<sub>C</sub>-def domain<sub>C</sub>-def* **using**  
*field.axioms(1)* **by** *blast*  
**qed**

**lemma** *enum-c-poly-mod-ring*:

**assumes** *enum<sub>C</sub> R ring<sub>C</sub> R*

**shows** *enum<sub>C</sub> (poly-mod-ring R f)*

**proof** (*rule enum-cI*)

**let** *?l = degree f*

**let** *?b = idx-size R*

**let** *?S = carrier (ring-of (poly-mod-ring R f))*

**note** *bij-0 = bij-betw-poly-enum*[**where** *l=degree f, OF assms(1,2)*]

**have** *?S = {xs ∈ carrier (poly-ring (ring-of R)). length xs ≤ ?l}*

**unfolding** *ring-of-poly*[*OF assms(2),symmetric*] *poly-mod-ring-def*

**by** (*simp add:ring-of-def*)

**hence** *bij-1:bij-betw (poly-enum R (degree f)) {..*idx-size R* ^ *degree f*}* *?S*

**using** *bij-0* **by** *simp*

**hence** *bij-2:bij-betw (idx-enum (poly-mod-ring R f)) {..*idx-size R* ^ *degree f*}* *?S*

**unfolding** *poly-mod-ring-def* **by** *simp*

**have** *order (ring-of (poly-mod-ring R f)) = card ?S*

**unfolding** *Coset.order-def* **by** *simp*

**also have** *... = card {..*idx-size R* ^ *degree f*}* **using** *bij-2* **by** (*metis*  
*bij-betw-same-card*)

**finally have** *ord-poly-mod-ring: order (ring-of (poly-mod-ring R f)) = *idx-size R* ^ *degree f**

**by** *simp*

**show** *finite ?S* **using** *bij-2* *bij-betw-finite* **by** *blast*

**show** *idx-size (poly-mod-ring R f) = order (ring-of (poly-mod-ring R f))*

**unfolding** *ord-poly-mod-ring* **by** (*simp add:poly-mod-ring-def*)

**show** *bij-betw (idx-enum (poly-mod-ring R f)) {..*order (ring-of (poly-mod-ring R f))*}* *?S*

**using** *bij-2* *ord-poly-mod-ring* **by** *auto*

**show** *idx-enum-inv (poly-mod-ring R f) (idx-enum (poly-mod-ring R f) x) = x* (**is** *?L = -*)

**if** *x < order (ring-of (poly-mod-ring R f))* **for** *x*

**proof** –

**have** *?L = poly-enum-inv R (degree f) (poly-enum R (degree f) x)*

**unfolding** *poly-mod-ring-def* **by** *simp*

**also have** *... = the-inv-into {..*?b* ^ *?l*}* (*poly-enum R ?l*) (*poly-enum R ?l x*)

**using** *that ord-poly-mod-ring*

**by** (*intro poly-enum-inv*[*OF assms(1,2),symmetric*] *bij-betw-apply*[*OF*

```

bij-0]) auto
  also have ... = x
  using that ord-poly-mod-ring by (intro the-inv-into-f-f bij-betw-imp-inj-on[OF
bij-0]) auto
  finally show ?thesis by simp
qed
qed
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 2<sup>64</sup>*).

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 ⇒ nat ⇒ (nat ⇒ bool) ⇒ nat
where
  find-first l u f = (
    if (l ≥ 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))
by pat-completeness auto

```

```

termination by (relation Wellfounded.measure (λ(l, u, -). u - l))
auto

```

**lemma** *Min-subset-eq*:

```

assumes A ⊆ B finite B A ≠ {} ∀ x ∈ B. ∃ y ∈ A. y ≤ x
shows Min A = Min B

```

**proof** (rule order-antisym)

```

show Min A ≤ Min B

```

```

  by (metis assms Min.coboundedI basic-trans-rules(23) bot-unique
infinite-super obtains-Min)

```

**next**

```

show Min B ≤ Min A

```

by (rule *Min-antimono*[*OF assms*(1,3,2)])  
 qed

**lemma** *find-first-eq*:  
 assumes *mono f*  
 shows  $\text{find-first } l \ u \ f = \text{Min } (\{x. f \ x\} \cap \{l..<u\} \cup \{u\})$   
 using *assms*  
**proof** (*induction l u f rule:find-first.induct*)  
 case (1 *l u f*)  
 define *x* where  $x = (l + u) \ \text{div } 2$   
 have *x-ran*:  $x \in \{l..<u\}$  if  $l < u$  using *that* unfolding *x-def* by  
*auto*  
 consider (a)  $l \geq u$  | (b)  $f \ x \wedge l < u$  | (c)  $\neg f \ x \wedge l < u$  by *linarith*  
 thus ?*case*  
**proof** (*cases*)  
 case a thus ?*thesis* by *simp*  
 next  
 case b  
 hence  $\text{find-first } l \ u \ f = \text{find-first } l \ x \ f$  by (*simp add:x-def Let-def*)  
 also have  $\dots = \text{Min } (\{x. f \ x\} \cap \{l..<x\} \cup \{x\})$  using 1(1,3)  
*x-def b* by *simp*  
 also have  $\dots = \text{Min } (\{x. f \ x\} \cap \{l..<u\} \cup \{u\})$   
 using *b x-ran* by (*intro Min-subset-eq*) *auto*  
 finally show ?*thesis* by *simp*  
 next  
 case c  
 have  $\neg f \ y$  if  $y \leq x$  for *y* using *mono-onD*[*OF 1(3) - - that*] *that*  
*c* by *simp*  
 hence  $*:y \geq \text{Suc } x$  if  $f \ y$  for *y* using *that*  
 by (*meson not-less-eq-eq*)  
 have  $\text{find-first } l \ u \ f = \text{find-first } (x+1) \ u \ f$  using *c* by (*simp*  
*add:x-def Let-def*)  
 also have  $\dots = \text{Min } (\{x. f \ x\} \cap \{(x+1)..<u\} \cup \{u\})$  using 1(2,3)  
*x-def c* by *simp*  
 also have  $\dots = \text{Min } (\{x. f \ x\} \cap \{l..<u\} \cup \{u\})$   
 using  $* \ c \ x\text{-ran}$  by (*intro arg-cong*[**where**  $f = \text{Min}$ ]) *force*  
 finally show ?*thesis* by *simp*  
 qed  
 qed

**lemma** *nth-root-nat-fast*[*code*]:  $\text{nth-root-nat } e \ n = (\text{if } e = 0 \ \text{then } 0 \ \text{else } \text{find-first } 0 \ (n+1) \ (\lambda x. x^e > n) - 1)$   
**proof** (*cases e > 0*)  
 case *True*  
  
 have *mono*: *mono* ( $\lambda x. x^e > n$ )  
 using *power-mono*[*OF - zero-le*] *order-less-le-trans*  
 by (*intro monoI le-boolI*) *blast*

**have**  $\text{nth-root-nat } e \ n \leq \text{nth-root-nat } e \ (n \hat{=} e)$  **if**  $n > 0$   
**using** *True that by (intro nth-root-nat-mono self-le-power) auto*

**hence**  $0 : \text{nth-root-nat } e \ n \leq n$   
**using**  $\text{nth-root-nat-nth-power}[OF \ True]$  **by** *(cases n = 0) auto*

**have**  $n < n+1$  **by** *simp*  
**also have**  $\dots \leq (n+1) \hat{=} e$  **by** *(intro self-le-power True) auto*  
**finally have**  $1 : n < (n+1) \hat{=} e$  **by** *simp*

**have**  $x \hat{=} e > n$  **if**  $x \geq \text{nth-root-nat } e \ n + 1$  **for**  $x$   
**using** *that nth-root-nat-ge[OF True]*  
**by** *(metis add-Suc-right nat-arith.rule0 not-less not-less-eq-eq numeral-nat(7))*

**hence**  $\text{nth-root-nat } e \ n+1 \leq x \longleftrightarrow x \hat{=} e > n$  **for**  $x$   
**using**  $\text{nth-root-nat-less}[OF \ True]$  **by** *fastforce*

**hence**  $\{\text{nth-root-nat } e \ n+1 .. (n+1)\} = \{x. x \hat{=} e > n \wedge x \leq n+1\}$   
**unfolding**  $\text{atLeastAtMost-def Int-def}$  **by** *simp*  
**also have**  $\dots = \{x. x \hat{=} e > n \wedge x < n+1\} \cup \{n+1\}$  **using** *True 1*  
**by** *auto*  
**finally have**  $1 : \{\text{nth-root-nat } e \ n+1 .. (n+1)\} = \{x. x \hat{=} e > n \wedge x < n+1\} \cup \{n+1\}$  **by** *simp*

**have**  $\text{nth-root-nat } e \ n = \text{Inf } \{\text{nth-root-nat } e \ n+1 .. (n+1)\} - 1$   
**using**  $0$  **by** *(subst cInf-atLeastAtMost) auto*  
**also have**  $\dots = \text{Inf } (\{x. x \hat{=} e > n \wedge x < n+1\} \cup \{n+1\}) - 1$   
**using**  $1$  **by** *simp*  
**also have**  $\dots = \text{Min } (\{x. x \hat{=} e > n \wedge x < n+1\} \cup \{n+1\}) - 1$   
**by** *(subst cInf-eq-Min) auto*  
**also have**  $\dots = (\text{if } e = 0 \text{ then } 0 \text{ else find-first } 0 \ (n+1) \ (\lambda x. x \hat{=} e > n) - 1)$   
**using** *True unfolding find-first-eq[OF mono] Int-def* **by** *simp*  
**finally show** *?thesis* **by** *simp*

**next**  
**case** *False*  
**thus** *?thesis* **by** *simp*  
**qed**

**lemma**  $\text{is-nth-power-nat-fast}[code] : \text{is-nth-power-nat } e \ n \longleftrightarrow ((\text{nth-root-nat } e \ n) \hat{=} e = n)$   
**proof** -  
**have**  $((\text{nth-root-nat } e \ n) \hat{=} e = n) \longrightarrow \text{is-nth-power } e \ n$   
**unfolding**  $\text{is-nth-power-def}$  **by** *metis*

**moreover have**  $((\text{nth-root-nat } e \ n) \hat{=} e = n)$  **if**  $\text{is-nth-power } e \ n \ e > 0$   
**using**  $\text{nth-root-nat-nth-power}[OF \ \text{that}(2)]$   $\text{that}(1)$

```

    unfolding is-nth-power-def by auto

    moreover have ((nth-root-nat 0 n) ^ 0 = n) if is-nth-power (0::nat)
    n
    using that unfolding is-zeroth-power by simp

    ultimately show ?thesis unfolding is-nth-power-nat-def by blast
qed

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 = 1C A#[ 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 -
  let ?f = ( $\lambda x$ . 1C R # map ( $\lambda j$ . idx-enum R (x j)) (rev [ 0.. $d$  ] ))
  let ?R = ring-of R

  note select-bij = enum-cD(3)[OF assms(2)]
  note fin-carr = enum-cD(1)[OF assms(2)]

```

**note**  $fo = field-cD[OF\ assms(1)]$

**interpret** *finite-field ring-of R*  
**using** *fin-carr assms(1)* **unfolding** *finite-field-def finite-field-axioms-def field<sub>C</sub>-def* **by** *auto*

**have**  $1:enum-monic-poly\ R\ d = ?f \circ (\lambda v. \lambda x \in \{..<d\}. nth-digit\ v\ x\ (order\ (ring-of\ R)))$   
**unfolding** *enum-monic-poly-def comp-def enum-cD[OF assms(2)]*  
**by** *(intro ext arg-cong2[where f=(#)] refl map-cong) auto*

**have**  $2: ?f = (\lambda x. 1_{C\ R}\ \# \ map\ x\ (rev\ [0..<d])) \circ (\lambda x. \lambda i \in \{..<d\}. id_x-enum\ R\ (x\ i))$   
**unfolding** *comp-def* **by** *auto*

**have**  $3: (\lambda x. \mathbf{1}_{ring-of\ R}\ \# \ map\ x\ (rev\ [0..<d])) = (\lambda x. \mathbf{1}_{ring-of\ R}\ \# \ x) \circ rev \circ (\lambda x. map\ x\ [0..<d])$   
**unfolding** *comp-def* **by** *(intro ext) (simp add:rev-map)*

**have** *ap-bij: bij-betw ((#) 1<sub>?R</sub>) {x. set x  $\subseteq$  carrier ?R  $\wedge$  length x = d} {f. monic-poly ?R f  $\wedge$  degree f = d}*  
**using** *list.collapse* **unfolding** *monic-poly-def univ-poly-carrier[symmetric] polynomial-def*  
**by** *(intro bij-betwI[where g=tl]) (fastforce intro:in-set-tlD)+*

**have** *rev-bij:*  
*bij-betw rev {x. set x  $\subseteq$  carrier ?R  $\wedge$  length x = d} {x. set x  $\subseteq$  carrier ?R  $\wedge$  length x = d}*  
**by** *(intro bij-betwI[where g=rev]) auto*

**have** *bij-betw*  $(\lambda x. \mathbf{1}_{?R}\ \# \ map\ x\ (rev\ [0..<d])) (\{..<d\} \rightarrow_E\ carrier\ ?R) \{f. monic-poly\ ?R\ f \wedge degree\ f = d\}$   
**unfolding**  $3$  **by** *(intro bij-betw-trans[OF lists-bij] bij-betw-trans[OF rev-bij] ap-bij)*  
**hence** *bij-betw*  $?f (\{..<d\} \rightarrow_E \{..<order\ ?R\}) \{f. monic-poly\ ?R\ f \wedge degree\ f = d\}$   
**unfolding**  $2$  **by** *(intro bij-betw-trans[OF lift-bij-betw[OF select-bij]]) (simp add:fo)*  
**thus** *?thesis*  
**unfolding**  $1$  **by** *(intro bij-betw-trans[OF nth-digit-bij])*

**qed**

**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 ← spmf-of-set {.. $\widehat{p}^n$ };
    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 (( $\lambda k. \neg$ rabin-test (mod-ring p) (f k))) ( $\lambda x. x + 1$ ) 0))

```

```

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

```

```

lemma cost-tick: cost (map-option ( $\lambda(x, c). (x, \text{Suc } c)$ ) c) = eSuc (cost c)
  by (cases c) (auto simp: cost-def eSuc-enat)

```

```

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)
  unfolding finite-field-def finite-field-axioms-def
  by (intro mod-ring-is-field conjI mod-ring-finite p-prime)

```

```

private lemmas field-ops = field-cD[OF field-c]

```

```

private lemma S-fin: finite S
  unfolding S-def

```

**using** *enum-monic-poly*[*OF field-c enum-c, where d=n*]  
*bij-betw-finite* **by** *auto*

**private lemma** *T-sub-S*:  $T \subseteq S$   
**unfolding** *S-def T-def monic-irreducible-poly-def* **by** *auto*

**private lemma** *T-card-gt-0*:  $\text{real}(\text{card } T) > 0$   
**proof** –  
**have**  $0 < \text{real}(\text{order}(\text{ring-of}(\text{mod-ring } p))) \wedge n / (2 * \text{real } n)$   
**using** *n-gt-0 finite-field-min-order* **by** (*intro divide-pos-pos*) (*simp-all*)  
**also have**  $\dots \leq \text{real}(\text{card } T)$  **unfolding** *T-def* **by** (*intro card-irred-gt-0 n-gt-0*)  
**finally show**  $\text{real}(\text{card } T) > 0$  **by** *auto*  
**qed**

**private lemma** *S-card-gt-0*:  $\text{real}(\text{card } S) > 0$   
**proof** –  
**have**  $0 < \text{card } T$  **using** *T-card-gt-0* **by** *simp*  
**also have**  $\dots \leq \text{card } S$  **by** (*intro card-mono T-sub-S S-fin*)  
**finally have**  $0 < \text{card } S$  **by** *simp*  
**thus** *?thesis* **by** *simp*  
**qed**

**private lemma** *S-ne*:  $S \neq \{\}$  **using** *S-card-gt-0* **by** *auto*

**private lemma** *sample-irreducible-poly-step-aux*:  
*do* {  
 $k \leftarrow \text{spmf-of-set } \{..<p \wedge n\}$ ;  
 $\text{let } \text{poly} = \text{enum-monic-poly}(\text{mod-ring } p) \ n \ k$ ;  
*if* *rabin-test* (*mod-ring* *p*) *poly* *then return-spmf* (*poly,c*) *else x*  
} =  
*do* {  
 $\text{poly} \leftarrow \text{spmf-of-set } S$ ;  
*if* *monic-irreducible-poly* (*ring-of* (*mod-ring* *p*)) *poly*  
*then return-spmf* (*poly,c*)  
*else x*  
}  
(*is* *?L* = *?R*)

**proof** –  
**have**  $\text{order}(\text{ring-of}(\text{mod-ring } p)) = p$   
**unfolding** *Finite-Fields-Mod-Ring-Code.mod-ring-def Coset.order-def ring-of-def* **by** *simp*  
**hence**  $0 : \text{spmf-of-set } S = \text{map-spmf}(\text{enum-monic-poly}(\text{mod-ring } p) \ n) (\text{spmf-of-set } \{..<p \wedge n\})$   
**using** *enum-monic-poly*[*OF field-c enum-c, where d=n*] **unfolding** *bij-betw-def S-def*  
**by** (*subst map-spmf-of-set-inj-on*) *auto*

**have**  $?L = \text{do } \{f \leftarrow \text{spmf-of-set } S; \text{if } \text{rabin-test}(\text{mod-ring } p) \ f \text{ then}$

```

return-spmf (f,c) else x}
  unfolding 0 bind-map-spmf by (simp add:Let-def comp-def)
  also have ... = ?R
  using set-spmf-of-set-finite[OF S-fin]
  by (intro bind-spmf-cong refl if-cong rabin-test field-c enum-c) (simp
add:S-def)
  finally show ?thesis by simp
qed

```

**private 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)
  }
by (subst sample-irreducible-poly.simps) (simp add:sample-irreducible-poly-step-aux)

```

**private lemma** *sample-irreducible-poly-aux-1*:

```

ord-spmf (=) (map-spmf fst (sample-irreducible-poly p n)) (spmf-of-set
T)

```

**proof** (*induction rule:sample-irreducible-poly.fixp-induct*)

case 1 thus ?case by simp

next

case 2 thus ?case by simp

next

case (3 rec)

let ?f = monic-irreducible-poly (ring-of (mod-ring p))

have real (card (S ∩ -{x. ?f x})) = real (card (S - T))

unfolding S-def T-def by (intro arg-cong[where f=card] arg-cong[where
f=of-nat]) (auto)

also have ... = real (card S - card T)

by (intro arg-cong[where f=of-nat] card-Diff-subset T-sub-S fi-
nite-subset[OF T-sub-S S-fin])

also have ... = real (card S) - card T

by (intro of-nat-diff card-mono S-fin T-sub-S)

finally have 0:real (card (S ∩ -{x. ?f x})) = real (card S) - card T
by simp

have S-card-gt-0: real (card S) > 0 using S-ne S-fin by auto

have do {f ← spmf-of-set S; if ?f f then return-spmf f else spmf-of-set
T} = spmf-of-set T

(is ?L = ?R)

**proof** (*rule spmf-eqI*)

fix i

have spmf ?L i = spmf (pmf-of-set S ≫=(λx. if ?f x then re-

$\text{turn-spmf } x \text{ else spmf-of-set } T)) \ i$   
**unfolding**  $\text{spmof-of-pmf-pmf-of-set}[OF \ S\text{-fin} \ S\text{-ne}, \ \text{symmetric}]$   
 $\text{spmof-of-pmf-def}$   
**by**  $(\text{simp add:bind-spmf-def bind-map-pmf})$   
**also have**  $\dots = (\int x. (\text{if } ?f \ x \ \text{then of-bool } (x=i) \ \text{else spmf } (\text{spmof-of-set } T) \ i) \ \partial \text{pmf-of-set } S)$   
**unfolding**  $\text{pmf-bind if-distrib if-distribR pmf-return-spmf indicator-def}$  **by**  $(\text{simp cong:if-cong})$   
**also have**  $\dots = (\sum x \in S. (\text{if } ?f \ x \ \text{then of-bool } (x = i) \ \text{else spmf } (\text{spmof-of-set } T) \ i)) / \text{card } S$   
**by**  $(\text{subst integral-pmf-of-set}[OF \ S\text{-ne} \ S\text{-fin}]) \ \text{simp}$   
**also have**  $\dots = (\text{of-bool } (i \in T) + \text{spmof } (\text{spmof-of-set } T) \ i * \text{real } (\text{card } (S \cap -\{x. ?f \ x\}))) / \text{card } S$   
**using**  $S\text{-fin} \ S\text{-ne}$   
**by**  $(\text{subst sum.If-cases}[OF \ S\text{-fin}]) \ (\text{simp add:of-bool-def } T\text{-def } \text{monic-irreducible-poly-def } S\text{-def})$   
**also have**  $\dots = (\text{of-bool } (i \in T) * (1 + \text{real } (\text{card } (S \cap -\{x. ?f \ x\}))) / \text{real } (\text{card } T)) / \text{card } S$   
**unfolding**  $\text{spmof-of-set indicator-def}$  **by**  $(\text{simp add:algebra-simps})$   
**also have**  $\dots = (\text{of-bool } (i \in T) * (\text{real } (\text{card } S) / \text{real } (\text{card } T))) / \text{card } S$   
**using**  $T\text{-card-gt-0}$  **unfolding**  $0$  **by**  $(\text{simp add:field-simps})$   
**also have**  $\dots = \text{of-bool } (i \in T) / \text{real } (\text{card } T)$   
**using**  $S\text{-card-gt-0}$  **by**  $(\text{simp add:field-simps})$   
**also have**  $\dots = \text{spmof } ?R \ i$   
**unfolding**  $\text{spmof-of-set}$  **by**  $\text{simp}$   
**finally show**  $\text{spmof } ?L \ i = \text{spmof } ?R \ i$   
**by**  $\text{simp}$   
**qed**  
**hence**  $\text{ord-spmf } (=)$   
 $(\text{spmof-of-set } S \gg (\lambda x. \text{if } ?f \ x \ \text{then return-spmf } x \ \text{else spmf-of-set } T)) \ (\text{spmof-of-set } T)$   
**by**  $\text{simp}$   
**moreover have**  $\text{ord-spmf } (=)$   
 $(\text{do } \{ \text{poly} \leftarrow \text{spmof-of-set } S; \ \text{if } ?f \ \text{poly} \ \text{then return-spmf } \text{poly} \ \text{else map-spmf } \text{fst } (\text{rec } p \ n)\})$   
 $(\text{do } \{ \text{poly} \leftarrow \text{spmof-of-set } S; \ \text{if } ?f \ \text{poly} \ \text{then return-spmf } \text{poly} \ \text{else spmf-of-set } T\})$   
**using**  $3$  **by**  $(\text{intro bind-spmf-mono'}) \ \text{simp-all}$   
**ultimately have**  $\text{ord-spmf } (=) \ (\text{spmof-of-set } S \gg (\lambda x. \text{if } ?f \ x \ \text{then return-spmf } x \ \text{else map-spmf } \text{fst } (\text{rec } p \ n))) \ (\text{spmof-of-set } T)$   
**using**  $\text{spmof.leq-trans}$  **by**  $\text{force}$   
**thus**  $?case$  **unfolding**  $\text{sample-irreducible-poly-step-aux map-spmf-bind-spmf}$   
**by**  $(\text{simp add:comp-def if-distribR if-distrib spmf.map-comp case-prod-beta cong:if-cong})$   
**qed**

**lemma**  $\text{cost-sample-irreducible-poly}$ :

```

( $\int^+ x. \text{cost } x \text{ } \partial \text{sample-irreducible-poly } p \text{ } n) \leq 2 * \text{real } n$  (is ?L  $\leq$  ?R)
proof –
  let ?f = monic-irreducible-poly (ring-of (mod-ring p))
  let ?a = ( $\lambda t. \text{measure} (\text{sample-irreducible-poly } p \text{ } n) \{ \omega. \text{enat } t < \text{cost } \omega \}$ )
  let ?b = ( $\lambda t. \text{measure} (\text{sample-irreducible-poly } p \text{ } n) \{ \omega. \text{enat } t \geq \text{cost } \omega \}$ )

  define  $\alpha$  where  $\alpha = \text{measure} (\text{pmf-of-set } S) \{ x. ?f x \}$ 
  have  $\alpha \text{-le-1}: \alpha \leq 1$  unfolding  $\alpha$ -def by simp

  have  $1 / (2 * \text{real } n) = (\text{card } S / (2 * \text{real } n)) / \text{card } S$ 
    using S-card-gt-0 by (simp add:algebra-simps)
  also have  $\dots = (\text{real} (\text{order} (\text{ring-of} (\text{mod-ring } p)))^{\wedge} n / (2 * \text{real } n)) / \text{card } S$ 
    unfolding S-def bij-betw-same-card [OF enum-monic-poly [OF field-c
enum-c, where d=n], symmetric]
    by simp
  also have  $\dots \leq \text{card } T / \text{card } S$ 
    unfolding T-def by (intro divide-right-mono card-irred-gt-0 n-gt-0)
  auto
  also have  $\dots = \alpha$ 
    unfolding  $\alpha$ -def measure-pmf-of-set [OF S-ne S-fin]
    by (intro arg-cong2 [where f=(/)] refl arg-cong [where f=of-nat]
arg-cong [where f=card])
    (auto simp: S-def T-def monic-irreducible-poly-def)
  finally have  $\alpha \text{-lb}: 1 / (2 * \text{real } n) \leq \alpha$ 
    by simp
  have  $0 < 1 / (2 * \text{real } n)$  using n-gt-0 by simp
  also have  $\dots \leq \alpha$  using  $\alpha$ -lb by simp
  finally have  $\alpha \text{-gt-0}: \alpha > 0$  by simp

  have a-step-aux:  $\text{norm} (a * b) \leq 1$  if  $\text{norm } a \leq 1$   $\text{norm } b \leq 1$  for
a b :: real
    using that by (simp add:abs-mult mult-le-one)

  have b-eval:  $?b \text{ } t = (\int x. (\text{if } ?f \text{ } x \text{ then } \text{of-bool}(t \geq 1) \text{ else } \text{measure} (\text{sample-irreducible-poly } p \text{ } n) \{ \omega. \text{enat } t \geq \text{eSuc} (\text{cost } \omega) \})) \partial \text{pmf-of-set } S)$ 
    (is ?L1 = ?R1) for t
  proof –
    have  $?b \text{ } t = \text{measure} (\text{bind-spmf} (\text{spmof-of-set } S) (\lambda x. \text{if } ?f \text{ } x \text{ then } \text{return-spmf} (x,1) \text{ else } \text{tick-spmf} (\text{sample-irreducible-poly } p \text{ } n))) \{ \omega. \text{enat } t \geq \text{cost } \omega \}$ 
      by (subst sample-irreducible-poly-step) simp
    also have  $\dots = \text{measure} (\text{bind-pmf} (\text{pmf-of-set } S) (\lambda x. \text{if } ?f \text{ } x \text{ then } \text{return-spmf} (x,1) \text{ else } \text{tick-spmf} (\text{sample-irreducible-poly } p \text{ } n))) \{ \omega. \text{enat } t \geq \text{cost } \omega \}$ 
      unfolding spmof-of-pmf-pmf-of-set [OF S-fin S-ne, symmetric]

```

```

    by (simp add: spmf-of-pmf-def bind-map-pmf bind-spmf-def)
  also have ... = (∫ x. (if ?f x then of-bool(t ≥ 1) else
    measure (tick-spmf (sample-irreducible-poly p n)) {ω. enat t ≥
cost ω})) ∂pmf-of-set S)
  unfolding measure-bind-pmf if-distrib if-distribR emeasure-return-pmf
  by (simp add: indicator-def cost-def comp-def cong-if-cong)
  also have ... = ?R1
  unfolding measure-map-pmf vimage-def
  by (intro arg-cong2[where f=integralL] refl ext if-cong arg-cong2[where
f=measure])
    (auto simp add: vimage-def cost-tick eSuc-enat[symmetric])
  finally show ?thesis by simp
qed

have b-eval-2: ?b t = 1 - (1-α)ˆt for t
proof (induction t)
  case 0
  have ?b 0 = 0 unfolding b-eval by (simp add: enat-0 cong-if-cong
)
  thus ?case by simp
next
  case (Suc t)
  have ?b (Suc t) = (∫ x. (if ?f x then 1 else ?b t) ∂pmf-of-set S)
  unfolding b-eval[of Suc t]
  by (intro arg-cong2[where f=integralL] if-cong arg-cong2[where
f=measure])
    (auto simp add: eSuc-enat[symmetric])
  also have ... = (∫ x. indicator {x. ?f x} x + ?b t * indicator {x.
¬?f x} x ∂pmf-of-set S)
  by (intro Bochner-Integration.integral-cong) (auto simp: algebra-simps)
  also have ... = (∫ x. indicator {x. ?f x} x ∂pmf-of-set S) +
    (∫ x. ?b t * indicator {x. ¬?f x} x ∂pmf-of-set S)
  by (intro Bochner-Integration.integral-add measure-pmf.integrable-const-bound[where
B=1]
    AE-pmfI a-step-aux) auto
  also have ... = α + ?b t * measure (pmf-of-set S) {x. ¬?f x}
  unfolding α-def by simp
  also have ... = α + (1-α) * ?b t
  unfolding α-def
  by (subst measure-pmf.prob-compl[symmetric]) (auto simp: Compl-eq-Diff-UNIV
Collect-neg-eq)
  also have ... = 1 - (1-α)ˆSuc t
  unfolding Suc by (simp add: algebra-simps)
  finally show ?case by simp
qed

hence a-eval: ?a t = (1-α)ˆt for t
proof -
  have ?a t = 1 - ?b t

```

**by** (*simp add: measure-pmf.prob-compl[symmetric] Compl-eq-Diff-UNIV[symmetric]*  
*Collect-neg-eq[symmetric] not-le*)  
**also have** ... =  $(1-\alpha)^{\wedge}t$   
**unfolding** *b-eval-2* **by** *simp*  
**finally show** *?thesis* **by** *simp*  
**qed**

**have**  $?L = (\sum t. \text{emeasure } (\text{sample-irreducible-poly } p \ n) \ \{\omega. \text{ enat } t < \text{cost } \omega\})$   
**by** (*subst nn-integral-enat-function*) *simp-all*  
**also have** ... =  $(\sum t. \text{ennreal } (?a \ t))$   
**unfolding** *measure-pmf.emeasure-eq-measure* **by** *simp*  
**also have** ... =  $(\sum t. \text{ennreal } ((1-\alpha)^{\wedge}t))$   
**unfolding** *a-eval* **by** (*intro arg-cong[where f=suminf] ext*) (*simp add:  $\alpha$ -def ennreal-mult'*)  
**also have** ... =  $\text{ennreal } (1 / (1-(1-\alpha)))$   
**using**  *$\alpha$ -le-1  $\alpha$ -gt-0*  
**by** (*intro arg-cong2[where f=(\*)] refl suminf-ennreal-eq geometric-sums*) *auto*  
**also have** ... =  $\text{ennreal } (1 / \alpha)$  **using**  *$\alpha$ -le-1  $\alpha$ -gt-0* **by** *auto*  
**also have** ...  $\leq ?R$   
**using**  *$\alpha$ -lb  $n$ -gt-0  $\alpha$ -gt-0* **by** (*intro ennreal-leI*) (*simp add:field-simps*)  
**finally show** *?thesis* **by** *simp*  
**qed**

**private lemma** *weight-sample-irreducible-poly:*  
*weight-spmf (sample-irreducible-poly p n) = 1 (is ?L = ?R)*  
**proof** (*rule ccontr*)  
**assume**  $?L \neq 1$   
**hence**  $?L < 1$  **using** *less-eq-real-def weight-spmf-le-1* **by** *blast*  
**hence**  $(\infty::\text{ennreal}) = \infty * \text{ennreal } (1-?L)$  **by** *simp*  
**also have** ... =  $\infty * \text{ennreal } (\text{pmf } (\text{sample-irreducible-poly } p \ n) \ \text{None})$   
**unfolding** *pmf-None-eq-weight-spmf[symmetric]* **by** *simp*  
**also have** ... =  $(\int^{+x}. \infty * \text{indicator } \{\text{None}\} \ x \ \partial \text{sample-irreducible-poly } p \ n)$   
**by** (*simp add:emeasure-pmf-single*)  
**also have** ...  $\leq (\int^{+x}. \text{cost } x \ \partial \text{sample-irreducible-poly } p \ n)$   
**unfolding** *cost-def* **by** (*intro nn-integral-mono*) (*auto simp:indicator-def*)  
**also have** ...  $\leq 2 * \text{real } n$  **by** (*intro cost-sample-irreducible-poly*)  
**finally have**  $(\infty::\text{ennreal}) \leq 2 * \text{real } n$  **by** *simp*  
**thus** *False* **using** *linorder-not-le* **by** *fastforce*  
**qed**

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

**have**  $?L = \text{spmf-of-set } T$  **using** *weight-sample-irreducible-poly*  
**by** (*intro eq-iff-ord-spmf sample-irreducible-poly-aux-1*) (*auto intro:weight-spmf-le-1*)  
**thus** *?thesis* **unfolding** *T-def* **by** *simp*  
**qed**

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

**have**  $\text{ex}:\exists k. ?f k \in T \wedge k < \text{order (ring-of (mod-ring p))}^{\wedge n}$   
**proof** (*rule ccontr*)  
**assume**  $\nexists k. ?f k \in T \wedge k < \text{order (ring-of (mod-ring p))}^{\wedge n}$   
**hence**  $?f \text{ ' } \{.. < \text{order (ring-of (mod-ring p))}^{\wedge n}\} \cap T = \{\}$  **by**  
*auto*  
**hence**  $S \cap T = \{\}$   
**unfolding** *S-def* **using** *bij-betw-imp-surj-on[OF enum-monic-poly[OF field-c enum-c]]* **by** *auto*  
**hence**  $T = \{\}$  **using** *T-sub-S* **by** *auto*  
**thus** *False* **using** *T-card-gt-0* **by** *simp*  
**qed**

**then obtain**  $k :: \text{nat}$  **where** *k-def: ?f k \in T \forall j < k. ?f j \notin T*  
**using** *exists-least-iff[where P=\lambda x. ?f x \in T]* **by** *auto*

**have** *k-ub: k < order (ring-of (mod-ring p))<sup>^n</sup>*  
**using** *ex k-def(2)* **by** (*meson dual-order.strict-trans1 not-less*)

**have** *a: monic-irreducible-poly (ring-of (mod-ring p)) (?f k)*  
**using** *k-def(1)* **unfolding** *T-def* **by** *simp*  
**have** *b: monic-poly (ring-of (mod-ring p)) (?f j) degree (?f j) = n* **if**  
 $j \leq k$  **for**  $j$   
**proof** –  
**have**  $j < \text{order (ring-of (mod-ring p))}^{\wedge n}$  **using** *k-ub that* **by** *simp*  
**hence**  $?f j \in S$  **unfolding** *S-def* **using** *bij-betw-apply[OF enum-monic-poly[OF field-c enum-c]]* **by** *auto*  
**thus** *monic-poly (ring-of (mod-ring p)) (?f j) degree (?f j) = n*  
**unfolding** *S-def* **by** *auto*  
**qed**

**have** *c: ¬monic-irreducible-poly (ring-of (mod-ring p)) (?f j)* **if**  $j < k$  **for**  $j$   
**using** *b[of j] that k-def(2)* **unfolding** *T-def* **by** *auto*

**have** *2: while ((λk. ¬rabin-test (mod-ring p) (?f k)) (λx. x + 1) (k-j) = k* **if**  $j \leq k$  **for**  $j$

```

using that proof (induction j)
  case 0
    have rabin-test (mod-ring p) (?f k) by (intro iffD2[OF rabin-test]
a b field-c enum-c) auto
    thus ?case by (subst while-unfold) simp
  next
    case (Suc j)
    hence ¬rabin-test (mod-ring p) (?f (k - Suc j))
      using b c by (subst rabin-test[OF field-c enum-c]) auto
    moreover have Suc (Suc (k - Suc j)) = Suc (k - j) using Suc
by simp
    ultimately show ?case using Suc(1) by (subst while-unfold) simp
  qed

```

```

have  $\exists$ :while (( $\lambda k$ . ¬rabin-test (mod-ring p) (?f k))) ( $\lambda x$ . x + 1) 0
= k
  using 2[of k] by simp

```

```

have ?f k  $\in$  T using a b unfolding T-def by auto
hence res  $\in$  T unfolding res-def find-irreducible-poly.simps Let-def
 $\exists$  by simp
thus monic-irreducible-poly (ring-of (mod-ring p)) res degree res =
n unfolding T-def by auto
qed

```

```

lemma monic-irred-poly-set-nonempty-finite:
  {f. monic-irreducible-poly (ring-of (mod-ring p)) f  $\wedge$  degree f = n}
 $\neq$  {} (is ?R1)
  finite {f. monic-irreducible-poly (ring-of (mod-ring p)) f  $\wedge$  degree f
= n} (is ?R2)
proof -
  have card T > 0 using T-card-gt-0 by auto
  hence T  $\neq$  {} finite T using card-ge-0-finite by auto
  thus ?R1 ?R2 unfolding T-def by auto
qed

```

**end**

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

```

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

```

```

lemma split-power-result:
  assumes (x,e) = split-power n
  shows n =  $x^e \wedge k$ . n > 1  $\implies$  k > e  $\implies$  ¬is-nth-power k n
proof -

```

```

define es where es = filter ( $\lambda x. \text{is-nth-power-nat } x \ n$ ) ( $1 \# [2..<\text{floorlog } 2 \ n]$ )
define m where m = max 2 (floorlog 2 n)

have 0: x < m if that0: is-nth-power-nat x n > 1 for x
proof (rule ccontr)
  assume a:  $\neg(x < m)$ 
  obtain y where n-def: n =  $y \hat{x}$  using that0 is-nth-power-def
is-nth-power-nat-def by auto
  have y  $\neq 0$  using that(2) unfolding n-def
  by (metis (mono-tags) nat-power-eq-Suc-0-iff not-less0 power-0-left
power-inject-exp)
  moreover have y  $\neq 1$  using that(2) unfolding n-def by auto
  ultimately have y-ge-2: y  $\geq 2$  by simp
  have n <  $2^{\text{floorlog } 2 \ n}$  using that floorlog-bounds by simp
  also have  $\dots \leq 2^{\hat{x}}$  using a unfolding m-def by (intro power-increasing)
auto
  also have  $\dots \leq y \hat{x}$  using y-ge-2 by (intro power-mono) auto
  also have  $\dots = n$  using n-def by auto
  finally show False by simp
qed

have 1: m = 2 if  $\neg(n > 1)$ 
proof –
  have floorlog 2 n  $\leq 2$  using that by (intro floorlog-leI) auto
  thus ?thesis unfolding m-def by auto
qed

have 2: n = 1 if is-nth-power-nat 0 n using that by (simp add:
is-nth-power-nat-code)

have set es =  $\{x \in \text{insert } 1 \ \{2..<\text{floorlog } 2 \ n\}. \text{is-nth-power-nat } x \ n\}$ 
unfolding es-def by auto
also have  $\dots = \{x. x \neq 0 \wedge x < m \wedge \text{is-nth-power-nat } x \ n\}$  unfolding
m-def by auto
also have  $\dots = \{x. \text{is-nth-power-nat } x \ n \wedge (n > 1 \vee x = 1)\}$ 
using 0 1 2 zero-neq-one by (intro Collect-cong iffI conjI) fast-force+
finally have set-es: set es =  $\{x. \text{is-nth-power-nat } x \ n \wedge (n > 1 \vee x = 1)\}$ 
by simp

have is-nth-power-nat 1 n unfolding is-nth-power-nat-def by simp
hence es-ne: es  $\neq []$  unfolding es-def by auto

have sorted: sorted es unfolding es-def by (intro sorted-wrt-filter)
simp

have e-def: e = last es and x-def: x = nth-root-nat e n
using assms unfolding es-def split-power-def by (simp-all add:Let-def)

```

hence *e-in-set-es*:  $e \in \text{set } es$  **unfolding** *e-def* **using** *es-ne* **by** (*intro last-in-set*) *auto*

**have** *e-max*:  $x \leq e$  **if** *that1*:  $x \in \text{set } es$  **for**  $x$

**proof** –

**obtain**  $k$  **where**  $k < \text{length } es$   $x = es ! k$  **using** *that1* **by** (*metis in-set-conv-nth*)

**moreover have**  $e = es ! (\text{length } es - 1)$  **unfolding** *e-def* **using** *es-ne last-conv-nth* **by** *auto*

**ultimately show** *?thesis* **using** *sorted-nth-mono[OF sorted]* *es-ne* **by** *simp*

**qed**

**have**  $\exists \text{is-nth-power-nat } e \ n \wedge (1 < n \vee e = 1)$  **using** *e-in-set-es* **unfolding** *set-es* **by** *simp*

hence  $e > 0$  **using**  $\exists$  *zero-neq-one* **by** *fast*

**thus**  $n = x^e$  **using**  $\exists$  **unfolding** *x-def* **using** *nth-root-nat-nth-power* **by** (*metis is-nth-power-nat-code nth-root-nat-naive-code power-eq-0-iff*)

**show**  $\neg \text{is-nth-power } k \ n$  **if**  $n > 1$   $k > e$  **for**  $k$

**proof** (*rule ccontr*)

**assume**  $\neg(\neg \text{is-nth-power } k \ n)$

hence  $k \in \text{set } es$  **using** *that* **unfolding** *set-es is-nth-power-nat-def* **by** *auto*

hence  $k \leq e$  **using** *e-max* **by** *auto*

**thus** *False* **using** *that(2)* **by** *auto*

**qed**

**qed**

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

**where** *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**  $\bigwedge p. \text{Factorial-Ring.prime } p \Longrightarrow k \ \text{dvd} \ (\text{multiplicity } p \ n)$

**shows** *is-nth-power*  $k \ n$

**proof** –

**have**  $n = (\prod p \in \text{prime-factors } n. \ p^{\text{multiplicity } p \ n})$  **using** *assms(1)*

**by** (*simp add: prod-prime-factors*)

**also have**  $\dots = (\prod p \in \text{prime-factors } n. \ p^{\sim((\text{multiplicity } p \ n \ \text{div } k) * k)})$

**by** (*intro prod.cong arg-cong2[where f=power] dvd-div-mult-self[symmetric] refl assms(2)*) *auto*

**also have**  $\dots = (\prod p \in \text{prime-factors } n. \ p^{\sim(\text{multiplicity } p \ n \ \text{div } k)})^{\sim k}$

**unfolding** *power-mult prod-power-distrib[symmetric]* **by** *simp*

**finally have**  $n = (\prod p \in \text{prime-factors } n. \ p^{\sim(\text{multiplicity } p \ n \ \text{div } k)})^{\sim k}$  **by** *simp*

**thus** *?thesis* **by** (*intro is-nth-powerI*) *simp*

**qed**

**lemma** *power-inj-aux*:  
**assumes** *not-perfect-power a not-perfect-power b*  
**assumes**  $n > 0$   $m > n$   
**assumes**  $a^n = b^m$   
**shows** *False*  
**proof** –  
**define** *s* **where**  $s = \text{gcd } n \ m$   
**define** *u* **where**  $u = n \ \text{div} \ \text{gcd } n \ m$   
**define** *t* **where**  $t = m \ \text{div} \ \text{gcd } n \ m$   
  
**have** *a-nz*:  $a \neq 0$  **and** *b-nz*:  $b \neq 0$  **using** *assms(1,2)* **unfolding**  
*not-perfect-power-def* **by** *auto*  
  
**have**  $\text{gcd } n \ m \neq 0$  **using** *assms (3,4)* **by** *simp*  
  
**then obtain** *t u* **where** *n-def*:  $n = t * s$  **and** *m-def*:  $m = u * s$   
**and** *cp*: *coprime t u*  
**using** *gcd-coprime-exists* **unfolding** *s-def t-def u-def* **by** *blast*  
  
**have** *s-gt-0*:  $s > 0$  **and** *t-gt-0*:  $t > 0$  **and** *u-gt-t*:  $u > t$   
**using** *assms(3,4)* **unfolding** *n-def m-def* **by** *auto*  
  
**have**  $(a^t)^s = (b^u)^s$  **using** *assms(5)* **unfolding** *n-def*  
*m-def power-mult* **by** *simp*  
**hence** *0*:  $a^t = b^u$  **using** *s-gt-0* **by** (*metis nth-root-nat-nth-power*)  
  
**have** *u dvd multiplicity p a* **if** *Factorial-Ring.prime p* **for** *p*  
**proof** –  
**have** *prime-elem p* **using** *that* **by** *simp*  
**hence**  $t * \text{multiplicity } p \ a = u * \text{multiplicity } p \ b$   
**using** *0 a-nz b-nz* **by** (*subst (1 2) prime-elem-multiplicity-power-distrib[symmetric]*)  
*auto*  
**hence** *u dvd t \* multiplicity p a* **by** *simp*  
**thus** *?thesis* **using** *cp coprime-commute coprime-dvd-mult-right-iff*  
**by** *blast*  
**qed**  
  
**hence** *is-nth-power u a* **using** *a-nz* **by** (*intro is-nth-power-from-multiplicities*)  
*auto*  
**moreover** **have**  $u > 1$  **using** *u-gt-t t-gt-0* **by** *auto*  
**ultimately show** *False* **using** *assms(1)* **unfolding** *not-perfect-power-def*  
*is-nth-power-def* **by** *auto*  
**qed**

Generalization of *prime-power-inj'*

**lemma** *power-inj*:  
**assumes** *not-perfect-power a not-perfect-power b*  
**assumes**  $n > 0$   $m > 0$

**assumes**  $a \wedge n = b \wedge m$   
**shows**  $a = b \wedge n = m$   
**proof** –  
**consider**  $(a) n < m \mid (b) m < n \mid (c) n = m$  **by** *linarith*  
**thus** *?thesis*  
**proof** (*cases*)  
**case**  $a$  **thus** *?thesis* **using** *assms power-inj-aux* **by** *auto*  
**next**  
**case**  $b$  **thus** *?thesis* **using** *assms power-inj-aux* [*OF* *assms(2,1,4)*  
 $b$ ] **by** *auto*  
**next**  
**case**  $c$  **thus** *?thesis* **using** *assms* **by** (*simp add: power-eq-iff-eq-base*)  
**qed**  
**qed**

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

**assumes**  $n > 1$   
**shows** *not-perfect-power* (*fst* (*split-power*  $n$ ))  
**proof** (*rule ccontr*)  
**obtain**  $b e$  **where** *be-def*:  $(b, e) = \text{split-power } n$  **by** (*metis surj-pair*)  
**have** *n-def*:  $n = b \wedge e$  **and** *e-max*:  $\bigwedge k. e < k \implies \neg \text{is-nth-power } k n$   
**using** *assms split-power-result* [*OF* *be-def*] **by** *auto*

**have** *e-gt-0*:  $e > 0$  **using** *assms* **unfolding** *n-def* **by** (*cases e*) *auto*

**assume**  $\neg \text{not-perfect-power}$  (*fst* (*split-power*  $n$ ))  
**hence**  $\neg \text{not-perfect-power } b$  **unfolding** *be-def* [*symmetric*] **by** *simp*  
**moreover** **have** *b-gt-1*:  $b > 1$  **using** *assms* **unfolding** *n-def*  
**by** (*metis less-one nat-neq-iff nat-power-eq-Suc-0-iff power-0-left*)  
**ultimately obtain**  $k b'$  **where**  $k \neq 1$  **and** *b-def*:  $b = b' \wedge k$   
**unfolding** *not-perfect-power-def* **by** *auto*  
**hence** *k-gt-1*:  $k > 1$  **using** *b-gt-1 nat-neq-iff* **by** *force*  
**have**  $n = b' \wedge (k * e)$  **unfolding** *power-mult n-def b-def* **by** *auto*  
**moreover** **have**  $k * e > e$  **using** *k-gt-1 e-gt-0* **by** *simp*  
**hence**  $\neg \text{is-nth-power}$  ( $k * e$ )  $n$  **using** *e-max* **by** *auto*  
**ultimately show** *False* **unfolding** *is-nth-power-def* **by** *auto*  
**qed**

**lemma** *prime-not-perfect*:

**assumes** *Factorial-Ring.prime*  $p$   
**shows** *not-perfect-power*  $p$   
**proof** –  
**have**  $k=1$  **if**  $p = x \wedge k$  **for**  $x k$  **using** *assms* **unfolding** *that* **by** (*simp*  
 $\text{add:prime-power-iff}$ )  
**thus** *?thesis* **using** *prime-gt-1-nat* [*OF* *assms*] **unfolding** *not-perfect-power-def*  
**by** *auto*  
**qed**

**lemma** *split-power-prime*:

**assumes** *Factorial-Ring.prime*  $p \ n > 0$   
**shows** *split-power*  $(p \hat{n}) = (p, n)$   
**proof** –  
**obtain**  $x \ e$  **where**  $xe:(x, e) = \text{split-power } (p \hat{n})$  **by** (*metis surj-pair*)  
  
**have**  $1 < p \hat{1}$  **using** *prime-gt-1-nat*[*OF assms(1)*] **by** *simp*  
**also have**  $\dots \leq p \hat{n}$  **using** *assms(2)* *prime-gt-0-nat*[*OF assms(1)*]  
**by** (*intro power-increasing*) *auto*  
**finally have**  $0 : p \hat{n} > 1$  **by** *simp*  
  
**have** *not-perfect-power*  $x$   
**using** *split-power-base-not-perfect*[*OF 0*] **unfolding**  $xe$ [*symmetric*]  
**by** *simp*  
**moreover have** *not-perfect-power*  $p$  **by** (*rule prime-not-perfect*[*OF assms(1)*])  
**moreover have**  $1 : p \hat{n} = x \hat{e}$  **using** *split-power-result*[*OF xe*] **by** *simp*  
**moreover have**  $e > 0$  **using** *0 1* **by** (*cases e*) *auto*  
**ultimately have**  $p=x \wedge n = e$  **by** (*intro power-inj assms(2)*)  
**thus ?thesis using**  $xe$  **by** *simp*  
**qed**

**definition** *is-prime-power* ::  $nat \Rightarrow bool$  **where**  
*is-prime-power*  $n = (\exists p \ k. \text{prime } p \wedge k > 0 \wedge n = p \hat{k})$

**lemma** *is-prime-powerI*:  
**assumes** *prime*  $p \ k > 0$   
**shows** *is-prime-power*  $(p \hat{k})$   
**unfolding** *is-prime-power-def* **using** *assms* **by** *auto*

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

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

**lemma** *GF-in-GF-R*:  
**assumes** *is-prime-power*  $n$   
**shows**  $GF \ n \in \text{set-spmf } (GF_R \ n)$   
**proof** –

```

obtain  $p\ k$  where  $n\text{-def}: n = p^{\wedge}k$  and  $p\text{-prime}: \text{prime } p$  and  $k\text{-gt-0}: k > 0$ 
  using assms unfolding is-prime-power-def by blast
  have  $pk\text{-def}: (p,k) = \text{split-power } n$ 
    unfolding  $n\text{-def}$  using split-power-prime[OF p-prime k-gt-0] by
  auto
  let  $?S = \{f. \text{monic-irreducible-poly } (\text{ring-of } (\text{mod-ring } p))\ f \wedge \text{degree } f = k\}$ 

  have  $S\text{-fin}: \text{finite } ?S$  by (intro monic-irred-poly-set-nonempty-finite p-prime k-gt-0)

  have  $\text{find-irreducible-poly } p\ k \in ?S$ 
    using find-irreducible-poly-result[OF p-prime k-gt-0] by auto
    also have  $\dots = \text{set-spmf } (\text{map-spmf } \text{fst } (\text{sample-irreducible-poly } p\ k))$ 
      unfolding sample-irreducible-poly-result[OF p-prime k-gt-0] set-spmf-of-set-finite[OF S-fin]
      by simp
    finally have  $0: \text{find-irreducible-poly } p\ k \in \text{set-spmf}(\text{map-spmf } \text{fst } (\text{sample-irreducible-poly } p\ k))$ 
      by simp

  have  $GF\ n = \text{poly-mod-ring } (\text{mod-ring } p) (\text{find-irreducible-poly } p\ k)$ 
    unfolding  $GF\text{-def } pk\text{-def}$ [symmetric] by (simp del:find-irreducible-poly.simps)
    also have  $\dots \in \text{set-spmf } (\text{map-spmf } \text{fst } (\text{sample-irreducible-poly } p\ k)) \gg (\lambda x. \{\text{poly-mod-ring } (\text{mod-ring } p)\ x\})$ 
      using  $0$  by force
    also have  $\dots = \text{set-spmf } (GF_R\ n)$ 
      unfolding  $GF_R\text{-def } pk\text{-def}$ [symmetric] by (simp add:set-bind-spmf comp-def bind-image)
    finally show  $?thesis$  by simp
qed

lemma galois-field-random-1:
  assumes is-prime-power n
  shows  $\bigwedge \omega. \omega \in \text{set-spmf } (GF_R\ n) \implies \text{enum}_C\ \omega \wedge \text{field}_C\ \omega \wedge \text{order } (\text{ring-of } \omega) = n$ 
    and lossless-spmf (GF_R n)
proof –
  let  $?pred = \lambda \omega. \text{enum}_C\ \omega \wedge \text{field}_C\ \omega \wedge \text{order } (\text{ring-of } \omega) = n$ 

  obtain  $p\ k$  where  $n\text{-def}: n = p^{\wedge}k$  and  $p\text{-prime}: \text{prime } p$  and  $k\text{-gt-0}: k > 0$ 
    using assms unfolding is-prime-power-def by blast
    let  $?r = (\lambda f. \text{poly-mod-ring } (\text{mod-ring } p)\ f)$ 
    let  $?S = \{f. \text{monic-irreducible-poly } (\text{ring-of } (\text{mod-ring } p))\ f \wedge \text{degree } f = k\}$ 

```

**have**  $fc$ :  $field_C$  ( $mod\text{-}ring$   $p$ ) **by** ( $intro$   $mod\text{-}ring\text{-}is\text{-}field\text{-}c$   $p\text{-}prime$ )  
**have**  $ec$ :  $enum_C$  ( $mod\text{-}ring$   $p$ ) **by** ( $intro$   $mod\text{-}ring\text{-}is\text{-}enum\text{-}c$ )

**have**  $S\text{-}fin$ :  $finite$   $?S$  **by** ( $intro$   $monic\text{-}irred\text{-}poly\text{-}set\text{-}nonempty\text{-}finite$   $p\text{-}prime$   $k\text{-}gt\text{-}0$ )  
**have**  $S\text{-}ne$ :  $?S \neq \{\}$  **by** ( $intro$   $monic\text{-}irred\text{-}poly\text{-}set\text{-}nonempty\text{-}finite$   $p\text{-}prime$   $k\text{-}gt\text{-}0$ )

**have**  $pk\text{-}def$ :  $(p,k) = split\text{-}power$   $n$   
**unfolding**  $n\text{-}def$  **using**  $split\text{-}power\text{-}prime$ [ $OF$   $p\text{-}prime$   $k\text{-}gt\text{-}0$ ] **by**  $auto$

**have**  $cond$ :  $?pred$  ( $?r$   $x$ ) **if**  $x \in ?S$  **for**  $x$   
**proof** –  
**have**  $order$  ( $ring\text{-}of$  ( $poly\text{-}mod\text{-}ring$  ( $mod\text{-}ring$   $p$ )  $x$ )) =  $idx\text{-}size$  ( $poly\text{-}mod\text{-}ring$  ( $mod\text{-}ring$   $p$ )  $x$ )  
**using**  $enum\text{-}cD$ [ $OF$   $enum\text{-}c\text{-}poly\text{-}mod\text{-}ring$ [ $OF$   $ec$   $field\text{-}c\text{-}imp\text{-}ring$ [ $OF$   $fc$ ]]] **by**  $simp$   
**also** **have**  $\dots = p^{\wedge}(\mathit{degree}$   $x$ )  
**by** ( $simp$   $add$ : $poly\text{-}mod\text{-}ring\text{-}def$   $Finite\text{-}Fields\text{-}Mod\text{-}Ring\text{-}Code.mod\text{-}ring\text{-}def$ )  
**also** **have**  $\dots = n$  **unfolding**  $n\text{-}def$  **using**  $that$  **by**  $simp$   
**finally** **have**  $order$  ( $ring\text{-}of$  ( $poly\text{-}mod\text{-}ring$  ( $mod\text{-}ring$   $p$ )  $x$ )) =  $n$   
**by**  $simp$

**thus**  $?thesis$  **using**  $that$   
**by** ( $intro$   $conjI$   $enum\text{-}c\text{-}poly\text{-}mod\text{-}ring$   $field\text{-}c\text{-}poly\text{-}mod\text{-}ring$   $ec$   $field\text{-}c\text{-}imp\text{-}ring$   $fc$ )  $auto$   
**qed**

**have**  $GF_R$   $n = bind\text{-}spmf$  ( $map\text{-}spmf$   $fst$  ( $sample\text{-}irreducible\text{-}poly$   $p$   $k$ )) ( $\lambda x. return\text{-}spmf$  ( $?r$   $x$ ))  
**unfolding**  $GF_R\text{-}def$   $pk\text{-}def$ [ $symmetric$ ]  $map\text{-}spmf\text{-}conv\text{-}bind\text{-}spmf$   
**by**  $simp$   
**also** **have**  $\dots = spmf\text{-}of\text{-}set$   $?S \gg= (\lambda f. return\text{-}spmf$  ( $(?r$   $f$ )))  
**unfolding**  $sample\text{-}irreducible\text{-}poly\text{-}result$ [ $OF$   $p\text{-}prime$   $k\text{-}gt\text{-}0$ ] **by** ( $simp$ )  
**also** **have**  $\dots = pmf\text{-}of\text{-}set$   $?S \gg= (\lambda f. return\text{-}spmf$  ( $?r$   $f$ ))  
**unfolding**  $spmf\text{-}of\text{-}pmf\text{-}pmf\text{-}of\text{-}set$ [ $OF$   $S\text{-}fin$   $S\text{-}ne$ ,  $symmetric$ ]  $spmf\text{-}of\text{-}pmf\text{-}def$   
**by** ( $simp$   $add$ : $bind\text{-}spmf\text{-}def$   $bind\text{-}map\text{-}pmf$ )  
**finally** **have**  $0:GF_R$   $n = map\text{-}pmf$  ( $Some \circ ?r$ ) ( $pmf\text{-}of\text{-}set$   $?S$ ) **by** ( $simp$   $add$ : $comp\text{-}def$   $map\text{-}pmf\text{-}def$ )

**show**  $enum_C$   $\omega \wedge field_C$   $\omega \wedge order$  ( $ring\text{-}of$   $\omega$ ) =  $n$  **if**  $\omega \in set\text{-}spmf$  ( $GF_R$   $n$ ) **for**  $\omega$   
**proof** –  
**have**  $Some$   $\omega \in set\text{-}pmf$  ( $GF_R$   $n$ ) **unfolding**  $in\text{-}set\text{-}spmf$ [ $symmetric$ ]  
**by** ( $rule$   $that$ )  
**also** **have**  $\dots = (Some \circ ?r)$  ‘  $?S$  **unfolding**  $0$   $set\text{-}map\text{-}pmf$   $set\text{-}pmf\text{-}of\text{-}set$ [ $OF$   $S\text{-}ne$   $S\text{-}fin$ ] **by**  $simp$

**finally have**  $\text{Some } \omega \in (\text{Some} \circ ?r) \text{ ' } ?S$  **by simp**  
**hence**  $\omega \in ?r \text{ ' } ?S$  **by auto**  
**then obtain**  $x$  **where**  $x : x \in ?S$  **and**  $\omega\text{-def} : \omega = ?r \ x$  **by auto**  
**show**  $?thesis$  **unfolding**  $\omega\text{-def}$  **by**  $(\text{intro cond } x)$   
**qed**

**have**  $\text{None} \notin \text{set-pmf}(GF_R \ n)$  **unfolding**  $0$   $\text{set-map-pmf set-pmf-of-set}[OF$   
 $S\text{-ne } S\text{-fin}]$  **by auto**  
**thus**  $\text{lossless-spmf}(GF_R \ n)$  **using**  $\text{lossless-iff-set-pmf-None}$  **by blast**  
**qed**

**lemma** *galois-field*:

**assumes**  $\text{is-prime-power } n$   
**shows**  $\text{enum}_C(GF \ n) \ \text{field}_C(GF \ n) \ \text{order}(\text{ring-of}(GF \ n)) = n$   
**using**  $\text{galois-field-random-1}(1)[OF \ \text{assms}(1) \ GF\text{-in-}GF\text{-R}[OF \ \text{assms}(1)]]$   
**by auto**

**lemma** *lossless-imp-spmf-of-pmf*:

**assumes**  $\text{lossless-spmf } M$   
**shows**  $\text{spmof-of-pmf}(\text{map-pmf the } M) = M$   
**proof** –  
**have**  $\text{spmof-of-pmf}(\text{map-pmf the } M) = \text{map-pmf}(\text{Some} \circ \text{the}) \ M$   
**unfolding**  $\text{spmof-of-pmf-def}$  **by**  $(\text{simp add: pmf.map-comp})$   
**also have**  $\dots = \text{map-pmf id } M$   
**using**  $\text{assms}$  **unfolding**  $\text{lossless-iff-set-pmf-None}$   
**by**  $(\text{intro map-pmf-cong refl})$   $(\text{metis id-apply o-apply option.collapse})$   
**also have**  $\dots = M$  **by simp**  
**finally show**  $?thesis$  **by simp**  
**qed**

**lemma** *galois-field-random-2*:

**assumes**  $\text{is-prime-power } n$   
**shows**  $\text{map-spmf}(\lambda \omega. \text{enum}_C \ \omega \wedge \text{field}_C \ \omega \wedge \text{order}(\text{ring-of } \omega) = n) (GF_R \ n) = \text{return-spmf True}$   
 $(\text{is } ?L = -)$   
**proof** –  
**have**  $?L = \text{map-spmf}(\lambda \omega. \text{True}) (GF_R \ n)$   
**using**  $\text{galois-field-random-1}[OF \ \text{assms}]$  **by**  $(\text{intro map-spmf-cong refl})$  **auto**  
**also have**  $\dots = \text{map-pmf}(\lambda \omega. \text{Some True}) (GF_R \ n)$   
**by**  $(\text{subst lossless-imp-spmf-of-pmf}[OF \ \text{galois-field-random-1}(2)[OF \ \text{assms}], \text{symmetric}])$  **simp**  
**also have**  $\dots = \text{return-spmf True}$  **unfolding**  $\text{map-pmf-def}$  **by simp**  
**finally show**  $?thesis$  **by simp**  
**qed**

**lemma** *bind-galois-field-cong*:

**assumes**  $\text{is-prime-power } n$   
**assumes**  $\bigwedge \omega. \text{enum}_C \ \omega \implies \text{field}_C \ \omega \implies \text{order}(\text{ring-of } \omega) = n \implies$

```

f ω = g ω
shows bind-spmf (GFR n) f = bind-spmf (GFR n) g
using galois-field-random-1(1)[OF assms(1)]
by (intro bind-spmf-cong refl assms(2)) auto

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](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.