

# Hilbert Basis Theorems\*

Benjamin Puyobro

Benoît Ballenghien

Burkhart Wolff

Université Paris Saclay, IRT SystemX, LMF, CNRS

March 17, 2025

## Contents

<b>1</b>	<b>A Proof of Hilbert Basis Theorems and an Extension to Formal Power Series</b>	<b>2</b>
<b>2</b>	<b>Ring Miscellaneous</b>	<b>3</b>
<b>3</b>	<b>Polynomials Ring Miscellaneous</b>	<b>9</b>
<b>4</b>	<b>The weak Hilbert Basis theorem</b>	<b>15</b>
4.1	Weak Hilbert Basis . . . . .	15
4.2	Some properties of noetherian rings . . . . .	23
4.3	Some properties of the polynomial rings regarding ideals and quotients . . . . .	27
4.4	The isomorphisms between the different models of polynomials	36
<b>5</b>	<b>The Hilbert Basis theorem for Indexed Polynomials Rings</b>	<b>47</b>
5.1	The isomorphism between $A[X_0..X_n]$ and $A[X_0..X_{n-1}][X_n]$ .	48
5.2	Preliminaries lemmas . . . . .	51
5.3	Hilbert Basis theorem . . . . .	52
<b>6</b>	<b>The Hilbert Basis theorem for Formal Power Series</b>	<b>54</b>
6.1	Preliminaries definition and lemmas . . . . .	54
6.2	Premisses for noetherian ring proof . . . . .	58
6.3	The Hilbert Basis theorem . . . . .	76

---

\*This work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute within the CVH project.

<b>7</b>	<b>The Real Ring definition</b>	<b>92</b>
<b>8</b>	<b>Examples</b>	<b>93</b>
8.1	Examples of noetherian rings with $\mathbb{Z}$ and $\mathbb{Z}[X]$ . . . . .	93
8.2	Another example with $\mathbb{R}$ and $\mathbb{R}[X]$ . . . . .	95

# 1 A Proof of Hilbert Basis Theorems and an Extension to Formal Power Series

The Hilbert Basis Theorem is enlisted in the extension of Wiedijk's catalogue "Formalizing 100 Theorems" [4], a well-known collection of challenge problems for the formalisation of mathematics.

In this paper, we present a formal proof of several versions of this theorem in Isabelle/HOL. Hilbert's basis theorem asserts that every ideal of a polynomial ring over a commutative ring has a finite generating family (a finite basis in Hilbert's terminology). A prominent alternative formulation is: every polynomial ring over a Noetherian ring is also Noetherian.

In more detail, the statement and our generalization can be presented as follows:

- **Hilbert's Basis Theorem.** Let  $\mathfrak{R}[X]$  denote the ring of polynomials in the indeterminate  $X$  over the commutative ring  $\mathfrak{R}$ . Then  $\mathfrak{R}[X]$  is Noetherian iff  $\mathfrak{R}$  is.
- **Corollary.**  $\mathfrak{R}[X_1, \dots, X_n]$  is Noetherian iff  $\mathfrak{R}$  is.
- **Extension.** If  $\mathfrak{R}$  is a Noetherian ring, then  $\mathfrak{R}[[X]]$  is a Noetherian ring, where  $\mathfrak{R}[[X]]$  denotes the formal power series over the ring  $\mathfrak{R}$ .

We also provide isomorphisms between the three types of polynomial rings defined in HOL-Algebra. Together with the fact that the noetherian property is preserved by isomorphism, we get Hilbert's Basis theorem for all three models. We believe that this technique has a wider potential of applications in the AFP library.

## 2 Ring Miscellaneous

**theory** *Ring-Misc*

**imports**

*HOL-Algebra.RingHom*  
*HOL-Algebra.QuotRing*  
*HOL-Algebra.Embedded-Algebras*

**begin**

Some lemmas that may be considered as useful, and that helps for the Hilbert's basis proof

**lemma** (in *ring*) *carrier-quot*:  $\langle \text{ideal } I \ R \implies \text{carrier } (R \ \text{Quot } I) = \{\{y \oplus x \mid y. y \in I\} \mid x. x \in \text{carrier } R\} \rangle$

**proof**(*safe*)

**fix** *x*

**assume** *h*: $\langle \text{ideal } I \ R \rangle \langle x \in \text{carrier } (R \ \text{Quot } I) \rangle$

**then have**  $\langle \exists xa \in \text{carrier } R. x = (\bigcup x \in I. \{x \oplus xa\}) \rangle$

**unfolding** *FactRing-def A-RCOSETS-def RCOSETS-def cgenideal-def r-coset-def*

**by**(*simp*)

**then obtain** *y* **where**  $\langle x = (\bigcup x \in I. \{x \oplus y\}) \wedge y \in \text{carrier } R \rangle$  **by** *blast*

**with** *h* **show**  $\langle \exists xa. x = \{y \oplus xa \mid y. y \in I\} \wedge xa \in \text{carrier } R \rangle$

**by**(*blast*)

**next**

**fix** *x xa*

**assume**  $\langle \text{ideal } I \ R \rangle \langle xa \in \text{carrier } R \rangle$

**then show**  $\langle \{y \oplus xa \mid y. y \in I\} \in \text{carrier } (R \ \text{Quot } I) \rangle$

**unfolding** *FactRing-def A-RCOSETS-def RCOSETS-def cgenideal-def r-coset-def*

**apply** *simp*

**apply**(*rule* *bexI*[**where** *x=xa*])

**by** *auto*

**qed**

**context**

**fixes** *A B h*

**assumes** *ring-A*:  $\langle \text{ring } A \rangle$

**assumes** *ring-B*:  $\langle \text{ring } B \rangle$

**assumes** *h1*:  $\langle h \in \text{ring-iso } A \ B \rangle$

**begin**

**interpretation** *ringA*: *ring A*

**using** *ring-A* **by** *auto*

**interpretation** *ringB*: *ring B*

**using** *ring-B* **by** *auto*

**interpretation** *rho*: *ring-hom-ring A B h*

**apply**(*unfold-locales*)

```

using h1 unfolding ring-iso-def by auto

lemma inv-img-exist:⟨∀ xa∈carrier B. ∃y. y ∈ carrier A ∧ h y = xa⟩
using h1 bij-betw-iff-bijections[of h ⟨carrier A⟩ ⟨carrier B⟩] unfolding ring-iso-def
by(auto)

lemma img-ideal-is-ideal:assumes j1:⟨ideal I A⟩
shows ⟨ideal (h ‘ I) B⟩
proof(intro idealI)
show ⟨ring B⟩
by(simp add: ringB.ring-axioms)
from j1 show ⟨subgroup (h ‘ I) (add-monoid B)⟩
by (metis (no-types, lifting) additive-subgroup-def ideal-def rhr.img-is-add-subgroup)
fix a x
assume hyp:⟨a ∈ h ‘ I⟩ ⟨x ∈ carrier B⟩
with j1 show fst:⟨x ⊗B a ∈ h ‘ I⟩
by (smt (verit, ccfv-threshold) inv-img-exist h1 ideal.I-l-closed ideal.Icarr image-iff
ring-iso-memE(2))
from j1 show ⟨a ⊗B x ∈ h ‘ I⟩
using inv-img-exist fst hyp(2)
by (smt (verit, best) hyp(1) ideal.I-r-closed ideal.Icarr image-iff rhr.hom-mult)
qed

lemma img-in-carrier-quot:⟨∀ x∈ carrier (A Quot I). h ‘ x ∈ carrier (B Quot
(h‘I))⟩ if j:⟨ideal I A⟩ for I
proof(subst ringA.carrier-quot(1)[OF j],subst ringB.carrier-quot[of ⟨h‘I⟩], safe)
show ⟨ideal (h ‘ I) B⟩
using img-ideal-is-ideal that by blast
next
fix x xa
assume h:⟨xa ∈ carrier A⟩
then show ⟨∃x. h ‘ {y ⊕A xa |y. y ∈ I} = {y ⊕B x |y. y ∈ h ‘ I} ∧ x ∈ carrier
B⟩
apply(intro exI[where x=⟨h xa⟩])
apply(safe)
using h1 j ideal.Icarr ring-iso-memE(3) that apply fastforce
using h1 ideal.Icarr image-iff mem-Collect-eq ring-iso-memE(3) that apply
fastforce
by (meson h1 ring-iso-memE(1))
qed

lemma f8:⟨xa∈carrier B ∧ xb∈I ⟹h(xb ⊕A inv-into (carrier A) h xa) = h xb
⊕B xa⟩ if j:⟨ideal I A⟩ for I xb xa
proof –
assume xa ∈ carrier B ∧ xb ∈ I
then show ?thesis
using inv-img-exist f-inv-into-f[of xa h ⟨carrier A⟩] ideal.Icarr[OF that, of xb]
inv-into-into[of xa h]

```

by(*auto*)  
qed

**lemma** *f9*: $\langle \forall xa \in \text{carrier } B. \forall xb \in \text{carrier } A. \exists y. h y = h xb \oplus_B xa \rangle$   
using *f8 ringA.oneideal* by *blast*

**lemma** *img-over-set-is-iso*:  $\langle \text{ideal } I A \implies ((\cdot) h) \in \text{ring-iso } (A \text{ Quot } I) (B \text{ Quot } (h'I)) \rangle$  for *I*

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

**fix** *x*  
**assume** *k*: $\langle \text{ideal } I A \rangle \langle x \in \text{carrier } (A \text{ Quot } I) \rangle$   
**then show**  $\langle h ' x \in \text{carrier } (B \text{ Quot } h ' I) \rangle$   
 using *h1 ringA.ring-axioms ringB.ring-axioms*  
 by(*simp add:img-in-carrier-quot*)  
**fix** *y*  
 {  
**fix** *xa xb xc*  
**assume** *g*: $\langle xa \in x \rangle \langle xb \in y \rangle \langle xc \in I \rangle \langle \text{ideal } I A \rangle \langle x \in \text{a-rcosets}_A I \rangle \langle y \in \text{a-rcosets}_A I \rangle$   
**have** *xa*: $\langle xa \in \text{carrier } A \rangle$   
 using *abelian-subgroup.a-rcosets-carrier abelian-subgroupI3 g(1) g(5)*  
*ideal-def k(1) ring-def* by *blast*  
**have** *xb*: $\langle xb \in \text{carrier } A \rangle$   
 using *abelian-subgroup.a-rcosets-carrier abelian-subgroupI3 g(2) g(6)*  
*ideal-def k(1) ring-def* by *blast*  
**have** *xc*: $\langle xc \in \text{carrier } A \rangle$   
 using *g(3) k(1) ringA.ideal-is-subalgebra ringA.subalgebra-in-carrier* by *fast-force*  
**have**  $\langle \exists x \in x. \exists xd \in y. \exists xe \in I. h (xc \oplus_A xa \otimes_A xb) = h xe \oplus_B h x \otimes_B h xd \rangle$   
**apply**(*rule beXI[where x=xa]*)  
**apply**(*rule beXI[where x=xb]*)  
**apply**(*rule beXI[where x=xc]*)  
 using *g rhr.hom-add[OF xc] rhr.hom-mult[OF xa xb]*  
 using *ringA.m-closed xa xb* by *presburger+*  
 }  
**note** *fst-prf=this*  
 {**fix** *xa xb xc*  
**assume** *g*: $\langle xa \in x \rangle \langle xb \in y \rangle \langle xc \in I \rangle \langle \text{ideal } I A \rangle \langle x \in \text{a-rcosets}_A I \rangle \langle y \in \text{a-rcosets}_A I \rangle$   
**have** *xa*: $\langle xa \in \text{carrier } A \rangle$   
 using *abelian-subgroup.a-rcosets-carrier abelian-subgroupI3 g(1) g(5)*  
*ideal-def k(1) ring-def* by *blast*  
**have** *xb*: $\langle xb \in \text{carrier } A \rangle$   
 using *abelian-subgroup.a-rcosets-carrier abelian-subgroupI3 g(2) g(6)*  
*ideal-def k(1) ring-def* by *blast*  
**have** *xc*: $\langle xc \in \text{carrier } A \rangle$   
 using *g(3) k(1) ringA.ideal-is-subalgebra ringA.subalgebra-in-carrier* by *fast-force*  
**have**  $\langle \exists ya \in x. \exists y \in y. \exists yb \in I. h xc \oplus_B h xa \otimes_B h yb = h (yb \oplus_A ya \otimes_A y) \rangle$   
**apply**(*rule beXI[where x=xa]*)

```

    apply(rule beXI[where x=xb])
    apply(rule beXI[where x=xc])
    using g rhr.hom-add[OF xc] rhr.hom-mult[OF xa xb]
    using ringA.m-closed xa xb by presburger+ }note snd-prf=this
  assume k1:⟨y ∈ carrier (A Quot I)⟩
  with k show ⟨h ‘ (x ⊗A Quot I y) = h ‘ x ⊗B Quot h ‘ I h ‘ y⟩
    by(auto simp:FactRing-def image-iff rcoset-mult-def r-coset-def a-r-coset-def
    snd-prf fst-prf)
  from k k1 show ⟨h ‘ (x ⊕A Quot I y) = h ‘ x ⊕B Quot h ‘ I h ‘ y⟩
    apply(simp add:FactRing-def rcoset-mult-def r-coset-def a-r-coset-def)
    using h1 ring-A ring-B unfolding ring-iso-def FactRing-def rcoset-mult-def
    r-coset-def a-r-coset-def
    by (metis (no-types, lifting) abelian-subgroup.a-rcosets-carrier abelian-subgroupI3
    ideal.axioms(1) mem-Collect-eq ring-def set-add-hom)
next
  assume k:⟨ideal I A⟩
  have important:⟨xa ∈ carrier (B Quot h ‘ I) ⟹ ∃ y∈carrier (A Quot I). h ‘ y
  = xa⟩ for xa
  proof(rule beXI[where x=⟨inv-into (carrier A) h ‘ xa⟩])
    assume g:⟨xa ∈ carrier (B Quot h ‘ I)⟩
    then show ⟨h ‘ inv-into (carrier A) h ‘ xa = xa⟩
      by (metis Sup-le-iff bij-betw-def img-ideal-is-ideal h1 image-inv-into-cancel k
      ringB.canonical-proj-vimage-in-carrier ring-iso-memE(5) subset-refl)
    {fix x
      assume g1:⟨x∈carrier B⟩ ⟨ xa = (⋃ xa∈I. {h xa ⊕B x})⟩
      {fix xaa
        assume g2:⟨xaa ∈ I⟩
        with g1 have ⟨∃ xa∈I. (SOME y. y ∈ carrier A ∧ h y = h xaa ⊕B x) = xa
        ⊕A inv-into (carrier A) h x⟩
          by (smt (verit, del-Insts) bij-betw-def bij-betw-iff-bijections h1 ideal.Icarr
          inv-into-f-f k rhr.hom-add ringA.add.m-closed ring-iso-memE(5)
          some-equality)
          }note 2=this
        {fix xaa
          assume ⟨xaa∈I⟩
          with g1 have ⟨xaa ⊕A inv-into (carrier A) h x = (SOME y. y ∈ carrier A
          ∧ h y = h xaa ⊕B x)⟩
            using h1 ring-A ring-B unfolding ring-iso-def
            by (smt (verit, del-Insts) bij-betw-def k inv-img-exist f8 h1 ideal.Icarr
            inv-into-f-f mem-Collect-eq ringA.add.m-closed someI-ex)
          }note 3=this
        from g1 have ⟨∃ xa∈carrier A. (λx. SOME y. y ∈ carrier A ∧ h y = x) ‘
        (⋃ xa∈I. {h xa ⊕B x}) = (⋃ x∈I. {x ⊕A xa})⟩
          apply(intro beXI[where x=⟨inv-into (carrier A) h x⟩])
          using inv-img-exist image-eqI inv-into-into[of x h ⟨carrier A⟩]
          by(auto simp: 2 3)
        }note 1 =this
      from g show ⟨inv-into (carrier A) h ‘ xa ∈ carrier (A Quot I)⟩
      unfolding FactRing-def inv-into-def A-RCOSETS-def RCOSETS-def r-coset-def

```

```

by(auto simp:1)
qed
have imp2:⟨∀ J⊆carrier A. ∀ K⊆carrier A. h ‘ J = h ‘ K ⟶ J = K⟩
  unfolding image-def using h1 apply(safe)
  using h1 ring-A ring-B unfolding ring-iso-def
  by (smt (verit, ccfv-SIG) bij-betw-iff-bijections in-mono mem-Collect-eq) +
with important have important3:⟨xa ∈ carrier (B Quot h ‘ I)
⟹ ∃!y∈carrier (A Quot I). h ‘ y = xa⟩ for xa
  apply(safe)
  apply blast
  apply (metis Sup-le-iff equalityE k ringA.canonical-proj-vimage-in-carrier)
  by (metis Sup-le-iff dual-order.refl k ringA.canonical-proj-vimage-in-carrier)
have bij-inv:⟨bij-betw (inv-into (carrier A) h) (carrier B) (carrier A)⟩
  by (simp add: bij-betw-inv-into h1 ring-iso-memE(5))
with k show ⟨h ‘ 1A Quot I = 1B Quot h ‘ I⟩
  apply(auto simp:image-def FactRing-def rcoset-mult-def r-coset-def a-r-coset-def)
[1]
  apply (smt (verit, ccfv-threshold) h1 ideal.Icarr insert-iff ringA.one-closed
ring-iso-memE(3) ring-iso-memE(4))
  by (metis (full-types) h1 ideal.Icarr ringA.one-closed ring-iso-memE(3) ring-iso-memE(4)
singletonI)
show ⟨bij-betw ((‘) h) (carrier (A Quot I)) (carrier (B Quot h ‘ I))⟩
proof(intro bij-betw-byWitness[where ?f' = (‘) (inv-into (carrier A) h)])
  from k show ⟨∀ a∈carrier (A Quot I). inv-into (carrier A) h ‘ h ‘ a = a⟩
  apply(intro ballI)
  apply(subst inv-into-image-cancel)
  using bij-betw-def h1 ring-A ring-B unfolding ring-iso-def apply blast
  apply (metis FactRing-def abelian-subgroup.a-rcosets-carrier
abelian-subgroupI3 ideal-def partial-object.select-convs(1) ring-def)
  by(simp)
  from k show ⟨∀ a'∈carrier (B Quot h ‘ I). h ‘ inv-into (carrier A) h ‘ a' = a'⟩
  using ring-A ring-B h1 unfolding ring-iso-def
  by (metis (no-types, lifting) Sup-le-iff bij-betw-def img-ideal-is-ideal im-
age-inv-into-cancel
mem-Collect-eq ringB.canonical-proj-vimage-in-carrier subset-refl)
  from k show ⟨(‘) h ‘ carrier (A Quot I) ⊆ carrier (B Quot h ‘ I)⟩
  using img-in-carrier-quot by blast
  from k show ⟨(‘) (inv-into (carrier A) h) ‘ carrier (B Quot h ‘ I) ⊆ carrier
(A Quot I)⟩
  apply(subst (1) image-def)
  apply(safe)
  by (metis ⟨∀ a∈carrier (A Quot I). inv-into (carrier A) h ‘ h ‘ a = a⟩
important3)
qed
qed
end

```

**lemma** *Quot-iso-cgen*:⟨a∈carrier A ∧ b:carrier B ∧ cring A ∧ cring B ∧ h ∈ ring-iso A B ∧ h(a) = b



```

⇒ A Quot (cgenideal A a) ≃ B Quot (cgenideal B b)
  unfolding is-ring-iso-def ring-iso-def
  proof(subst ex-in-conv[symmetric])
    assume h1:⟨a∈carrier A ∧ b:carrier B ∧ cring A ∧ cring B ∧ h ∈ {h ∈ ring-hom
A B. bij-betw h (carrier A) (carrier B)} ∧ h a = b⟩
    have h1':⟨h ∈ ring-iso A B⟩
      using h1 apply(fold ring-iso-def) by simp
    interpret ringA: cring A
      using h1 by auto
    interpret ringB: cring B
      using h1 by simp
    have f1:⟨∀ xa∈carrier B. ∃ y. y ∈ carrier A ∧ h y = xa⟩
      by (metis (no-types, lifting) bij-betw-iff-bijections h1 mem-Collect-eq)
    have f0:⟨ideal (PIdl_A a) A ∧ ideal (PIdl_B b) B⟩
      using ringA.cgenideal-ideal[of a] ringB.cgenideal-ideal[of b] h1 by(simp)
    then have f2:⟨(carrier (A Quot PIdl_A a)) = {{y⊕_A x | y. y∈PIdl_A a} | x.
x∈carrier A}
  › ⟨(carrier (B Quot PIdl_B b)) = {{y⊕_B x | y. y∈PIdl_B b} | x. x∈carrier B}⟩
      using ringA.carrier-quot ringB.carrier-quot by simp+
    then have ⟨h'(PIdl_A a) = (PIdl_B b)⟩
      unfolding image-def cgenideal-def
    proof(safe)
      fix x xa xb
      assume h2:⟨carrier (A Quot {x ⊗_A a | x. x ∈ carrier A}) = {{y ⊕_A x | y. y
∈ {x ⊗_A a | x. x ∈ carrier A}} | x. x ∈ carrier A}⟩
        ⟨carrier (B Quot {x ⊗_B b | x. x ∈ carrier B}) = {{y ⊕_B x | y. y ∈ {x ⊗_B b
|x. x ∈ carrier B}} | x. x ∈ carrier B}⟩
        ⟨xb ∈ carrier A⟩
      then show ⟨∃ x. h (xb ⊗_A a) = x ⊗_B b ∧ x ∈ carrier B⟩
        using h1 ring-iso-def ring-iso-memE(1) ring-iso-memE(2) by fastforce
    next
      fix x xa
      assume h2:⟨carrier (A Quot {x ⊗_A a | x. x ∈ carrier A}) = {{y ⊕_A x | y. y
∈ {x ⊗_A a | x. x ∈ carrier A}} | x. x ∈ carrier A}⟩
        ⟨carrier (B Quot {x ⊗_B b | x. x ∈ carrier B}) = {{y ⊕_B x | y. y ∈ {x ⊗_B b
|x. x ∈ carrier B}} | x. x ∈ carrier B}⟩
        ⟨xa ∈ carrier B⟩
      show ⟨∃ x∈{x ⊗_A a | x. x ∈ carrier A}. xa ⊗_B b = h x ⟩
        using f1 h1 h1' h2(3) ring-iso-memE(2) by fastforce
    qed
  then have ⟨∀ x∈(PIdl_B b). ∃!y∈(PIdl_A a). h y = x⟩
    by (smt (verit) bij-betw-iff-bijections f0 h1 ideal.Icarr image-def mem-Collect-eq)
  then have ⟨x∈carrier (A Quot {x ⊗_A a | x. x ∈ carrier A}) ⇒ ∃ y'∈carrier A.
x = {y⊕_A y' | y. y∈PIdl_A a}⟩ for x
  proof -
    assume a1: x ∈ carrier (A Quot {x ⊗_A a | x. x ∈ carrier A})
    have f2: ∀ Aa Ab. Ab ∉ carrier (A Quot Aa) ∨ ¬ ideal Aa A
      ∨ (∃ a. Ab = {aa ⊕_A a | aa. aa ∈ Aa} ∧ a ∈ carrier A)
    using ringA.carrier-quot by auto

```

```

have  $x \in \text{carrier } (A \text{ Quot } \text{PIdl}_A \ a)$ 
  using  $a1$  by (simp add: cgenideal-def)
then show ?thesis
  using  $f2 \ f0$  by blast
qed
show  $\langle \exists x. x \in \{h \in \text{ring-hom } (A \text{ Quot } \text{PIdl}_A \ a) \ (B \text{ Quot } \text{PIdl}_B \ b). \text{bij-betw } h \ (\text{carrier } (A \text{ Quot } \text{PIdl}_A \ a)) \ (\text{carrier } (B \text{ Quot } \text{PIdl}_B \ b))\} \rangle$ 
  apply (fold ring-iso-def)
  apply (intro exI[where  $x = \langle \lambda x. h \ 'x \rangle$ ])
  using  $\langle h \ ' ( \text{PIdl}_A \ a) = \text{PIdl}_B \ b \rangle \ f0 \ h1 \ ' \text{img-over-set-is-iso } \text{ringA.ring-axioms}$ 
ringB.ring-axioms
  by force
qed

```

**end**

### 3 Polynomials Ring Miscellaneous

**theory** *Polynomials-Ring-Misc*

**imports** *HOL-Algebra.Polynomials*

**begin**

Some lemmas that may be considered as useful, and that helps for the Hilbert's basis proof

**definition** (**in** *ring*) *deg-poly-set*:  $\langle \text{deg-poly-set } S \ k = \{a. a \in S \wedge \text{degree } a = k\} \cup \{\emptyset\} \rangle$

**definition** (**in** *ring*) *lead-coeff-set*:  $\langle 'a \text{ list set} \Rightarrow \text{nat} \Rightarrow 'a \text{ set} \rangle$   
**where**  $\langle \text{lead-coeff-set } S \ k \equiv \{\text{coeff } a \ (\text{degree } a) \mid a. a \in \text{deg-poly-set } S \ k\} \rangle$

**lemma** *rule-union*:  $\langle x \in (\bigcup n \leq k. A \ l \ n) \longleftrightarrow (\exists n \leq k. x \in A \ l \ n) \rangle$   
**by** (*auto*)

**lemma** (**in** *ring*) *add-0-eq-0-is-0*:  $\langle a \in \text{carrier } ((\text{carrier } R)[X]) \Longrightarrow \emptyset \oplus_{(\text{carrier } R)} [X] \ a = \emptyset \implies a = \emptyset \rangle$

**proof** –

```

assume  $h1: \langle a \in \text{carrier } ((\text{carrier } R)[X]) \rangle$  and  $h2: \langle \emptyset \oplus_{(\text{carrier } R)} [X] \ a = \emptyset \rangle$ 
have  $\langle \text{poly-add } \emptyset \ a = a \rangle$ 
  apply (rule local.poly-add-zero(2)[of  $\langle (\text{carrier } R) \rangle$ ])
  apply (simp add: carrier-is-subring)
  by (simp add: h1 univ-poly-carrier)
then show ?thesis
  using  $h2$  unfolding univ-poly-add by presburger
qed

```

**lemma** (in domain) *inv-coeff-sum*:  $\langle a \in \text{carrier}((\text{carrier } R)[X]) \implies aa \in \text{carrier}((\text{carrier } R)[X]) \rangle$   
 $\implies a \oplus_{(\text{carrier } R)[X]} aa = [] \iff (\forall n. \text{coeff } a \ n = \text{inv}_{\text{add-monoid } R} (\text{coeff } aa \ n)) \rangle$   
**proof** (safe, induct a)  
  case Nil  
  then have  $\langle aa = [] \rangle$   
  by (simp add: Nil.premis(2) Nil.premis(3) add-0-eq-0-is-0)  
  then show ?case by (auto)  
next  
  case (Cons a1 a2)  
  then show ?case  
  by (metis add.comm-inv-char coeff.simps(1) coeff-in-carrier local.add.m-comm local.ring-axioms  
  poly-add-coeff polynomial-in-carrier ring.carrier-is-subring univ-poly-add univ-poly-carrier)  
next  
  interpret *kxr*: *cring* (carrier R)[X]  
  using carrier-is-subring univ-poly-is-cring by blast  
  assume *h1*:  $\langle a \in \text{carrier}((\text{carrier } R)[X]) \rangle$  and *h2*:  $\langle aa \in \text{carrier}((\text{carrier } R)[X]) \rangle$   
  and *h3*:  $\langle \forall n. \text{local.coeff } a \ n = \text{inv}_{\text{add-monoid } R} \text{local.coeff } aa \ n \rangle$   
  then show  $\langle a \oplus_{(\text{carrier } R)[X]} aa = [] \rangle$   
  by (metis (no-types, lifting) abelian-group-def abelian-monoid.a-monoid add.Units-eq carrier-is-subring coeff-in-carrier kxr.add.m-closed kxr.add.m-comm lead-coeff-simp local.ring-axioms  
  mem-Collect-eq monoid.Units-r-inv monoid.select-convs(1) monoid.select-convs(2) partial-object.select-convs(1)  
  poly-add-coeff polynomial-def polynomial-in-carrier ring-def univ-poly-add univ-poly-def)  
**qed**

**lemma** (in ring) *coeffs-of-add-poly*:  $\langle a \in \text{carrier}((\text{carrier } R)[X]) \implies aa \in \text{carrier}((\text{carrier } R)[X]) \rangle$   
 $\implies \text{coeff } (a \oplus_{(\text{carrier } R)[X]} aa) \ n = \text{coeff } a \ n \oplus \text{coeff } aa \ n \rangle$   
  by (metis local.ring-axioms poly-add-coeff ring.polynomial-incl univ-poly-add univ-poly-carrier)

**lemma** (in ring) *length-add*:  $\langle a \in \text{carrier}((\text{carrier } R)[X]) \implies aa \in \text{carrier}((\text{carrier } R)[X]) \rangle$   
 $\implies \text{coeff } a \ (\text{degree } a) \neq \text{inv}_{\text{add-monoid } R} \text{coeff } aa \ (\text{degree } aa)$   
 $\implies \text{degree } (a \oplus_{(\text{carrier } R)[X]} aa) = \max (\text{degree } a) (\text{degree } aa) \rangle$   
**proof** –  
  assume *h1*:  $\langle a \in \text{carrier}((\text{carrier } R)[X]) \rangle$   
  and *h2*:  $\langle aa \in \text{carrier}((\text{carrier } R)[X]) \rangle$   
  and *h3*:  $\langle \text{coeff } a \ (\text{degree } a) \neq \text{inv}_{\text{add-monoid } R} \text{coeff } aa \ (\text{degree } aa) \rangle$   
  have *f0*:  $\langle \forall n > (\max (\text{degree } a) (\text{degree } aa)). \text{coeff } (a \oplus_{(\text{carrier } R)[X]} aa) \ n = \mathbf{0} \rangle$

**by** (*simp add: coeff-degree coeffs-of-add-poly h1 h2*)  
**then have**  $f1: \langle \text{degree } a = \text{degree } aa \implies \text{coeff } (a \oplus_{(\text{carrier } R)[X]} aa) (\text{degree } a) \rangle$   
 $= \text{coeff } a (\text{degree } a) \oplus \text{coeff } aa (\text{degree } aa) \rangle$   
**using** *coeffs-of-add-poly h1 h2 by presburger*  
**also have**  $f2: \langle \text{coeff } a (\text{degree } a) \oplus \text{coeff } aa (\text{degree } aa) \neq \mathbf{0} \rangle$  **using**  $h3$   
**by** (*meson add.inv-comm add.inv-unique' coeff-in-carrier h1 h2 local.ring-axioms*)  
  
*ring.polynomial-incl univ-poly-carrier*)  
**then show** *?thesis*  
**apply**(*cases degree a = degree aa*)  
**using**  $f0 f1$   
**apply** (*metis coeff-degree le-neq-implies-less max.idem poly-add-degree univ-poly-add*)  
**apply**(*cases <degree a > degree aa*)  
**by** (*metis carrier-is-subring h1 h2 local.ring-axioms*  
*ring.poly-add-degree-eq univ-poly-add univ-poly-carrier*)+  
**qed**

**lemma** (*in domain*) *inv-imp-zero*:  $\langle a \in \text{carrier}((\text{carrier } R)[X]) \implies a \oplus_{(\text{carrier } R)[X]} a = \mathbf{0} \rangle$   
*inv-add-monoid ((carrier R) [X]) a = []*  
**using** *local.add.Units-eq local.add.Units-r-inv univ-poly-zero*  
**by** (*metis a-inv-def abelian-group.r-neg carrier-is-subring domain.univ-poly-is-abelian-group*  
*domain-axioms*)

**lemma** (*in domain*) *R-subdom*:  $\langle \text{subdomain } (\text{carrier } R) R \rangle$   
**by** (*simp add: carrier-is-subring subdomainI'*)

**lemma** (*in domain*) *lead-coeff-in-carrier*:  
 $\langle \text{ideal } I ((\text{carrier } R)[X]) \implies a \in I \implies \text{coeff } a (\text{degree } a) \in (\text{carrier } R) \rangle$  **for**  $I a$   
**using** *poly-coeff-in-carrier[of <carrier R> a]*  
**by** (*simp add: carrier-is-subring ideal.Icarr univ-poly-carrier*)

**lemma** (*in domain*) *degree-of-inv*:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies \text{degree } (\text{inv}_{\text{add-monoid } ((\text{carrier } R)[X])} p) = \text{degree } p \rangle$  **for**  $p$   
**using** *univ-poly-a-inv-degree[of <carrier R> p]*  
**by** (*simp add: a-inv-def carrier-is-subring*)

**lemma** (*in domain*) *inv-in-deg-poly-set*:  $\langle \text{ideal } I ((\text{carrier } R)[X]) \implies a \in \text{deg-poly-set } I k \implies \text{inv}_{\text{add-monoid } ((\text{carrier } R)[X])} a \in \text{deg-poly-set } I k \rangle$  **for**  $I k a$

**proof** –

**interpret**  $kxr: \text{cring } (\text{carrier } R)[X]$   
**using** *carrier-is-subring univ-poly-is-cring by blast*  
**assume**  $h1: \langle \text{ideal } I ((\text{carrier } R)[X]) \rangle \langle a \in \text{deg-poly-set } I k \rangle$   
**then show** *?thesis*  
**unfolding** *deg-poly-set*  
**apply**(*safe*)  
**apply** (*meson additive-subgroup-def group.subgroupE(3) ideal-def kxr.add.is-group*)

```

    apply (meson degree-of-inv ideal.Icarr)
  by (metis kxr.add.inv-one univ-poly-zero)+
qed

lemma (in domain) ideal-lead-coeff-set:⟨ideal (lead-coeff-set I k) R⟩
  if h':⟨ideal I ((carrier R)[X])⟩ for I k
proof(rule idealI)
  show ⟨ring R⟩
  by (simp add: local.ring-axioms)
next
interpret kxr: cring (carrier R)[X]
  using carrier-is-subring univ-poly-is-cring by blast
show ⟨subgroup (lead-coeff-set I k) (add-monoid R)⟩
  unfolding subgroup-def lead-coeff-set-def
proof(safe)
  fix a x
  assume h1:⟨a ∈ deg-poly-set I k⟩
  show ⟨local.coeff a (degree a) ∈ carrier (add-monoid R)⟩
  using lead-coeff-in-carrier h' h1
  by (metis (no-types, lifting) Un-iff deg-poly-set empty-iff insert-iff
    kxr.oneideal mem-Collect-eq partial-object.select-convs(1) univ-poly-zero-closed)
next
fix x y a aa
assume h1:⟨a ∈ deg-poly-set I k⟩ and h2:⟨aa ∈ deg-poly-set I k⟩
then have imp:⟨a ∈ carrier ((carrier R)[X]) ∧ aa ∈ carrier ((carrier R)[X])⟩
  unfolding deg-poly-set using h' unfolding ideal-def
  by(auto simp:additive-subgroup.a-Hcarr)
then show ⟨∃ ab. local.coeff a (degree a) ⊗add-monoid R local.coeff aa (degree
aa)
= local.coeff ab (degree ab) ∧ ab ∈ deg-poly-set I k⟩
  apply(cases ⟨a=[]⟩)
  using lead-coeff-in-carrier h2 kxr.oneideal apply auto[1]
  apply(cases ⟨aa=[]⟩)
  using lead-coeff-in-carrier h1 kxr.oneideal apply auto[1]
  apply(cases ⟨local.coeff aa (length aa - Suc 0)
≠ invadd-monoid R local.coeff a (length a - Suc 0)⟩)
  apply(rule exI[where x=⟨a ⊕(carrier R)[X] aa⟩])
  using imp length-add h1 h2 unfolding deg-poly-set apply(safe)
  apply (metis One-nat-def coeffs-of-add-poly kxr.add.m-comm max.idem
monoid.select-convs(1))
  apply (meson additive-subgroup.a-closed ideal-def that)
  apply (metis One-nat-def kxr.add.m-comm max.idem)
  by (metis (no-types, lifting) One-nat-def Un-iff add.comm-inv-char add.r-inv-ex
coeff.simps(1)
lead-coeff-in-carrier insert-iff monoid.select-convs(1) that)
next
show ⟨∃ a. 1add-monoid R = local.coeff a (degree a) ∧ a ∈ deg-poly-set I k⟩
  by (smt (verit, ccfv-threshold) Un-insert-right coeff.simps(1) deg-poly-set
insertI1 monoid.select-convs(2))

```

```

next
  fix a
  assume ⟨a ∈ deg-poly-set I k⟩
  obtain a' where ⟨a' = invadd-monoid ((carrier R)[X]) a ∧ a ∈ I⟩
  using h'
  by (metis (no-types, lifting) Un-iff ⟨a ∈ deg-poly-set I k⟩ deg-poly-set empty-iff
insert-iff kxr.add.normal-invE(1)
      kxr.ideal-is-normal mem-Collect-eq monoid.select-convs(2) subgroup-def
univ-poly-zero)
  then show ⟨∃ aa. invadd-monoid R local.coeff a (degree a) = local.coeff aa
(degree aa) ∧ aa ∈ deg-poly-set I k⟩
  apply (intro exI [where x = ⟨invadd-monoid ((carrier R)[X]) a⟩])
  apply (safe)
  apply (metis (no-types, opaque-lifting) degree-of-inv ideal.Icarr kxr.add.Units-eq
kxr.add.Units-inv-closed
      kxr.add.Units-l-inv inv-coeff-sum that univ-poly-zero)
  using ⟨a ∈ deg-poly-set I k⟩ inv-in-deg-poly-set that by blast
qed
next
interpret kxr: cring (carrier R)[X]
using carrier-is-subring univ-poly-is-cring by blast
fix a y
assume h1: ⟨a ∈ lead-coeff-set I k⟩ and h2: ⟨y ∈ (carrier R)⟩
then obtain l where h3: ⟨l ∈ deg-poly-set I k ∧ a = coeff l (degree l)⟩
using lead-coeff-set-def by auto
then have t0: ⟨set l ⊆ (carrier R)⟩
by (metis (no-types, lifting) Un-iff additive-subgroup.a-Hcarr deg-poly-set h'
ideal.axioms(1)
      kxr.zeroideal mem-Collect-eq partial-object.select-convs(1) polynomial-incl
univ-poly-def
      univ-poly-zero)
have t1: ⟨l ∈ carrier ((carrier R)[X])⟩ using h3 h' unfolding deg-poly-set ideal-def

by (auto simp: additive-subgroup.a-Hcarr)
have h4: ⟨y ≠ 0 ⟹ [y] ∈ carrier ((carrier R)[X])⟩
using h2 by (simp add: polynomial-def univ-poly-def)
have f4a: ⟨subring (carrier R) R⟩
using carrier-is-subring by auto
have h5: ⟨y ≠ 0 ⟹ [y] ∈ carrier ((carrier R) [X]) ⟹ l ∈ carrier ((carrier R)[X])

⟹ l ≠ [] ⟹ [y] ⊗(carrier R) [X] l ∈ deg-poly-set I k⟩
using h3 h4 unfolding deg-poly-set apply (safe)
apply (meson h' ideal-axioms-def ideal-def)
unfolding univ-poly-mult
using poly-mult-degree-eq [of (carrier R) ⟨[y]⟩ l]
using f4a univ-poly-carrier by auto
have t4: ⟨y ≠ 0 ⟹ [y] ∈ carrier ((carrier R) [X]) ⟹ l ∈ carrier ((carrier R)[X])
⟹ l ≠ [] ⟹ y ⊗ a = local.coeff ([y] ⊗(carrier R) [X] l) (degree ([y] ⊗(carrier R) [X]
l))⟩

```

```

unfolding univ-poly-mult
by (metis f4a h3 lead-coeff-simp list.sel(1) not-Cons-self poly-mult-integral
      poly-mult-lead-coeff univ-poly-carrier)
have t6:⟨a≠0 ⟹ l≠[]⟩
using h3 by fastforce
show symet:⟨y ⊗ a ∈ lead-coeff-set I k⟩
unfolding lead-coeff-set-def deg-poly-set apply(safe)
apply(cases ⟨a = 0⟩)
apply(rule exI[where x=⟨[]⟩])
apply (simp add: h2)
apply(cases ⟨y=0⟩)
apply(rule exI[where x=⟨[]⟩])
using coeff.simps(1) coeff-in-carrier h2 h3 integral-iff t0 apply simp
apply(rule exI[where x=⟨ [y] ⊗(carrier R) [X] l⟩])
apply(safe)
apply (metis One-nat-def coeff.simps(1) h3 h4 t1 t4)
using h5 h4 t6 by(auto simp add: deg-poly-set t1)
show ⟨a ⊗ y ∈ lead-coeff-set I k⟩
using h2 h3 m-comm symet t0 by auto
qed

lemma (in ring) deg-poly-set-0:⟨deg-poly-set x' 0 = {[a] | a. [a]∈x'⟩∪{[]}⟩ for
x':⟨'c list set⟩
unfolding deg-poly-set
apply(safe)
apply (metis One-nat-def Suc-pred length-0-conv length-Suc-conv length-greater-0-conv)
by(auto)

lemma (in ring) lead-coeff-set-0:⟨lead-coeff-set x' 0 = {a. [a]∈x'⟩∪{0}⟩ for x'
unfolding lead-coeff-set-def
proof(subst deg-poly-set-0, safe)
fix x a aa
assume h1:⟨local.coeff [aa] (degree [aa]) ∉ {}⟩ ⟨local.coeff [aa] (degree [aa]) ≠
0⟩
  ⟨[aa] ∈ x'⟩
then show ⟨local.coeff [aa] (degree [aa]) ∈ x'⟩
by(simp)
next
fix x a
assume h1:⟨local.coeff [] (degree []) ∉ {}⟩ ⟨local.coeff [] (degree []) ≠ 0⟩
then show ⟨local.coeff [] (degree []) ∈ x'⟩ by simp
next
fix x
assume h1:⟨[x] ∈ x'⟩
then show ⟨∃ a. x = local.coeff a (degree a) ∧ a ∈ {[a] | a. [a] ∈ x'⟩ ∪ {[]}⟩
apply(intro exI[where x=⟨[x]⟩])
by(simp)
next
fix x

```

```

show ⟨ $\exists a. \mathbf{0} = \text{local.coeff } a \text{ (degree } a) \wedge a \in \{[a] \mid a. [a] \in x'\} \cup \{\{\}\}$ ⟩
  apply(rule exI[where x=⟨ $\{\}$ ⟩])
  by(simp)
qed

end

```

## 4 The weak Hilbert Basis theorem

**theory** *Weak-Hilbert-Basis*

**imports**

*HOL-Algebra.Polynomials*  
*HOL-Algebra.Indexed-Polynomials*  
*Polynomials-Ring-Misc*  
*Padic-Field.Cring-Multivariable-Poly*  
*HOL-Algebra.Module*  
*Ring-Misc*

**begin**

In this section, we show what we called "weak" Hilbert basis theorem, meaning Hilbert basis theorem for univariate polynomials. The theorem is done for all three (Polynomials, UP, IP with card = 1) models of polynomials that exists in HOL-Algebra

### 4.1 Weak Hilbert Basis

```

lemma (in noetherian-domain) weak-Hilbert-basis:⟨noetherian-ring ((carrier R)[X])⟩
proof(rule ring.trivial-ideal-chain-imp-noetherian)
  show ⟨ring ((carrier R) [X])⟩
    using carrier-is-subring univ-poly-is-ring by blast
  next
  interpret kxr: cring (carrier R)[X]
    using carrier-is-subring univ-poly-is-cring by blast
  fix C
  assume F:⟨ $C \neq \{\}$ ⟩ ⟨subset.chain {I. ideal I ((carrier R) [X])} C⟩
  have f1:⟨ $I \in C \implies \text{ideal } I \text{ (carrier R)[X]}$ ⟩ for I
    using F unfolding subset.chain-def by(auto)
  have f2:⟨ $a \in \text{carrier}((\text{carrier } R)[X]) \wedge aa \in \text{carrier}((\text{carrier } R)[X])$ 
     $\implies \text{coeff } (a \oplus_{(\text{carrier } R)[X]} aa) k = \text{coeff } a k \oplus \text{coeff } aa k$ ⟩
    for a aa k
  unfolding univ-poly-add
  apply(subst poly-add-coeff)
  using polynomial-in-carrier[of ⟨carrier R⟩ a] polynomial-in-carrier[of ⟨carrier
R⟩ aa]
    polynomial-def carrier-is-subring
  by (simp add: univ-poly-carrier)+
  have f4a:⟨subring (carrier R) R⟩

```



```

using carrier-is-subring by auto
have degree-of-inv:
  ⟨p ∈ carrier((carrier R)[X]) ⟹ degree (inv_add-monoid ((carrier R)[X]) p) = de-
gree p⟩ for p
  by (metis a-inv-def local.ring-axioms ring.carrier-is-subring univ-poly-a-inv-degree)
from f1 have ⟨I ∈ C ⟹ a ∈ I ⟹ coeff a (degree a) ∈ (carrier R)⟩ for a I
  using lead-coeff-in-carrier by blast
have emp-in-i: ⟨ideal I ((carrier R)[X]) ⟹ [] ∈ I⟩ for I
  by (simp add: additive-subgroup-def ideal-def subgroup-def univ-poly-zero)
have g0: ⟨I ⊆ I' ⟹ lead-coeff-set I k ⊆ lead-coeff-set I' k⟩
  for I I' k
  unfolding lead-coeff-set-def deg-poly-set by (auto)
have g1: ⟨ideal I ((carrier R)[X]) ⟹ {X ⊗(carrier R)[X] l | l. l ∈ I} ⊆ I⟩ for I
  using f4a ideal.I-l-closed var-closed(1) by fastforce
then have g2:
  ⟨ideal I ((carrier R)[X]) ⟹ lead-coeff-set {X ⊗(carrier R)[X] l | l. l ∈ I} k ⊆
lead-coeff-set I k⟩
  for I k
  using g0 g1 by auto
have f7b: ⟨ideal I ((carrier R)[X]) ⟹ lead-coeff-set I k ⊆ lead-coeff-set I (k+1)⟩
for I k
  unfolding lead-coeff-set-def deg-poly-set
proof (safe)
  fix x a
  assume y1: ⟨ideal I (poly-ring R)⟩ ⟨a ∈ I⟩ ⟨k = degree a⟩
  then show
    ⟨∃ aa. local.coeff a (degree a) = local.coeff aa (degree aa) ∧ aa ∈ {aa ∈ I.
degree aa = degree a + 1} ∪ {[]}⟩
    apply (cases ⟨a = []⟩)
    apply (rule exI[where x = ⟨[]⟩])
    apply blast
    apply (rule exI[where x = ⟨a ⊗(carrier R)[X] X⟩])
    apply (safe)
    unfolding ideal-def univ-poly-mult
    using poly-mult-var[of (carrier R) a for a]
    apply (metis One-nat-def additive-subgroup.a-Hcarr
append-is-Nil-conv f4a hd-append2 lead-coeff-simp univ-poly-mult)
    apply (simp add: f4a ideal-axioms-def univ-poly-mult var-closed(1))
    using poly-mult-var[of ⟨(carrier R)⟩ a for a]
    by (metis Suc-eq-plus1 Suc-pred' diff-Suc-Suc f4a ideal.Icarr length-append-singleton

length-greater-0-conv minus-nat.diff-0 univ-poly-mult y1(1))
  next
  assume y1: ⟨ideal I (poly-ring R)⟩
  then show
    ⟨∃ a. local.coeff [] (degree []) = local.coeff a (degree a) ∧ a ∈ {a ∈ I. degree a
= k + 1} ∪ {[]}⟩
    by force
qed

```

```

then have f7:⟨y∈C ⟹ lead-coeff-set y k ⊆ lead-coeff-set y (k+1)⟩ for k y
  using f1 by blast
then have f8:⟨k≤k' ⟹ y∈C ⟹ lead-coeff-set y k ⊆ lead-coeff-set y k'⟩ for k
k' y
  apply(induct k')
  using le-Suc-eq by(auto)
  have n:⟨noetherian-ring R⟩
  by (simp add: noetherian-ring-axioms)
  have c:⟨x∈C ⟹ subset.chain {I. ideal I R} {lead-coeff-set x k | k. k∈UNIV}⟩
for x
  apply(subst subset-chain-def)
  apply(safe)
  apply (simp add: f1 ideal-lead-coeff-set)
  by (meson f8 nle-le subsetD)
  have c':⟨ subset.chain {I. ideal I R} {lead-coeff-set x k | x. x∈C}⟩ for k
proof(rule Zorn.subset.chainI)
  show ⟨{lead-coeff-set x k | x. x ∈ C} ⊆ {I. ideal I R}⟩
  using f1 ideal-lead-coeff-set by blast
next
  fix xa y
  assume 1:⟨xa ∈ {lead-coeff-set x k | x. x ∈ C}⟩ ⟨ y ∈ {lead-coeff-set x k | x. x ∈
C}⟩
  obtain z z' where g10:⟨xa = lead-coeff-set z k ∧ y = lead-coeff-set z' k ∧ z∈C
∧ z' ∈ C ⟩
  using 1(1) 1(2) by blast
  then have ⟨z⊆z' ∨ z'⊆z⟩
  using F unfolding subset.chain-def by(auto)
  then show ⟨(C)== xa y ∨ (C)== y xa⟩
  using g0 g10 by blast+
qed
then have U0:⟨∀ x∈C. (⋃ {lead-coeff-set x k | k. k∈UNIV}) ∈ {lead-coeff-set x
k | k. k∈UNIV}⟩
proof(safe)
  fix x
  assume a1: x ∈ C
  have ∀ A. ¬ subset.chain {A. ideal A R} A ∨ ⋃ A ∈ A ∨ A = {}
  using ideal-chain-is-trivial by blast
  then show ⟨∃ k. ⋃ {lead-coeff-set x k | k. k ∈ UNIV} = lead-coeff-set x k ∧ k
∈ UNIV⟩
  using a1 c by auto
qed
have t9:⟨x∈C ⟹ ideal (lead-coeff-set x k) R⟩ for k x
  using f1 ideal-lead-coeff-set by blast
then have degree-of-inv:⟨{lead-coeff-set x k | x. x∈C} ≠ {}⟩ for x::⟨'a set⟩ and
k
  using F(1) by blast
then have U1:⟨∀ k. (⋃ {lead-coeff-set x k | x. x ∈ C}) ∈ {lead-coeff-set x k | x.
x∈C}⟩
  using ideal-lead-coeff-set f7b n c'

```

```

    using ideal-chain-is-trivial[OF degree-of-inv c'] by(auto)
  have kl0:⟨x∈C ∧ y∈C⟹x=y ⟷ (∀k. deg-poly-set x k = deg-poly-set y k)⟩ for
x y
  proof(safe)
    fix xa :: 'a list
    assume a1: y ∈ C
    assume a2: ∀k. deg-poly-set x k = deg-poly-set y k
    assume xa ∈ x
    then have ∃n. xa ∈ deg-poly-set x n
      using deg-poly-set noetherian-domain-axioms by fastforce
    then show xa ∈ y
      using a2 a1
      by (metis (no-types, lifting) UnE emp-in-i f1 local.ring-axioms
        mem-Collect-eq ring.deg-poly-set singleton-iff)
  next
    fix xa :: 'a list
    assume a1: x ∈ C
    assume a2: ∀k. deg-poly-set x k = deg-poly-set y k
    assume xa ∈ y
    then have ∃n. xa ∈ deg-poly-set y n
      using deg-poly-set noetherian-domain-axioms by fastforce
    then show xa ∈ x
      using a2 a1
      by (metis (no-types, lifting) UnE emp-in-i f1 local.ring-axioms
        mem-Collect-eq ring.deg-poly-set singleton-iff)
  qed
  have kl:⟨x'∈C ∧ y∈C ∧ x'⊆y⟹(∀k≤n. lead-coeff-set x' k = lead-coeff-set y k)
    ⟷ (∀k≤n. deg-poly-set x' k = deg-poly-set y k)⟩
  for x' y n
  apply(rule iffI)
  subgoal
  proof(induct n)
    case z:0
    from lead-coeff-set-0 have d2:⟨{a. [a] ∈ x'} = {a. [a] ∈ y}⟩
      using z(2)[rule-format, of 0] unfolding lead-coeff-set-def
      using z.prem1 f1 unfolding ideal-def
    by (simp add:f1 ideal-def polynomial-def univ-poly-carrier additive-subgroup.a-Hcarr)
      (metis (mono-tags, lifting) additive-subgroup.a-Hcarr insert-iff
        list.sel(1) list.simps(3) mem-Collect-eq polynomial-def univ-poly-carrier)
    show ?case
      apply(insert z)
      apply(simp)
      apply(subst (asm) (1 2) lead-coeff-set-0)
      apply(subst (1 2) deg-poly-set-0)
      using d2 by(auto)
  next
    case (Suc n)
    have t0:⟨∀k≤n. deg-poly-set x' k = deg-poly-set y k⟩

```

```

using Suc.hyps Suc.prem(1) Suc.prem(2) le-Suc-eq by blast
have  $t': \langle \text{ideal } x' ((\text{carrier } R)[X]) \rangle$ 
using Suc.prem(1) f1 by blast
have  $t: \langle \text{deg-poly-set } x' (\text{Suc } n) = \text{deg-poly-set } y (\text{Suc } n) \implies ?\text{case} \rangle$ 
using Suc.hyps Suc.prem(1) Suc.prem(2) le-Suc-eq by presburger
have  $\langle \forall k. \exists S. \text{lead-coeff-set } x' k = \text{genideal } R (S k) \wedge \text{finite } (S k) \rangle$ 
by (meson ideal-lead-coeff-set finetely-gen t')
then have  $\langle \exists S. \forall k. \text{lead-coeff-set } x' k = \text{genideal } R (S k) \wedge \text{finite } (S k) \rangle$ 
by moura
then obtain  $S$  where  $t1: \langle \forall k. \text{lead-coeff-set } x' k = \text{genideal } R (S k) \wedge \text{finite } (S k) \rangle$ 
by (blast)
then have  $\langle \forall k \leq \text{Suc } n. \text{lead-coeff-set } y (k) = \text{genideal } R (S k) \rangle$ 
using Suc.prem(2) le-Suc-eq by presburger
show  $?\text{case}$ 
proof(rule t)
show  $\langle \text{deg-poly-set } x' (\text{Suc } n) = \text{deg-poly-set } y (\text{Suc } n) \rangle$ 
unfolding deg-poly-set
proof(safe)
fix  $x$ 
assume  $2: \langle x \notin \{ \} \rangle \langle x \neq [] \rangle \langle x \in x' \rangle \langle \text{degree } x = \text{Suc } n \rangle$ 
then show  $\langle x \in y \rangle$ 
using Suc.prem(1) by blast
next
fix  $x$ 
assume  $2: \langle x \notin \{ \} \rangle \langle x \neq [] \rangle \langle x \in y \rangle \langle \text{degree } x = \text{Suc } n \rangle$ 
{assume  $1: \langle x \neq [] \rangle \langle x \in y \rangle \langle \text{length } x - \text{Suc } 0 = \text{Suc } n \rangle \langle x \notin x' \rangle$ 
have  $\langle \text{lead-coeff-set } x' (\text{Suc } n) = \text{lead-coeff-set } y (\text{Suc } n) \rangle$ 
using Suc.prem(2) by auto
then have  $tp: \langle \text{coeff } x (\text{degree } x) \in \text{lead-coeff-set } x' (\text{Suc } n) \rangle$ 
by (metis (mono-tags, lifting) 1(2) 1(3) One-nat-def Un-iff deg-poly-set lead-coeff-set-def mem-Collect-eq)
then have  $\langle \exists x2. x2 \neq x \wedge x2 \in x' \wedge \text{coeff } x2 (\text{degree } x2) = \text{coeff } x (\text{degree } x) \wedge \text{degree } x2 = \text{Suc } n \rangle$ 
unfolding lead-coeff-set-def by(simp) (metis (mono-tags, lifting) 1(1) 1(2) 1(4) One-nat-def Suc.prem(1) Un-iff coeff.simps(1) deg-poly-set f1 ideal.Icarr lead-coeff-simp mem-Collect-eq partial-object.select-convs(1) polynomial-def singletonD univ-poly-def)
then obtain  $x2$  where  $g1: \langle \text{coeff } x2 (\text{degree } x2) \in \text{lead-coeff-set } x' (\text{Suc } n) \wedge x2 \neq x \wedge \text{degree } x2 = \text{Suc } n \wedge x2 \in x' \wedge \text{coeff } x2 (\text{degree } x2) = \text{coeff } x (\text{degree } x) \rangle$ 
using  $tp$  by force
then have  $g2: \langle x2 \in y \rangle$ 
using Suc.prem(1) by blast
then have  $g3: \langle x \oplus (\text{carrier } R)[X] \text{ inv-add-monoid } ((\text{carrier } R)[X]) x2 \in y \rangle$ 
using  $t'$ 

```

**by** (*meson* 1(2) *Suc.prem*s(1) *additive-subgroup.a-closed additive-subgroup-def*  
*f1 group.subgroupE*(3) *ideal-def kxr.add.group-l-invI kxr.add.l-inv-ex*)  
**then have**  $g4: \langle x \oplus_{(\text{carrier } R)[X]} \text{inv\_add\_monoid } ((\text{carrier } R)[X]) \ x2 \notin x' \rangle$   
**using** *t' g1 1(2) 1(4) f1 Suc.prem*s(1)  
*kxr.add.m-assoc kxr.add.r-inv-ex kxr.add.subgroupE*(4) *kxr.minus-unique*  
*kxr.r-zero*  
**unfolding** *additive-subgroup-def ideal-def*  
**by** (*smt* (*verit*, *best*) *f1 ideal.Icarr kxr.add.comm-inv-char*)  
**have**  $\langle \text{degree } x = \text{Suc } n \wedge \text{degree } x2 = \text{Suc } n \rangle$   
**using** 1 *g1* **by** *auto*  
**also have**  $\langle \text{coeff } (\text{inv\_add\_monoid } ((\text{carrier } R)[X]) \ x2) (\text{degree } x2) =$   
 $\text{inv\_add\_monoid } R (\text{coeff } x (\text{degree } x)) \rangle$   
**by** (*smt* (*verit*, *best*) *a-inv-def diff-0-eq-0 f4a g1 ideal.Icarr kxr.add.inv-closed*  
*kxr.l-neg length-add list.size*(3) *max.idem nat.discI t' univ-poly-a-inv-degree*  
*univ-poly-zero*)  
**then have**  $\langle \text{coeff } ((x \oplus_{(\text{carrier } R)[X]} \text{inv\_add\_monoid } ((\text{carrier } R)[X]) \ x2))$   
 $(\text{Suc } n) = \mathbf{0} \rangle$   
**by** (*smt* (*verit*, *best*) 1(2) *Suc.prem*s(1)  $\langle \text{degree } x = \text{Suc } n \wedge \text{degree } x2$   
 $= \text{Suc } n \rangle$   
*a-inv-def add.Units-eq add.Units-r-inv lead-coeff-in-carrier f1 f2 g1*  
*ideal.Icarr kxr.add.inv-closed*)  
**then have**  $\ast: \langle \forall k \geq \text{Suc } n. \text{coeff } ((x \oplus_{(\text{carrier } R)[X]} \text{inv\_add\_monoid } ((\text{carrier } R)[X])$   
 $x2)) (k) = \mathbf{0} \rangle$   
**by** (*smt* (*verit*, *best*) 1(2) *Suc.prem*s(1) *a-inv-def calculation coeff-degree*  
*f1 f2 f4a g2*  
*ideal.Icarr kxr.add.inv-closed l-zero le-eq-less-or-eq univ-poly-a-inv-degree*  
*zero-closed*)  
**then have**  $\ast\ast: \langle \text{degree } (x \oplus_{(\text{carrier } R)[X]} \text{inv\_add\_monoid } ((\text{carrier } R)[X])$   
 $x2) \leq \text{Suc } n \rangle$   
**unfolding** *univ-poly-add*  
**by** (*metis* (*no-types*, *lifting*) *a-inv-def calculation f4a g1 ideal.Icarr*  
*max.idem poly-add-degree t' univ-poly-a-inv-degree univ-poly-add*)  
**then have**  $b0: \langle \text{coeff } ((x \oplus_{(\text{carrier } R)[X]} \text{inv\_add\_monoid } ((\text{carrier } R)[X])$   
 $x2)) (\text{Suc } n) = \mathbf{0} \rangle$   
**using**  $\ast$  **by** *auto*  
**have**  $b1: \langle x \in (\text{carrier } ((\text{carrier } R)[X])) \implies \text{degree } x \leq \text{Suc } n \wedge \text{coeff } x$   
 $(\text{Suc } n) = \mathbf{0} \implies \text{degree } x \leq n \rangle$  **for**  $x$   
**by** (*metis* *diff-0-eq-0 diff-Suc-1 le-SucE lead-coeff-simp list.size*(3)  
*polynomial-def univ-poly-carrier*)  
**then have**  $\langle \text{degree } (x \oplus_{(\text{carrier } R)[X]} \text{inv\_add\_monoid } ((\text{carrier } R)[X]) \ x2)$   
 $\leq n \rangle$   
**using**  $b0 \ b1 \ \ast\ast$   
**by** (*meson* *Suc.prem*s(1) *f1 g3 ideal.Icarr*)  
**then obtain**  $k$  **where**  $n: \langle k \leq n \wedge k = \text{degree } (x \oplus_{(\text{carrier } R)[X]}$   
 $\text{inv\_add\_monoid } ((\text{carrier } R)[X]) \ x2) \rangle$   
**by** *blast*

**then have**  $\langle x \oplus_{(\text{carrier } R)[X]} \text{inv\_add-monoid } ((\text{carrier } R)[X]) \ x2 \in \text{deg-poly-set } y \ k \wedge x \oplus_{(\text{carrier } R)[X]} \text{inv\_add-monoid } ((\text{carrier } R)[X]) \ x2 \notin \text{deg-poly-set } x' \ k \rangle$

**unfolding** *deg-poly-set* **using** *g1 g2 g3 monoid.cases monoid.simps(1) monoid.simps(2)*

*partial-object.select-convs(1) emp-in-i g4 t'* **by** *fastforce*

**then have** *False* **using** *n t0* **by** *blast*

**note** *this-is-proof=this*

**then show**  $\langle x \in x' \rangle$

**using** *this-is-proof 2(2) 2(3) 2(4) One-nat-def* **by** *argo*

**qed**

**qed**

**qed**

**using** *lead-coeff-set-def* **by** *presburger*

**have** *chain-is:  $\langle x' \in C \wedge y \in C \implies x' \subseteq y \vee y \subseteq x' \rangle$*  **for**  $x' \ y$

**using** *F* **unfolding** *subset.chain-def* **by** *(auto)*

**from** *kl* **have** *imppp:  $\langle x' \in C \wedge y \in C \wedge x' \subseteq y \implies (\forall k. \text{lead-coeff-set } x' \ k = \text{lead-coeff-set } y \ k) \iff (\forall k. \text{deg-poly-set } x' \ k = \text{deg-poly-set } y \ k) \rangle$*

**for**  $x' \ y$

**by** *(meson dual-order.refl)*

**then have** *impp:  $\langle x' \in C \wedge y \in C \implies (\forall k. \text{lead-coeff-set } x' \ k = \text{lead-coeff-set } y \ k) \iff (\forall k. \text{deg-poly-set } x' \ k = \text{deg-poly-set } y \ k) \rangle$*

**for**  $x' \ y$

**by** *(metis chain-is)*

**then have** *sup1:  $\langle x' \in C \wedge y \in C \implies (x' = y) \iff (\forall k. \text{lead-coeff-set } x' \ k = \text{lead-coeff-set } y \ k) \rangle$*  **for**  $x' \ y$

**using** *kl0* **by** *presburger*

**then have**  $\langle \exists Ux. \forall k. Ux \ k = \bigcup \{ \text{lead-coeff-set } x \ k \mid x. x \in C \} \rangle$

**by** *auto*

**then obtain**  $Ux$  **where**  $Ux: \langle \forall k. Ux \ k = \bigcup \{ \text{lead-coeff-set } x \ k \mid x. x \in C \} \rangle$  **by** *blast*

**then have**  $\langle \exists Uk. \forall x \in C. (Uk \ x = \bigcup \{ \text{lead-coeff-set } x \ k \mid k. k \in UNIV \}) \rangle$  **using** *U0* **by** *auto*

**then obtain**  $Uk$  **where**  $Uk: \langle \forall x \in C. (Uk \ x = \bigcup \{ \text{lead-coeff-set } x \ k \mid k. k \in UNIV \}) \rangle$

**using** *U0* **by** *(auto)*

**have**  $\langle (\bigcup \{ \text{lead-coeff-set } x \ k \mid x \ k. x \in C \wedge k \in UNIV \}) = (\bigcup x \in C. (\bigcup k. \text{lead-coeff-set } x \ k)) \rangle$

**by** *auto*

**have**  $\langle (\bigcup x \in C. (\bigcup k. \text{lead-coeff-set } x \ k)) \in \{ \text{lead-coeff-set } x \ k \mid x \ k. x \in C \} \rangle$

**proof** –

**have**  $n0: \langle x \in C \wedge y \in C \wedge x \subseteq y \implies (\bigcup k. \text{lead-coeff-set } x \ k) \subseteq (\bigcup k. \text{lead-coeff-set } y \ k) \rangle$  **for**  $x \ y$

**by** *(simp add: SUP-mono' g0)*

**obtain**  $s1$  **where**  $n1: \langle (\forall x \in C. (\bigcup k. \text{lead-coeff-set } x \ k) = \text{lead-coeff-set } x \ (s1 \ x)) \rangle$

**using** *U0*

**by** *(simp)(metis full-SetCompr-eq)*

**then have**  $n4: \langle (\bigcup x \in C. (\bigcup k. \text{lead-coeff-set } x \ k)) = (\bigcup x \in C. \text{lead-coeff-set } x \ (s1 \ x)) \rangle$

```

(s1 x))›
  by auto
  have ⟨x ∈ C ∧ y ∈ C ⟹ x ⊆ y ∨ y ⊆ x⟩ for x y
    using F unfolding subset.chain-def by(auto)
  then have n1:⟨x ∈ C ∧ y ∈ C ⟹ lead-coeff-set x (s1 x) ⊆ lead-coeff-set y (s1
y) ∨
    lead-coeff-set y (s1 y) ⊆ lead-coeff-set x (s1 x)⟩
    for x y
    apply(cases ⟨x ⊆ y⟩)
    apply(rule disjI1)
    subgoal using n0 n1 by auto[1]
    by (metis n0 n1)
  have n2:⟨subset.chain {I. ideal I R} {lead-coeff-set x (s1 x) | x. x ∈ C}⟩
    apply(rule subset.chainI)
    using ⟨∧ x k. x ∈ C ⟹ ideal (lead-coeff-set x k) R⟩ apply force
    using n1 by auto
  have n3:⟨{lead-coeff-set x (s1 x) | x. x ∈ C} ≠ {}⟩
    using F(1) by blast
  have ⟨(∪ x ∈ C. lead-coeff-set x (s1 x)) = (∪ {lead-coeff-set x (s1 x) | x. x ∈ C})⟩
    by auto
  then have ⟨(∪ x ∈ C. lead-coeff-set x (s1 x)) ∈ {lead-coeff-set x (s1 x) | x. x ∈ C}⟩
    using ideal-chain-is-trivial[OF n3 n2]
    by(auto)
  then show ⟨(∪ x ∈ C. ∪ (range (lead-coeff-set x))) ∈ {lead-coeff-set x k | x k. x
∈ C} ⟩
    using n4 by auto
  qed
  then obtain x l where n5:⟨(∪ {lead-coeff-set x k | x k. x ∈ C}) = lead-coeff-set
x l ∧ x ∈ C⟩
    using ⟨∪ {lead-coeff-set x k | x k. x ∈ C ∧ k ∈ UNIV} = (∪ x ∈ C. ∪ (range
(lead-coeff-set x)))⟩
    by auto
  then have ⟨∀ y ∈ C. x ⊆ y ⟶ (∀ n ≥ l. (lead-coeff-set y n = lead-coeff-set x l))⟩
    apply(safe)
    subgoal using UnionI by blast
    by (meson f8 g0 in-mono)
  have ⟨∀ k. ∃ y'. ∪ {lead-coeff-set x k | x. x ∈ C} = lead-coeff-set (y' k) k ∧ y' k
∈ C⟩
    using U1 by fastforce
  then have ⟨∃ y'. ∀ k. ∪ {lead-coeff-set x k | x. x ∈ C} = lead-coeff-set (y' k) k ∧
y' k ∈ C⟩
    by moura
  then obtain y' where n10:⟨∪ {lead-coeff-set x k | x. x ∈ C} = lead-coeff-set (y'
k) k ∧ y' k ∈ C⟩
    for k
    by blast
  have n8:⟨({y' k | k. k ≤ l} ∪ {x}) ⊆ C⟩
    using ⟨∧ k. ∪ {lead-coeff-set x k | x. x ∈ C} = lead-coeff-set (y' k) k ∧ y' k ∈
C⟩ n5 by auto

```

```

then have fin:⟨finite ( $\{y' k \mid k. k \leq l\} \cup \{x\}$ )⟩
  by (auto)
have n9:⟨subset.chain C ( $\{y' k \mid k. k \leq l\} \cup \{x\}$ )⟩
  apply (rule subset.chainI)
  using n8 apply force
  using F(2) n8 unfolding subset.chain-def
  by (meson subset-eq)
then obtain M where n11:⟨ $M \in C \wedge (\bigcup (\{y' k \mid k. k \leq l\} \cup \{x\}) = M)$ ⟩
  unfolding subset-chain-def
  by (metis (no-types, lifting) Un-empty Union-in-chain n9 fin insert-not-empty subsetD)
  then have ⟨ $\forall y \in C. M \subseteq y \longrightarrow (\forall n \leq l. \text{lead-coeff-set } y \ n = \text{lead-coeff-set } (y' \ n))$ ⟩
  using n10 g0 apply (safe)
  using Sup-le-iff mem-Collect-eq by blast+
  then have nn:⟨ $\forall y \in C. \forall n \leq l. M \subseteq y \longrightarrow (\text{lead-coeff-set } (y) \ n = \text{lead-coeff-set } M \ n)$ ⟩
  using ⟨ $M \in C \wedge \bigcup (\{y' k \mid k. k \leq l\} \cup \{x\}) = M$ ⟩ by auto
  then have ⟨ $\forall y \in C. \forall n \geq l. M \subseteq y \longrightarrow (\text{lead-coeff-set } (y) \ n = \text{lead-coeff-set } M \ n)$ ⟩
  using ⟨ $M \in C \wedge \bigcup (\{y' k \mid k. k \leq l\} \cup \{x\}) = M$ ⟩
  using ⟨ $\forall y \in C. x \subseteq y \longrightarrow (\forall n \geq l. \text{lead-coeff-set } y \ n = \text{lead-coeff-set } x \ l)$ ⟩ by
auto
  then have n-1:⟨ $\forall y \in C. M \subseteq y \longrightarrow M = y$ ⟩
  by (metis n11 sup1 nn linorder-le-cases)
  have ⟨ $\bigcup C = M$ ⟩
  proof (rule ccontr)
    assume h-1:⟨ $\bigcup C \neq M$ ⟩
    then have f-0:⟨ $\exists x \in \bigcup C. x \notin M$ ⟩
    by (meson UnionI ⟨ $M \in C \wedge \bigcup (\{y' k \mid k. k \leq l\} \cup \{x\}) = M$ ⟩ subset-antisym subset-iff)
    then obtain x where f-1:⟨ $x \in \bigcup C \wedge x \notin M$ ⟩ by blast
    then have f-3:⟨ $\exists M' \in C. x \in M'$ ⟩
    by blast
    then obtain M' where f-2:⟨ $x \in M'$ ⟩ by blast
    then have ⟨ $M \subseteq M' \wedge M \neq M'$ ⟩
    using F unfolding subset-chain-def
    by (metis f-1 f-3 n11 n-1 subsetD)
    then show False
    using n-1 n11 f-1 f-3 F(2) unfolding subset-chain-def
    by (metis subsetD)
  qed
  then show ⟨ $\bigcup C \in C$ ⟩
  by (simp add: n11)
qed

```

## 4.2 Some properties of noetherian rings

Assuming  $I$  is an ideal of  $A$  and  $A$  is noetherian, then  $A/I$  is noetherian.

**lemma** *noetherian-ring-imp-quot-noetherian-ring*:



```

assumes  $h1$ :⟨noetherian-ring  $A$ ⟩ and  $h2$ :⟨ideal  $I A$ ⟩
shows⟨noetherian-ring ( $A \text{ Quot } I$ )⟩
proof –
interpret  $cr$ :ring  $A$ 
  using  $h1$  unfolding noetherian-ring-def by(auto)
interpret  $crI$ : ring ( $A \text{ Quot } I$ )
  by (simp add:  $h2$  ideal.quotient-is-ring)
interpret  $rhr$ :ring-hom-ring  $A$  ( $A \text{ Quot } I$ ) (( $+>_A$ )  $I$ )
  using  $h2$  ideal.rcos-ring-hom-ring by blast
have  $rhr$ -p:⟨ring-hom-ring  $A$  ( $A \text{ Quot } I$ ) (( $+>_A$ )  $I$ )⟩
  using  $h2$  ideal.rcos-ring-hom-ring by blast
from  $h1$  show ?thesis
proof(intro ring.trivial-ideal-chain-imp-noetherian)
  assume  $1$ :⟨noetherian-ring  $A$ ⟩
  show ⟨ring ( $A \text{ Quot } I$ )⟩
    by (simp add:  $crI$ .ring-axioms)
next
fix  $C$ 
  assume  $1$ :⟨noetherian-ring  $A$ ⟩ ⟨ $C \neq \{\}$ ⟩ ⟨subset.chain { $Ia$ . ideal  $Ia$  ( $A \text{ Quot } I$ )}  $C$ ⟩
  let ? $f$ =⟨the-inv-into ({ $J$ . ideal  $J A \wedge I \subseteq J$ }) ( $\lambda x$ . ( $+>_A$ )  $I$  '  $x$ )⟩
  have  $inv$ -imp:⟨ $\forall J \in \{J$ . ideal  $J A \wedge I \subseteq J\}$ . ? $f$  (( $+>_A$ )  $I$  '  $J$ ) =  $J$ ⟩
    using the-inv-into-onto[of ⟨ $\lambda x$ . ( $+>_A$ )  $I$  '  $x$ ⟩ ⟨ $J$ . ideal  $J A \wedge I \subseteq J$ ⟩]
    apply(subst set-eq-iff)
    by (metis (no-types, lifting) Collect-cong bij-betw-def  $cr$ .ring-axioms  $h2$ 
      ring.quot-ideal-correspondence the-inv-into-f-f)+
  have  $rule$ -inv:⟨ $x \in$  the-inv-into { $J$ . ideal  $J A \wedge I \subseteq J$ } (( $\cdot$ ) (( $+>_A$ )  $I$ ))  $J$ 
     $\implies$  ideal  $J$  ( $A \text{ Quot } I$ )  $\implies$  ideal  $J'$  ( $A \text{ Quot } I$ )  $\implies J \subseteq J'$ 
     $\implies x \in$  the-inv-into { $J$ . ideal  $J A \wedge I \subseteq J$ } (( $\cdot$ ) (( $+>_A$ )  $I$ ))  $J'$ ⟩
  for  $x J J'$ 
  by (smt (verit, best) Collect-cong additive-subgroup.a-subset bij-betw-imp-surj-on

       $cr$ .canonical-proj-vimage-mem-iff  $f$ -the-inv-into-f-bij-betw  $h2$  ideal-def im-
age-eqI
      image-eqI inj-onI mem-Collect-eq mem-Collect-eq ring.ideal-incl-iff
      ring.quot-ideal-correspondence subsetD the-inv-into-onto)
  have  $inv$ :⟨bij-betw ? $f$  { $J$ . ideal  $J$  ( $A \text{ Quot } I$ )} { $J$ . ideal  $J A \wedge I \subseteq J$ }
     $\wedge$  ( $\forall J J'$ . { $J, J'$ }  $\subseteq$  { $J$ . ideal  $J$  ( $A \text{ Quot } I$ )}  $\wedge J \subseteq J' \implies ?f J \subseteq ?f J'$ )⟩
    using ring.quot-ideal-correspondence[of  $A I$ ] the-inv-into-onto[of ⟨ $\lambda x$ . ( $+>_A$ )
 $I$  '  $x$ ⟩
      ⟨{ $J$ . ideal  $J A \wedge I \subseteq J$ }⟩]
  unfolding bij-betw-def
  using  $cr$ .ring-axioms  $h2$  the-inv-into-onto inj-on-the-inv-into  $f$ -the-inv-into- $f$ 
    inj-on-the-inv-into[of ⟨ $\lambda x$ . ( $+>_A$ )  $I$  '  $x$ ⟩ ⟨{ $J$ . ideal  $J A \wedge I \subseteq J$ }⟩]
    additive-subgroup.a-subset  $cr$ .canonical-proj-vimage-mem-iff
     $f$ -the-inv-into- $f$ [of ⟨ $\lambda x$ . ( $+>_A$ )  $I$  '  $x$ ⟩ ⟨{ $J$ . ideal  $J A \wedge I \subseteq J$ }⟩]
    ideal-def image-eqI mem-Collect-eq ring.ideal-incl-iff subsetD
  by(auto simp:  $rule$ -inv)
then have ⟨ $\forall c \in C$ . ideal ( $?f c$ )  $A$ ⟩

```

```

    using 1(3) inv unfolding subset.chain-def
    using bij-betwE by fast
  have inv-imp2:⟨∀ J∈{J. ideal J (A Quot I)}. ((+>_A) I ‘ ?f J) = J⟩
    by (smt (verit, del-insts) Collect-cong bij-betw-def cr.ring-axioms
        h2 imageE inv-imp ring.quot-ideal-correspondence)
  have ⟨∀ c c'. c ∈ C ∧ c' ∈ C ∧ c ⊆ c' ⟶ ?f c ⊆ ?f c'⟩
    using inv using 1(3) unfolding subset.chain-def
    by (meson empty-subsetI insert-subset subsetD)
  then have sub1:⟨subset.chain {Ia. ideal Ia (A)} (?f C)⟩
    using 1(3) unfolding subset.chain-def image-def
    using ⟨∀ c∈C. ideal (?f c) A⟩ apply(safe)
    apply (simp add: image-def)
    by (meson in-mono)
  have sub2 :⟨(?f C) ≠ {}⟩
    using 1(2) by blast
  then have k0:⟨(⋃ (?f C)) ∈ (?f C)⟩
    by (metis (no-types) h1 noetherian-ring.ideal-chain-is-trivial sub1 sub2)
  then have ⟨(+>_A) I ‘ (⋃ (?f C)) = (⋃ C)⟩
    apply(safe)
    apply (smt (verit, del-insts) 1(3) UnionI image-eqI inv-imp2 subset.chain-def
        subsetD)
    by (smt (verit, best) 1(3) SUP-upper in-mono inv-imp2 subset.chain-def
        subset-image-iff)
    then show ⟨⋃ C ∈ C⟩
    by (smt (verit) 1(3) k0 image-iff inv-imp2 subset.chain-def subsetD)
qed
qed

```

If  $A$  is noetherian and  $A \simeq B$  then  $B$  is noetherian.

```

lemma noetherian-isom-imp-noetherian:
  assumes h1:⟨noetherian-ring A ∧ ring B ∧ A ≃ B⟩
  shows ⟨noetherian-ring B⟩
proof(rule ring.trivial-ideal-chain-imp-noetherian)
  show ⟨ring B⟩ using h1 by(simp)
next
fix C
assume h2:⟨C≠{}⟩ and h3:⟨subset.chain {I. ideal I B} C⟩
obtain g where bij-g:⟨bij-betw g (carrier A) (carrier B) ∧ g∈ring-hom A B⟩
  using h1 is-ring-iso-def ring-iso-def by fastforce
obtain h where bij-h:⟨bij-betw h (carrier B) (carrier A) ∧ h∈ring-hom B A ∧
h = the-inv-into (carrier A) g⟩
  using h1 is-ring-iso-def ring-iso-def
by (smt (verit, ccfv-SIG) bij-betwE bij-betw-def bij-betw-the-inv-into bij-g f-the-inv-into-f

    noetherian-ring.axioms(1) ring.ring-simprules(1) ring.ring-simprules(5)
ring.ring-simprules(6)
    ring-hom-add ring-hom-memI ring-hom-mult ring-hom-one the-inv-into-f-f)
from bij-g have f0:⟨ideal I A ⟹ ideal (g ‘ I) B⟩ for I
  using h1 img-ideal-is-ideal noetherian-ring-def ring-iso-def by fastforce

```

```

from bij-h have f2:⟨ideal I B ⟹ ideal (h ‘ I) A⟩ for I
  using h1 img-ideal-is-ideal noetherian-ring-def ring-iso-def by fastforce
then obtain g' where jj1:⟨g' = the-inv-into (carrier A) (g)⟩
  by blast
then have f1:⟨∀ a∈carrier A. ∀ b∈carrier B. g (g' b) = b ∧ g' (g a) = a⟩
  by (meson bij-betw-def bij-g f-the-inv-into-f-bij-betw the-inv-into-f-f)
then have ⟨∃ f'. bij-betw f' {I. ideal I A} {I. ideal I B}⟩
  apply(intro exI[where x=⟨(‘) g⟩])
  apply(rule bij-betw-byWitness[where f'=⟨(‘) h⟩])
  unfolding image-def apply(safe)
  using jj1 bij-h h1 ideal.Icarr ring.ring-simprules(6) apply fastforce
  using jj1 additive-subgroup.a-subset bij-h h1 ideal.axioms(1) ring.ring-simprules(6)
apply fastforce
  apply (metis bij-betwE bij-h ideal.Icarr jj1)
  using bij-g bij-h f-the-inv-into-f-bij-betw ideal.Icarr apply fastforce
  apply(fold image-def)
  using f0 apply presburger
  using f2 by presburger
then have f5:⟨∀ J∈{I. ideal I A}. h ‘ g ‘ J = J ∧ (∀ J∈{I. ideal I B}. g ‘ h ‘ J
= J)⟩
  unfolding image-def apply(safe)
  apply (metis bij-betw-def bij-g bij-h ideal.Icarr the-inv-into-f-f)
  apply (smt (verit, best) bij-betwE bij-g bij-h f1 ideal.Icarr jj1 mem-Collect-eq)
  apply (metis bij-g bij-h f-the-inv-into-f-bij-betw ideal.Icarr)
  by (metis (mono-tags, lifting) bij-g bij-h f-the-inv-into-f-bij-betw ideal.Icarr
mem-Collect-eq)
then have ⟨∀ c∈C. ideal (h ‘ c) A⟩
  unfolding subset.chain-def
  by (metis f2 h3 mem-Collect-eq subset-chain-def subset-eq)
then have inv-imp2:⟨∀ J∈{J. ideal J (B)}. (g ‘ h ‘ J) = J⟩
  by (metis f5 f2 mem-Collect-eq)
then have sub1:⟨subset.chain {Ia. ideal Ia (A)} ((λx. h ‘ x) ‘ C)⟩
  unfolding subset-chain-def image-def apply(safe)
  apply (metis ⟨∀ c∈C. ideal (h ‘ c) A⟩ image-def)
  by (metis (no-types, lifting) h3 subsetD subset-chain-def)
have sub2 :⟨((λx. h ‘ x) ‘ C) ≠ {}⟩
  using h2 by blast
then have f10:⟨(⋃((λx. h ‘ x) ‘ C)) ∈ ((λx. h ‘ x) ‘ C)⟩
  by (meson h1 noetherian-ring.ideal-chain-is-trivial sub1)
then have f9:⟨g ‘ (⋃((λx. h ‘ x) ‘ C)) = (⋃ C)⟩
  apply(safe)
  apply (metis UnionI additive-subgroup.a-Hcarr bij-h f1 h1 h3 ideal.axioms(1)
jj1
  mem-Collect-eq noetherian-ring-def ring.ring-simprules(6) subset.chain-def
subsetD)
  by (smt (verit, del-insts) UN-iff h3 image-def inv-imp2 mem-Collect-eq subsetD
subset-chain-def)
show ⟨⋃ C ∈ C⟩
  by (smt (verit, best) f10 f9 h3 image-iff in-mono inv-imp2 subset.chain-def)

```

qed

**lemma** (in domain) subring:⟨subring (carrier R) R⟩  
using carrier-is-subring by auto

### 4.3 Some properties of the polynomial rings regarding ideals and quotients

**lemma** (in domain) gen-is-cgen:⟨genideal ((carrier R)[X]) {X} = cgenideal ((carrier R)[X]) X⟩  
by (simp add: cring.cgenideal-eq-genideal domain.univ-poly-is-cring domain-axioms subring var-closed(1))

**lemma** (in domain) principal-X:⟨principalideal (genideal ((carrier R)[X]) {X}) ((carrier R)[X])⟩  
apply(subst gen-is-cgen)  
by (simp add: cring.cgenideal-is-principalideal domain.univ-poly-is-cring domain-axioms subring var-closed(1))

**named-theorems** poly

**lemma** (in ring) PIdl-X[poly]:  
⟨cgenideal ((carrier R)[X]) X = { $a_{\otimes_{(\text{carrier } R) [X]} X}$  |  $a. a \in \text{carrier}((\text{carrier } R)[X])$ }⟩  
unfolding cgenideal-def by(auto)

**lemma** (in domain) Idl-X[poly]:  
⟨genideal ((carrier R)[X]) {X} = { $a_{\otimes_{(\text{carrier } R) [X]} X}$  |  $a. a \in \text{carrier}((\text{carrier } R)[X])$ }⟩  
using PIdl-X gen-is-cgen by argo

**lemma** (in domain) Idl-X-is-X[poly]:  
⟨ $p \in \text{genideal}((\text{carrier } R)[X]) \{X\} \implies \exists a \in \text{carrier}((\text{carrier } R)[X]). p = a_{\otimes_{(\text{carrier } R) [X]} X}$ ⟩  
using gen-is-cgen Idl-X by auto

**lemma** (in ring) degree-of-nonempty-p[poly]:⟨ $a \in \text{carrier}((\text{carrier } R)[X]) \wedge a \neq [] \implies \text{coeff } a (\text{degree } a) \neq \mathbf{0}$ ⟩  
by (metis lead-coeff-simp polynomial-def univ-poly-carrier)

**lemma** (in domain) coeff-0-of-mult-X[poly]:⟨ $a \in \text{carrier}((\text{carrier } R)[X]) \implies \text{coeff } (a_{\otimes_{(\text{carrier } R) [X]} X}) 0 = \mathbf{0}$ ⟩  
apply(cases ⟨ $a = []$ ⟩)  
apply (simp add: domain.poly-mult-var domain-axioms subring univ-poly-zero-closed)  
apply(induct a)  
using coeff.simps(1) poly-mult.simps(1)  
apply (simp add: univ-poly-mult)  
by (simp add: append-coeff poly-mult-var subring)

**lemma** (in domain) zero-coeff-of-Idl-X[poly]:⟨ $p \in \text{genideal}((\text{carrier } R)[X]) \{X\} \implies$ ⟩

$\text{coeff } p \ 0 = \mathbf{0}$   
**using** *Idl-X coeff-0-of-mult-X* **by** *auto*

**lemma** (**in** *domain*) *mult-X-append-0*[*poly*]:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies p \neq [] \implies \text{poly-mult } p \ X = p @ [\mathbf{0}] \rangle$   
**using** *poly-mult-var*[*of*  $\langle (\text{carrier } R) \rangle p$ ]  
**by**(*auto simp add: poly-mult-var'(2) polynomial-incl subring univ-poly-carrier univ-poly-mult*)

**lemma** (**in** *ring*) *polynomial-incl'*:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies \text{set } p \subseteq (\text{carrier } R) \rangle$  **for**  $p$   
**unfolding** *univ-poly-def*  
**using** *polynomial-incl* **by** *auto*

**lemma** (**in** *ring*) *hd-in-carrier*:  $\langle p \neq [] \implies p \in \text{carrier}((\text{carrier } R)[X]) \implies \text{hd } p \in (\text{carrier } R) \rangle$  **for**  $p$   
**using** *polynomial-incl'* **unfolding** *univ-poly-def*  
**using** *list.set-sel(1)* **by** *blast*

**lemma** (**in** *ring*) *inv-in-carrier*:  
 $\langle p \neq [] \implies p \in \text{carrier}((\text{carrier } R)[X]) \implies (\text{inv}_{\text{add-monoid } R} (\text{hd } p)) \in (\text{carrier } R) \rangle$   
**for**  $p$   
**using** *hd-in-carrier* **by** *simp*

**lemma** (**in** *ring*) *inv-ld-coeff*:  
 $\langle p \neq [] \implies p \in \text{carrier}((\text{carrier } R)[X]) \implies (\text{inv}_{\text{add-monoid } R} (\text{hd } p) \# \text{replicate } (\text{degree } p) \ \mathbf{0}) \in \text{carrier}((\text{carrier } R)[X]) \rangle$   
**for**  $p$   
**using** *inv-in-carrier* **by** (*metis a-inv-def add.inv-eq-1-iff hd-in-carrier list.sel(1) local.monom-def monom-in-carrier polynomial-def univ-poly-carrier*)

**lemma** (**in** *ring*) *take-in-RX*:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies n \leq \text{length } p \implies (\text{set } (\text{take } n \ p)) \subseteq (\text{carrier } R) \rangle$  **for**  $p \ n$   
**using** *set-take-subset*[*of*  $n \ p$ ] *polynomial-incl'* **by** *blast*

**lemma** (**in** *ring*) *normalize-take-is-poly*:  
 $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies n \leq \text{length } p \implies \text{normalize } (\text{take } n \ p) \in \text{carrier}((\text{carrier } R)[X]) \rangle$  **for**  $n \ p$   
**using** *take-in-RX* **by** (*meson normalize-gives-polynomial univ-poly-carrier*)

**lemma** (**in** *ring*) *normalize-take-is-take*:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \wedge n \leq \text{length } p \implies \text{normalize } (\text{take } n \ p) = \text{take } n \ p \rangle$   
**by** (*metis bot-nat-0.not-eq-extremum degree-of-nonempty-p hd-take lead-coeff-simp normalize.elims normalize.simps(1) take-eq-Nil*)

**lemma** (in ring) take-in-carrier:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies n \leq \text{length } p \implies (\text{take } n \text{ } p) \in \text{carrier}((\text{carrier } R)[X]) \rangle$

using normalize-take-is-poly normalize-take-is-take by force

**lemma** (in domain) take-misc-poly:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies p \neq [] \implies \text{coeff } p \ 0 = \mathbf{0} \implies ((\text{take } (\text{degree } p) \ p)) \otimes_{(\text{carrier } R) [X]} X = p \rangle$  for  $p$

apply(unfold univ-poly-mult)

apply(cases  $\langle p = [] \rangle$ )

subgoal by(simp)

apply(subst mult-X-append-0)

apply(simp add: normalize-take-is-poly univ-poly-carrier)

using normalize-take-is-poly normalize-take-is-take apply force

using degree-of-nonempty-p normalize-take-is-take apply force

by (metis One-nat-def Suc-pred coeff-nth diff-Suc-eq-diff-pred diff-less le-refl

length-greater-0-conv less-one take-Suc-conv-app-nth take-all)

**lemma** (in ring) length-geq-2:  $\langle \text{normalize } p \neq [] \wedge \neg(\exists a. \text{normalize } p = [a]) \implies \text{length } p \geq 2 \rangle$  for  $p$ :  $\langle 'a \text{ list} \rangle$

apply(induct p)

using not-less-eq-eq

by (auto split: if-splits)

**lemma** (in ring) norm-take-not-mt:  $\langle \text{length } (\text{normalize } p) \geq 2 \implies \text{normalize } (\text{take } (\text{degree } p) \ p) \neq [] \rangle$  for  $p$ :  $\langle 'a \text{ list} \rangle$

using length-geq-2

apply(induct p rule: normalize.induct)

apply simp

using One-nat-def Suc-eq-plus1 Suc-le-lessD list.sel(3) list.size(3)

list.size(4) nat-less-le normalize.elims numeral-2-eq-2 take-Cons' take-eq-Nil

by (smt (z3) length-tl list.sel(1) normalize.simps(2))

**lemma** (in ring) normalize-take-invariant:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies p \neq [] \implies (\text{normalize } (\text{take } (\text{degree } p) \ p)) \otimes_{[\text{coeff } p \ 0]} = p \rangle$

for  $p$

apply(subst normalize-take-is-take)

apply simp

by (metis One-nat-def Suc-pred coeff-nth diff-Suc-eq-diff-pred diff-less le-refl

length-greater-0-conv less-one take-Suc-conv-app-nth take-all)

**lemma** (in domain) lower-coeff-add:  $\langle p \neq [] \implies p \in \text{carrier}((\text{carrier } R)[X]) \wedge b \in (\text{carrier } R) \implies \text{coeff } (((\text{normalize } p) \ @[\mathbf{0}]) \oplus_{(\text{carrier } R) [X]} [b]) = \text{coeff } ((\text{normalize } p) \ @[b]) \rangle$

for  $p \ b$

unfolding univ-poly-add

apply(subst poly-add-coeff)

apply (metis local.ring-axioms mult-X-append-0 normalize-polynomial ring.poly-mult-in-carrier

ring.polynomial-in-carrier subring univ-poly-carrier var-closed(2))

by(auto simp add: fun-eq-iff append-coeff polynomial-incl' normalize-polynomial

*univ-poly-carrier*)

**lemma** (in ring) *cons-in-RX*:  $\langle a @ p \in \text{carrier}((\text{carrier } R)[X]) \implies \text{normalize } p \in \text{carrier}((\text{carrier } R)[X]) \rangle$

**proof** –

assume *h1*:  $\langle a @ p \in \text{carrier}((\text{carrier } R)[X]) \rangle$

then have  $\langle \text{set } (a @ p) \subseteq (\text{carrier } R) \rangle$

using *polynomial-incl'* by *presburger*

then have  $\langle \text{set } p \subseteq (\text{carrier } R) \rangle$

by *simp*

then show *?thesis*

using *normalize-gives-polynomial univ-poly-carrier* by *blast*

qed

**lemma** (in ring) *p-in-norm*:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies \text{normalize } p = p \rangle$

by (*simp add: normalize-polynomial univ-poly-carrier*)

**lemma** (in domain) *lower-coeff-add'*:  $\langle p \neq [] \implies p \in \text{carrier}((\text{carrier } R)[X]) \wedge b \in (\text{carrier } R) \implies (((\text{normalize } p) @ [0]) \oplus_{(\text{carrier } R)} [X] [b]) = ((\text{normalize } p) @ [b]) \rangle$

for *p b*

**proof** –

interpret *kcr*: *cring* (*carrier R*)[*X*]

using *carrier-is-subring univ-poly-is-cring* by *auto*

assume *h1*:  $\langle p \neq [] \rangle \langle p \in \text{carrier}((\text{carrier } R)[X]) \wedge b \in (\text{carrier } R) \rangle$

have *f0*:  $\langle b \neq 0 \implies \text{polynomial } (\text{carrier } R) p \wedge \text{polynomial } (\text{carrier } R) [b] \rangle$

by (*metis h1(2) insert-subset polynomial-incl' list.sel(1) list.simps(15) polynomial-def univ-poly-carrier*)

with *h1* show *?thesis*

apply (*cases*  $\langle b = 0 \rangle$ )

apply (*metis append-self-conv2 domain.mult-X-append-0 domain-axioms kcr.r-zero kcr.zero-closed*)

*polynomial-incl' p-in-norm poly-add-append-zero poly-mult-var'(2) univ-poly-add univ-poly-zero*)

unfolding *univ-poly-add* apply (*subst coeff-iff-polynomial-cond*[of  $\langle (\text{carrier } R) \rangle$ ])

apply (*metis polynomial-incl' mult-X-append-0 normalize-polynomial poly-add-closed poly-mult-is-polynomial subring var-closed(1)*)

apply (*metis (mono-tags, lifting) Un-insert-right append-Nil2 hd-append2 insert-subset*)

*list.simps(15) normalize-polynomial polynomial-def set-append*)

by (*metis lower-coeff-add univ-poly-add*)

qed

**lemma** (in domain) *poly-invariant*:  $\langle p \in \text{carrier}((\text{carrier } R)[X]) \implies p \neq [] \implies ((\text{normalize } (\text{take } (\text{degree } p) p)) \otimes_{(\text{carrier } R)} [X]^X) \oplus_{(\text{carrier } R)} [X] [\text{coeff } p 0] = p \rangle$

for *p*

**proof** –

interpret *kcr*: *cring* (*carrier R*)[*X*]

using *carrier-is-subring univ-poly-is-cring* by *auto*

assume *h1*:  $\langle p \in \text{carrier } (\text{poly-ring } R) \rangle \langle p \neq [] \rangle$

**with** *h1 show ?thesis*  
**using** *take-misc-poly apply (cases ⟨p=⟦⟩) apply (simp)*  
**apply** (*cases ⟨∃ a. p=⟦a⟧⟩*)  
**apply** (*metis One-nat-def diff-is-0-eq' kcr.l-zero le-refl lead-coeff-simp length-Cons*  
*list.sel(1) list.size(3) normalize.simps(1) poly-mult.simps(1) take0 univ-poly-mult*  
*univ-poly-zero)*  
**unfolding** *univ-poly-mult*  
**apply** (*subst mult-X-append-0*)  
**using** *diff-le-self normalize-take-is-poly apply presburger*  
**using** *length-geq-2[of p] norm-take-not-mt[of p]*  
**apply** (*metis coeff-iff-length-cond degree-of-nonempty-p lead-coeff-simp nor-*  
*malize-coeff normalize-length-eq)*  
**by** (*metis (no-types, lifting) append.right-neutral append-self-conv2 coeff-in-carrier*  
*diff-le-self polynomial-incl' normalize-take-invariant lower-coeff-add' normal-*  
*ize-take-is-poly local.normalize-idem)*  
**qed**

**lemma** (*in domain*) *gen-ideal-X-iff:⟨p∈(genideal ((carrier R)[X]) {X}) ⟷ (p∈carrier*  
*((carrier R)[X]) ∧ coeff p 0 = 0)⟩* **for** *p::⟨'a list⟩*  
**using** *poly take-misc-poly apply (safe)*  
**using** *domain.univ-poly-is-ring domain-axioms monoid.m-closed ring-def subring*  
*var-closed(1)*  
**apply** (*metis (no-types, lifting)*)  
**apply** (*meson domain.univ-poly-is-ring domain-axioms monoid.m-closed ring-def*  
*subring var-closed(1)*)  
**by** (*smt (verit, ccfv-threshold) mem-Collect-eq nat-le-linear poly-mult.simps(1)*  
*take-all*  
*take-in-carrier univ-poly-mult)*

**lemma** (*in domain*) *gen-ideal-X-iff':⟨(genideal ((carrier R)[X]) {X}) = {p∈carrier*  
*((carrier R)[X]). coeff p 0 = 0}⟩* **for** *p::⟨'a list⟩*  
**using** *gen-ideal-X-iff by auto*

**lemma** (*in domain*) *quot-X-is-R:⟨carrier (((carrier R)[X]) Quot (genideal ((carrier*  
*R)[X]) {X}))*  
*= {⟨x∈carrier((carrier R)[X]). coeff x 0 = a⟩ | a. a∈(carrier R)⟩*  
**proof** (*subst set-eq-subset, safe*)  
**interpret** *kcr:cring (carrier R)[X]*  
**using** *carrier-is-subring univ-poly-is-cring by auto*  
**fix** *x*  
**assume** *h1:⟨x ∈ carrier ((carrier R) [X] Quot (genideal ((carrier R)[X]) {X}))⟩*  
**have** *l0:⟨as≠[] ⟹ take (length as) (a#as) = a#take (degree as) as⟩* **for** *a::'a*  
**and** *as*  
**by** (*simp add: take-Cons'*)  
**have** *rule-U:⟨xaa ∈ (⋃ x∈Idl<sub>poly-ring</sub> R {X}. {x ⊕<sub>poly-ring</sub> R xa})*  
*= (∃ x∈Idl<sub>poly-ring</sub> R {X}. xaa = x ⊕<sub>poly-ring</sub> R xa)⟩*



```

for xaa xa
by auto
from h1 have ⟨∃ xa ∈ carrier (poly-ring R). x = (⋃ x ∈ Idlpoly-ring R {X}. {x
⊕poly-ring R xa})⟩
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def by simp
  with h1 show ⟨∃ a. x = {x ∈ carrier (poly-ring R). local.coeff x 0 = a} ∧ a ∈
carrier R⟩
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def
  proof (safe, fold FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def )
  fix xa
  assume h1:⟨(⋃ x ∈ Idlpoly-ring R {X}. {x ⊕poly-ring R xa})
    ∈ carrier (poly-ring R Quot Idlpoly-ring R {X})⟩
    ⟨xa ∈ carrier (poly-ring R)⟩
    ⟨x = (⋃ x ∈ Idlpoly-ring R {X}. {x ⊕poly-ring R xa})⟩
    ⟨x ∈ carrier (poly-ring R Quot Idlpoly-ring R {X})⟩
    ⟨∃ xa ∈ carrier (poly-ring R). x = (⋃ x ∈ Idlpoly-ring R {X}. {x ⊕poly-ring R
xa})⟩
  show ⟨∃ a. (⋃ x ∈ Idlpoly-ring R {X}. {x ⊕poly-ring R xa}) =
    {x ∈ carrier (poly-ring R). local.coeff x 0 = a} ∧
    a ∈ carrier R⟩
  proof (rule exI[where x = ⟨coeff xa 0⟩], safe)
  fix x' xaa
  assume h2:⟨xaa ∈ Idlpoly-ring R {X}⟩
  with h1 show ⟨xaa ⊕poly-ring R xa ∈ carrier (poly-ring R)⟩
    unfolding FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def
    using Idl-X subring var-closed(1) by auto[1]
  show ⟨local.coeff (xaa ⊕poly-ring R xa) 0 = local.coeff xa 0⟩
  apply (insert h1 h2)
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def
  using Idl-X subring var-closed(1) apply (safe)
  apply (frule coeff-0-of-mult-X)
  apply (frule zero-coeff-of-Idl-X)
  apply (subst coeffs-of-add-poly)
  using gen-ideal-X-iff apply blast
  apply blast
  by (simp add: polynomial-incl univ-poly-carrier)
next
fix y
assume h2:⟨y ∈ carrier (poly-ring R)⟩
  ⟨local.coeff y 0 = local.coeff xa 0⟩
with h1 show ⟨y ∈ (⋃ x ∈ Idlpoly-ring R {X}. {x ⊕poly-ring R xa})⟩
  apply (subst rule-U)
  apply (rule beXI[where x = ⟨y ⊕(carrier R) [X]⟩ (inv add-monoid ((carrier R)[X])
xa)⟩))
  apply (metis a-inv-def kcr.add.inv.solve-right' kcr.minus-closed kcr.minus-eq)
by (metis a-inv-def coeff.simps(1) coeffs-of-add-poly gen-ideal-X-iff kcr.add.inv-closed
kcr.add.inv.solve-right kcr.add.m-closed kcr.add.m-lcomm

```

```

      kcr.r-zero kcr.zero-closed univ-poly-zero)
next
  from h1 show ⟨local.coeff xa 0 ∈ carrier R⟩
    by (simp add: polynomial-incl univ-poly-carrier)
qed
qed
next
interpret kcr:cring (carrier R)[X]
  using carrier-is-subring univ-poly-is-cring by auto
fix a
assume h1:⟨a ∈ (carrier R)⟩
have p-h1:⟨a≠0 ⇒ [a] ∈ carrier ((carrier R)[X])⟩
  by (metis Diff-iff const-is-polynomial empty-iff h1 insert-iff univ-poly-carrier)
have rule-s:⟨{x ∈ carrier (poly-ring R). local.coeff x 0 = a} ∈ carrier (poly-ring
R Quot Idlpoly-ring R {X}) =
(∃ x∈carrier (poly-ring R).
  {x ∈ carrier (poly-ring R). local.coeff x 0 = a} =
  (⋃ xa∈Idlpoly-ring R {X}. {xa ⊕poly-ring R x}) )⟩
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def by(auto)
show ⟨{x ∈ carrier (poly-ring R). local.coeff x 0 = a}
  ∈ carrier (poly-ring R Quot Idlpoly-ring R {X})⟩
  apply(subst rule-s)
  apply(cases ⟨a=0⟩)
  apply(rule bexI[where x=⟨[]⟩])
  apply(subst Idl-X) apply(safe)[1]
    apply (metis (no-types, lifting) PIdl-X UN-iff gen-ideal-X-iff gen-is-cgen
      insert-iff kcr.r-zero univ-poly-zero)
  using subring var-closed(1) apply force
  apply (metis coeff-0-of-mult-X kcr.m-closed kcr.r-zero subring univ-poly-zero
var-closed(1))
  apply blast
  apply(rule bexI[where x=⟨[a]⟩])
  apply(subst Idl-X)
  apply(safe)
  apply(simp)
  apply (metis poly-invariant coeff.simps(1) diff-le-self normalize-take-is-poly)
  using h1 subring var-closed(1) p-h1 apply(auto)[1]
  apply (metis coeffs-of-add-poly diff-Suc-1 domain.coeff-0-of-mult-X domain.poly-mult-var

      domain-axioms kcr.l-zero kcr.m-closed kcr.zero-closed lead-coeff-simp length-Cons
      list.distinct(1) list.sel(1) list.size(3) p-h1 subring univ-poly-zero var-closed(1))
  using p-h1 by auto
qed

lemma (in domain) uniq-a-quot:
  ⟨c∈ carrier (((carrier R)[X]) Quot (genideal ((carrier R)[X]) {X})) ⇒ ∃!a∈(carrier
R). ∀ y∈c. coeff y 0 = a⟩
proof(subst (asm) quot-X-is-R, safe)
  fix a

```

```

assume  $h1:\langle a \in \text{carrier } R \rangle \langle c = \{x \in \text{carrier } (\text{poly-ring } R). \text{local.coeff } x \ 0 = a\} \rangle$ 
then show  $\langle \exists aa. aa \in \text{carrier } R \wedge$ 
     $(\forall y \in \{x \in \text{carrier } (\text{poly-ring } R). \text{local.coeff } x \ 0 = a\}. \text{local.coeff } y \ 0 =$ 
 $aa) \rangle$ 
  apply(intro exI[where x=a])
  by fastforce
next
  fix  $a \ aa \ y$ 
  assume  $h1:\langle a \in \text{carrier } R \rangle \langle c = \{x \in \text{carrier } (\text{poly-ring } R). \text{local.coeff } x \ 0 = a\} \rangle$ 
 $\langle aa \in \text{carrier } R \rangle$ 
   $\langle \forall y \in \{x \in \text{carrier } (\text{poly-ring } R). \text{local.coeff } x \ 0 = a\}. \text{local.coeff } y \ 0 = aa \rangle \langle y \in$ 
 $\text{carrier } R \rangle$ 
   $\langle \forall ya \in \{x \in \text{carrier } (\text{poly-ring } R). \text{local.coeff } x \ 0 = a\}. \text{local.coeff } ya \ 0 = y \rangle$ 
  have  $\langle \{x \mid x. x \in \text{carrier } ((\text{carrier } R) [X]) \wedge \text{local.coeff } x \ 0 = a\} \neq \{\} \rangle$ 
  apply(subst ex-in-conv[symmetric]) apply(cases <a=0>)
  apply(rule exI[where x=⟨[]⟩)
  apply(fastforce)
  apply(rule exI[where x=⟨[a]⟩)
  using  $h1(1)$  apply(safe)
  apply(rule exI[where x=⟨[a]⟩) apply(simp)
  by (metis empty-subsetI insert-subset list.sel(1))
    list.simps(15) polynomialI set-empty univ-poly-carrier)
  then show  $\langle aa = y \rangle$ 
    using  $h1(4) \ h1(6)$  all-not-in-conv[of <{x | x. x ∈ carrier (poly-ring R) ∧ local.coeff x 0 = a}>]
    by (metis (no-types, lifting))
qed

```

```

lemma (in ring) append-in-carrier:  $\langle a \in \text{carrier } ((\text{carrier } R) [X]) \wedge b \in \text{carrier } ((\text{carrier } R) [X]) \implies a @ b \in \text{carrier } ((\text{carrier } R) [X]) \rangle$ 
apply(induct b arbitrary:a)
by (metis append-self-conv2 hd-append2 le-sup-iff mem-Collect-eq)
  partial-object.select-convs(1) polynomial-def set-append univ-poly-def+)

```

```

lemma (in domain) The-a-is-a:  $\langle a \in (\text{carrier } R) \implies$ 
 $(\text{THE } aa. \forall y \in \{x \mid x. x \in \text{carrier } ((\text{carrier } R) [X]) \wedge \text{local.coeff } x \ 0 = a\}. \text{local.coeff } y \ 0 = aa) = a \rangle$ 
proof –
  assume  $h1:\langle a \in (\text{carrier } R) \rangle$ 
  have  $\langle \exists c \in \text{carrier } (((\text{carrier } R) [X]) \text{Quot } (\text{genideal } ((\text{carrier } R) [X]) \{X\})) .$ 
     $c = \{x \mid x. x \in \text{carrier } ((\text{carrier } R) [X]) \wedge \text{local.coeff } x \ 0 = a\} \rangle$ 
  apply(subst quot-X-is-R)
  using  $h1$  by auto
  then obtain  $c$  where  $f0:\langle c = \{x \mid x. x \in \text{carrier } ((\text{carrier } R) [X]) \wedge \text{local.coeff } x \ 0 = a\}$ 
 $\wedge c \in \text{carrier } (((\text{carrier } R) [X]) \text{Quot } (\text{genideal } ((\text{carrier } R) [X]) \{X\})) \rangle$ 
  by blast
  then have  $\langle (\text{THE } aa. \forall y \in c. \text{local.coeff } y \ 0 = aa) = a \rangle$ 

```

**by** (*smt* (*verit*, *best*) *coeff.simps(1)* *h1 mem-Collect-eq theI uniq-a-quot univ-poly-zero-closed zero-closed*)  
**then show** *?thesis*  
**by** (*simp add:f0*)  
**qed**

**lemma** (*in ring*) *poly-mult-in-carrier2*:  
 $\llbracket \text{set } p1 \subseteq \text{carrier } R; \text{ set } p2 \subseteq \text{carrier } R \rrbracket \implies \text{poly-mult } p1 \ p2 \in \text{carrier } ((\text{carrier } R)[X])$   
**using** *poly-mult-is-polynomial polynomial-in-carrier carrier-is-subring*  
**by** (*simp add: univ-poly-carrier*)

**lemma** (*in ring*) *normalize-equiv*:  $\langle \text{polynomial } (\text{carrier } R) (\text{normalize } p) \longleftrightarrow (\text{coeff } (\text{normalize } p)) \in \text{carrier } (UP\ R) \rangle$   
**proof** (*safe*)  
**interpret** *UP-r*: *UP-ring R UP R*  
**by** (*simp add: UP-ring-def local.ring-axioms*) +  
**assume**  $\langle \text{polynomial } (\text{carrier } R) (\text{normalize } p) \rangle$   
**then show**  $\langle \text{coeff } (\text{normalize } p) \in \text{carrier } (UP\ R) \rangle$   
**by** (*meson carrier-is-subring coeff-degree poly-coeff-in-carrier UP-r.UP-car-memI*)  
**next**

**interpret** *UP-r*: *UP-ring R UP R*  
**by** (*simp add: UP-ring-def local.ring-axioms*) +  
**assume**  $\langle \text{coeff } (\text{normalize } p) \in \text{carrier } (UP\ R) \rangle$   
**then show**  $\langle \text{polynomial } (\text{carrier } R) (\text{normalize } p) \rangle$   
**unfolding** *polynomial-def UP-r.P-def UP-def apply* (*safe*)  
**using** *coeff-img-restrict*[*of*  $\langle (\text{normalize } p) \rangle$ ] *imageE*[*of* -  $\langle \text{coeff } (\text{normalize } p) \rangle$ ] ]  
*mem-upD*[*of*  $\langle \text{coeff } (\text{normalize } p) \rangle$ ] *partial-object.select-convs(1)*  
**apply** (*metis* (*no-types*, *lifting*))  
**by** (*meson ring-axioms polynomial-def ring.normalize-gives-polynomial subsetI*)  
**qed**

**lemma** (*in ring*) *p-in-RX-imp-in-P*:  $\langle p \in \text{carrier } ((\text{carrier } R)[X]) \implies \text{coeff } p \in \text{up } R \rangle$   
**by** (*meson bound.intro coeff-in-carrier coeff-length linorder-not-less mem-upI nat-le-linear polynomial-incl'*)

**lemma** (*in ring*) *X-has-correp*:  $\langle \text{coeff } X = (\lambda i. \text{if } i = 1 \text{ then } \mathbf{1} \text{ else } \mathbf{0}) \rangle$   
**unfolding** *var-def* **by** (*auto*)

**lemma** (*in ring*) *mult-is-mult*:  
 $\langle \{x, y\} \subseteq \text{carrier } ((\text{carrier } R)[X]) \implies \text{coeff } (x \otimes_{(\text{carrier } R)[X]} y) = \text{coeff } x \otimes_{UP\ R} \text{coeff } y \rangle$   
**proof** –  
**interpret** *UP-r*: *UP-ring R UP R*  
**by** (*simp add: UP-ring-def local.ring-axioms*) +  
**assume** *a1*:  $\{x, y\} \subseteq \text{carrier } ((\text{carrier } R)[X])$

```

then have a2:  $y \in \text{carrier } (\text{poly-ring } R)$   $x \in \text{carrier } (\text{poly-ring } R)$ 
  by auto
then have f3:  $\text{coeff } y \in \text{carrier } (UP\ R)$ 
  by (metis p-in-norm normalize-equiv univ-poly-carrier)
have  $\text{coeff } x \in \text{carrier } (UP\ R)$ 
  using a2 by (metis p-in-norm normalize-equiv univ-poly-carrier)
then show ?thesis
  unfolding univ-poly-mult
  apply(subst poly-mult-coeff)
  apply (simp add: polynomial-incl' a2)+
  unfolding UP-r.P-def UP-def
  using UP-r.p-in-RX-imp-in-P UP-r.UP-ring-axioms a2(1)
  by (simp add: local.ring-axioms ring.p-in-RX-imp-in-P)
qed

```

```

lemma (in ring) add-is-add:  $\langle x \in \text{carrier } (\text{poly-ring } R) \implies$ 
   $y \in \text{carrier } (\text{poly-ring } R)$ 
 $\implies \text{coeff } (x \oplus_{\text{poly-ring } R} y) = \text{coeff } x \oplus_{UP\ R} \text{coeff } y \rangle$ 
proof –
  interpret UP-r: UP-ring R UP R
  by (simp add: UP-ring-def local.ring-axioms)+
  assume a1:  $x \in \text{carrier } (\text{poly-ring } R)$ 
  assume a2:  $y \in \text{carrier } (\text{poly-ring } R)$ 
  then have f3:  $\text{coeff } y \in \text{carrier } (UP\ R)$ 
  by (metis p-in-norm normalize-equiv univ-poly-carrier)
  have  $\text{coeff } x \in \text{carrier } (UP\ R)$ 
  using a1 by (metis p-in-norm normalize-equiv univ-poly-carrier)
  then show ?thesis
  using f3 a2 a1 UP-r.cfs-add[of <coeff x> <coeff y>] coeffs-of-add-poly[of x y] by
presburger
qed

```

#### 4.4 The isomorphisms between the different models of polynomials

```

lemma (in ring) coeff-iso-RX-P:  $\langle \text{coeff} \in \text{ring-iso } (\text{poly-ring } R) (UP\ R) \rangle$ 
proof –
  interpret UP-r: UP-ring R UP R
  by (simp add: UP-ring-def local.ring-axioms)+
  {
  fix  $x$ 
  assume h1:  $\langle x \in \text{carrier } (UP\ R) \rangle$ 
  then obtain  $n::\text{nat}$  where  $\langle \text{bound } \mathbf{0} \ n \ x \rangle$  using UP-r.P-def unfolding UP-def
by auto
  then have  $\langle x \neq (\lambda-. \mathbf{0}) \implies \exists n'. \forall m > n'. x\ m = \mathbf{0} \wedge x\ n' \neq \mathbf{0} \rangle$ 
  by (metis UP-ring.coeff-simp UP-r.UP-ring-axioms UP-r.deg-gtE UP-r.deg-nzero-nzero)
  h1 UP-r.lcoeff-nonzero not-gr-zero
  then obtain  $n'::\text{nat}$  where  $f5: \langle x \neq (\lambda-. \mathbf{0}) \implies \forall m > n'. x\ m = \mathbf{0} \wedge x\ n' \neq \mathbf{0} \rangle$ 

```

```

by blast
  define l: 'a list where l-is:  $l \equiv \text{rev } (\text{map } x [0..< \text{Suc } n'])$ 
  then have  $\langle x \neq (\lambda \cdot \mathbf{0}) \implies \text{normalize } l = l \rangle$ 
    using f5 by (auto)
  from l-is have  $\langle l \neq [] \rangle$ 
    by simp
  then have f6:  $\langle k \leq \text{length } l - 1 \implies \text{coeff } l \ k = l!(\text{length } l - 1 - k) \rangle$  for k
    apply (induct l rule: coeff.induct)
    using coeff-nth diff-diff-left le-neq-implies-less plus-1-eq-Suc by auto
  have gen-ideal-X-iff:  $\langle k \leq \text{length } g - 1 \implies g!k = (\text{rev } g) ! (\text{length } g - 1 - k) \rangle$ 
for g: 'a list and k::nat
  apply (induct g)
  apply force
  by (metis One-nat-def diff-Suc-Suc length-Cons length-rev less-Suc-eq-le minus-nat.diff-0 rev-nth rev-rev-ident)
  then have  $\langle \text{length } l - 1 = n' \rangle$  using l-is by (auto)
  then have f9:  $\langle \forall n \leq n'. x \ n = \text{coeff } l \ n \rangle$ 
    using l-is f6
  by (metis add-0 diff-Suc-Suc diff-diff-cancel diff-less-Suc diff-zero l-is length-map length-upt nth-map-upt rev-nth)
  then have  $\langle \forall n > n'. \text{coeff } l \ n = \mathbf{0} \rangle$ 
    using coeff-degree  $\langle \text{Polynomials.degree } l = n' \rangle$  by blast
  then have f8:  $\langle \forall n > n'. x \ n = \text{coeff } l \ n \rangle$ 
    using f5 by (auto)
  have f10:  $\langle \forall n. x \ n = \text{coeff } l \ n \rangle$ 
    using f8 f9
  by (meson linorder-not-less)
  then have  $\langle \exists xa \in \text{carrier } (\text{poly-ring } R). x = \text{coeff } xa \rangle$ 
    apply (cases  $\langle x = (\lambda \cdot \mathbf{0}) \rangle$ )
    apply (rule beXI[where x= $\langle [] \rangle$ ])
    apply simp
    apply (simp add: univ-poly-zero-closed)
    apply (rule beXI[where x=l])
    apply blast
  by (metis  $\langle x \neq (\lambda \cdot \mathbf{0}) \implies \text{normalize } l = l \rangle$  ext h1 mem-Collect-eq
    normalize-equiv partial-object.select-convs(1) univ-poly-def) } note subg=this
show ?thesis
  unfolding is-ring-iso-def ring-iso-def
  apply (safe)
  subgoal unfolding ring-hom-def apply (safe)
    apply (simp add: local.ring-axioms UP-def ring.p-in-RX-imp-in-P univ-poly-def)

    apply (simp add: mult-is-mult)
    apply (simp add: add-is-add)
    using UP-r.P-def unfolding univ-poly-def UP-def by (simp add: fun-eq-iff)
  unfolding bij-betw-def inj-on-def apply (safe)
    apply (simp add: coeff-iff-polynomial-cond univ-poly-carrier)
  apply (metis normalize-polynomial mem-Collect-eq normalize-equiv partial-object.select-convs(1))

```

```

    univ-poly-def)
  apply(simp add:image-def)
  by(simp add:subg)
qed

lemma (in ring) RX-iso-P:⟨(carrier R)[X] ≃ (UP R)⟩
  unfolding is-ring-iso-def
  using coeff-iso-RX-P by force

lemma (in domain) R-isom-RX-X:⟨R ≃ (((carrier R)[X]) Quot (genideal ((carrier
R)[X]) {X})))⟩
proof(unfold is-ring-iso-def, subst ex-in-conv[symmetric])
  show ⟨∃ x. x ∈ ring-iso R ((carrier R) [X] Quot Idl(carrier R) [X] {X})⟩
  proof(rule exI[where x=⟨λx. {y. y∈carrier((carrier R)[X]) ∧ coeff y 0 = x}⟩],
rule ring-iso-memI)
    fix x
    assume h1:⟨x∈(carrier R)⟩
    then show ⟨{y ∈ carrier ((carrier R) [X]). local.coeff y 0 = x} ∈ carrier
((carrier R) [X] Quot Idl(carrier R) [X] {X})⟩
    using quot-X-is-R by auto
  next
  interpret kcr:cring (carrier R)[X]
    using carrier-is-subring univ-poly-is-cring by auto
  fix x y
  assume h1:⟨x∈(carrier R)⟩ and h2:⟨y∈(carrier R)⟩
  interpret RcR: cring R
    by (simp add: is-cring)
  interpret QcR: cring ⟨(carrier R) [X] Quot Idl(carrier R) [X] {X}⟩
    by (simp add: ideal.quotient-is-cring kcr.genideal-ideal kcr.is-cring subring
var-closed(1))
  have left:⟨x∈(carrier R) ∧ y∈(carrier R) ⟹ x = 0 ⟹
{ya ∈ carrier ((carrier R) [X]). local.coeff ya 0 = x ⊗ y} =
{y ∈ carrier ((carrier R) [X]). local.coeff y 0 = x}
⊗(carrier R) [X] Quot Idl(carrier R) [X] {X} {ya ∈ carrier ((carrier R) [X]). lo-
cal.coeff ya 0 = y}⟩
  if h3:⟨x∈(carrier R) ∧ y∈(carrier R)⟩ for x y
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def rcoset-mult-def r-coset-def
a-r-coset-def
  apply(simp, safe, simp)
  apply (metis Diff-iff One-nat-def coeff.simps(1) const-is-polynomial diff-self-eq-0
empty-iff
gen-ideal-X-iff insert-iff kcr.l-null kcr.r-zero lead-coeff-simp length-Cons
list.distinct(1) list.sel(1)
list.size(3) univ-poly-carrier univ-poly-zero univ-poly-zero-closed )
  using gen-ideal-X-iff apply blast
  unfolding univ-poly-mult univ-poly-add
  apply(frule zero-coeff-of-Idl-X)

```

```

apply(subst (asm) Idl-X)
using h3
by (metis (no-types, lifting) PIdl-X coeffs-of-add-poly gen-ideal-X-iff gen-is-cgen
ideal.I-l-closed
      kcr.cgenideal-ideal kcr.m-comm l-zero subring univ-poly-add univ-poly-mult
var-closed(1))
have right : $\langle y = \mathbf{0} \implies \{ya \in \text{carrier } ((\text{carrier } R) [X]). \text{local.coeff } ya \ 0 = x \otimes y\} =$ 
 $\{y \in \text{carrier } ((\text{carrier } R) [X]). \text{local.coeff } y \ 0 = x\} \otimes_{(\text{carrier } R) [X]} \text{Quot Idl}_{(\text{carrier } R) [X]} \{X\}$ 
 $\langle \{ya \in \text{carrier } ((\text{carrier } R) [X]). \text{local.coeff } ya \ 0 = y\} \rangle$ 
apply(subst m-comm[OF h1 h2])
apply(subst QcR.m-comm)
using h1 quot-X-is-R left h1 by auto
have poly-mult-a-b : $\langle a \in (\text{carrier } R) \wedge b \in (\text{carrier } R) \wedge a \neq \mathbf{0} \wedge b \neq \mathbf{0} \implies \text{poly-mult}$ 
([a]) ([b]) = [a $\otimes$ b] $\rangle$  for a b
using integral-iff by force
have poly-mult-0 : $\langle a \in \text{carrier } ((\text{carrier } R)[X]) \wedge b \in \text{carrier } ((\text{carrier } R)[X]) \implies$ 
coeff (poly-mult a b) 0 = coeff a 0  $\otimes$  coeff b 0 $\rangle$ 
for a b
apply(subst poly-mult-coeff)
by (simp add: polynomial-incl')+
have j0 : $\langle xa \in \text{carrier } (\text{poly-ring } R) \implies \text{local.coeff } xa \ 0 = x \otimes y \implies x \neq \mathbf{0}$ 
 $\implies y \neq \mathbf{0}$ 
 $\implies \exists xb. xb \in \text{carrier } (\text{poly-ring } R) \wedge \text{local.coeff } xb \ 0 = x \wedge (\exists x. x \in \text{carrier}$ 
(poly-ring R)  $\wedge$ 
local.coeff x 0 = y  $\wedge (\exists xc \in \text{Idl}_{\text{poly-ring } R} \{X\}. xa = xc \oplus_{\text{poly-ring } R} xb \otimes_{\text{poly-ring } R}$ 
x) $\rangle$ 
for xa
apply(rule exI[where x= $\langle [x] \rangle$ ])
apply(safe)
subgoal by (metis Diff-iff const-is-polynomial empty-iff h1 insert-iff univ-poly-carrier)
subgoal by simp
apply(rule exI[where x= $\langle [y] \rangle$ ])
apply(safe)
subgoal by (metis Diff-iff const-is-polynomial empty-iff h2 insert-iff univ-poly-carrier)
subgoal by simp
apply(rule beXI[where x= $\langle \text{normalize } (\text{take } (\text{degree } xa) \ xa \ @[\mathbf{0}]) \rangle$ ])
unfolding univ-poly-add univ-poly-mult
apply(subst poly-mult-a-b)
subgoal using h1 h2 by(simp)
subgoal by (metis (no-types, lifting) diff-le-self
domain.coeff-0-of-mult-X domain.m-lcancel domain.poly-mult-var do-
main-axioms h1 h2
poly-invariant take-in-RX normalize-take-is-take poly-mult-var'(2) r-null
subring univ-poly-add
univ-poly-mult zero-closed)
apply(subst Idl-X)
by (metis (no-types, lifting) PIdl-X coeff-0-of-mult-X diff-le-self gen-ideal-X-iff

```



```

gen-is-cgen
  kcr.m-closed take-in-RX poly-mult-var'(2) subring take-in-carrier univ-poly-mult
var-closed(1))
  show fst:⟨{ya ∈ carrier ((carrier R) [X]). local.coeff ya 0 = x ⊗ y} =
    {y ∈ carrier ((carrier R) [X]). local.coeff y 0 = x} ⊗(carrier R) [X] Quot Idl(carrier R) [X] {X}
    {ya ∈ carrier ((carrier R) [X]). local.coeff ya 0 = y}⟩
  proof(safe)
    fix xa
    assume h1:⟨xa ∈ carrier (poly-ring R)⟩ ⟨local.coeff xa 0 = x ⊗ y⟩
    then show ⟨xa ∈ {y ∈ carrier (poly-ring R). local.coeff y 0 = x} ⊗poly-ring R Quot Idlpoly-ring R {X}
      {ya ∈ carrier (poly-ring R). local.coeff ya 0 = y}⟩
    apply(cases ⟨x=0 ∨ y=0⟩)
    using h2 left right apply blast
    unfolding FactRing-def A-RCOSETS-def RCOSETS-def rcoset-mult-def
r-coset-def a-r-coset-def
    using j0 by(auto) [1]
  next
  fix xa
  assume h1':⟨xa ∈ {y ∈ carrier (poly-ring R). local.coeff y 0 = x} ⊗poly-ring R Quot Idlpoly-ring R {X}
    {ya ∈ carrier (poly-ring R). local.coeff ya 0 = y}⟩
  then show ⟨xa ∈ carrier (poly-ring R)⟩
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def rcoset-mult-def
r-coset-def a-r-coset-def
  by simp (metis gen-ideal-X-iff kcr.add.m-closed kcr.m-closed univ-poly-add
univ-poly-mult)
  from h1' show ⟨local.coeff xa 0 = x ⊗ y⟩
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def rcoset-mult-def
r-coset-def a-r-coset-def
  apply(simp, safe)
  apply(frule zero-coeff-of-Idl-X)
  apply (simp add: polynomial-incl' domain-axioms gen-ideal-X-iff)
  using polynomial-incl' poly-mult-in-carrier
  by (metis coeffs-of-add-poly h1 h2 kcr.m-closed l-distr l-null l-zero poly-mult-0
univ-poly-mult zero-closed)
  qed
  have poly-add-a-b:⟨a∈(carrier R) ∧ b∈(carrier R) ∧ a≠0 ∧ b≠0 ⇒ poly-add
([a]) ([b]) = normalize [a⊕b]⟩ for a b
  by(auto)
  have is-inv-0:⟨local.normalize [invadd-monoid R y ⊕ y] = []⟩
  by (simp add: h2)
  have poly-add-comm: ⟨{x,y,z}⊆carrier ((carrier R)[X]) ⇒poly-add (poly-add
y z) x = poly-add y (poly-add z x) ⟩ for x y z
  by (metis insert-subset kcr.add.m-assoc univ-poly-add)
  show ⟨{ya ∈ carrier ((carrier R) [X]). local.coeff ya 0 = x ⊕ y} =
    {y ∈ carrier ((carrier R) [X]). local.coeff y 0 = x} ⊕(carrier R) [X] Quot Idl(carrier R) [X] {X}
    {ya ∈ carrier ((carrier R) [X]). local.coeff ya 0 = y}⟩
  proof(safe)
    fix xa

```

```

assume h1':⟨xa ∈ carrier (poly-ring R)⟩⟨local.coeff xa 0 = x ⊕ y ⟩
then show ⟨xa ∈ {y ∈ carrier (poly-ring R). local.coeff y 0 = x} ⊕poly-ring R Quot Idlpoly-ring R {X}
  {ya ∈ carrier (poly-ring R). local.coeff ya 0 = y}⟩
  apply(cases ⟨x=0 ∨ y=0⟩)
    unfolding FactRing-def A-RCOSETS-def RCOSETS-def rcoset-mult-def
r-coset-def a-r-coset-def
  set-add-def set-mult-def apply(simp, safe)[1]
  apply (metis coeff.simps(1) h2 kcr.l-zero l-zero univ-poly-zero univ-poly-zero-closed)
  apply (metis coeff.simps(1) h1 kcr.r-zero r-zero univ-poly-zero univ-poly-zero-closed)
    unfolding FactRing-def A-RCOSETS-def RCOSETS-def rcoset-mult-def
r-coset-def a-r-coset-def
  set-add-def set-mult-def apply(simp)
  apply(rule exI[where x=⟨xa ⊕(carrier R) [X] [inv-add-monoid R y]⟩])
  apply(safe)
    apply (metis a-inv-def add.Units-eq add.Units-inv-closed add.inv-eq-1-iff
h2 insert-subset
  kcr.add.m-closed list.sel(1) list.simps(15) polynomial-def polynomial-incl
univ-poly-carrier)
    apply (metis (no-types, lifting) a-assoc add.Units-eq add.Units-inv-closed
add.Units-r-inv
  coeffs-of-add-poly diff-Suc-1 h1 h2 insert-subset polynomial-incl' lead-coeff-simp
length-Cons list.distinct(1) list.sel(1) list.simps(15) list.size(3) mem-Collect-eq par-
tial-object.select-convs(1) polynomial-def r-zero univ-poly-def)
    apply(rule exI[where x=⟨[y]⟩])
    apply(safe) apply(simp add:h2 univ-poly-def polynomial-def)
    apply(simp)
    apply(cases xa)
    unfolding univ-poly-add
    using add.Units-eq add.inv-eq-one-eq add.Units-inv-closed add.Units-l-inv
h2 r-zero apply(auto)[1]
    apply(subst poly-add-comm)
    apply (metis Diff-iff One-nat-def append.right-neutral const-is-polynomial
diff-self-eq-0
  empty-iff empty-subsetI h2 insert-iff insert-subset inv-ld-coeff length-Cons
list.distinct(1) list.sel(1)
  list.size(3) normalize.simps(1) normalize-trick univ-poly-carrier)
    apply(subst poly-add-a-b)
    apply(simp add:h2 add.inv-eq-one-eq)
    apply(subst is-inv-0)
    by (metis polynomial-incl' p-in-norm poly-add-zero'(1))
next
  fix xa
  assume h1':⟨xa ∈ {y ∈ carrier (poly-ring R). local.coeff y 0 = x} ⊕poly-ring R Quot Idlpoly-ring R {X}
    {ya ∈ carrier (poly-ring R). local.coeff ya 0 = y} ⟩
  then show ⟨xa ∈ carrier (poly-ring R)⟩
    unfolding FactRing-def A-RCOSETS-def RCOSETS-def rcoset-mult-def
r-coset-def a-r-coset-def
    set-add-def set-mult-def by(auto)

```

```

from h1' show ⟨local.coeff xa 0 = x ⊕ y⟩
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def rcoset-mult-def
r-coset-def a-r-coset-def
  set-add-def set-mult-def using polynomial-incl' poly-add-coeff coeffs-of-add-poly
by auto
  qed
next
  interpret kcr:cring (carrier R)[X]
  using carrier-is-subring univ-poly-is-cring by auto
  show ⟨y ∈ carrier ((carrier R) [X]). local.coeff y 0 = 1⟩ = 1(carrier R) [X] Quot Idl(carrier R) [X] {X}⟩
  unfolding FactRing-def a-r-coset-def r-coset-def
  using gen-ideal-X-iff apply(simp, safe, simp)
  apply (metis (no-types, lifting) diff-le-self domain.coeff-0-of-mult-X
domain.poly-mult-var domain-axioms gen-ideal-X-iff kcr.m-closed poly-invariant
normalize-take-is-poly monoid.simps(2) subring univ-poly-def
var-closed(1) zero-not-one)
  apply force
  by (metis One-nat-def coeff.simps(1) coeffs-of-add-poly diff-self-eq-0 kcr.l-zero
kcr.one-closed
lead-coeff-simp length-Cons list.distinct(1) list.sel(1) list.size(3) univ-poly-one
univ-poly-zero
univ-poly-zero-closed)
  next
  interpret kcr:cring (carrier R)[X]
  using carrier-is-subring univ-poly-is-cring by auto
  have rule-1:⟨{y ∈ carrier (poly-ring R). local.coeff y 0 = xa} ∈ carrier (poly-ring
R Quot Idlpoly-ring R {X}) =
(∃x ∈ carrier (poly-ring R). {y ∈ carrier (poly-ring R). local.coeff y 0 = xa} =
(⋃ xa ∈ Idlpoly-ring R {X}. {xa ⊕poly-ring R x}))⟩ for xa
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def by(auto)

  have rule-2:⟨(∧x. x ∈ carrier (poly-ring R Quot Idlpoly-ring R {X}) ⇒
x ∈ (λx. {y ∈ carrier (poly-ring R). local.coeff y 0 = x}) ‘ carrier R)
⇒ (∧xa. xa ∈ carrier (poly-ring R) ⇒
(⋃ xa ∈ Idlpoly-ring R {X}. {x ⊕poly-ring R xa}) ∈ (λx. {y ∈ carrier (poly-ring
R). local.coeff y 0 = x}) ‘ carrier R)⟩
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def
  using UN-singleton by auto
  have rule-2':⟨(∧xa. xa ∈ carrier (poly-ring R) ⇒
(⋃ xa ∈ Idlpoly-ring R {X}. {x ⊕poly-ring R xa}) ∈ (λx. {y ∈ carrier (poly-ring
R). local.coeff y 0 = x}) ‘ carrier R)
⇒ (∧x. x ∈ carrier (poly-ring R Quot Idlpoly-ring R {X}) ⇒
x ∈ (λx. {y ∈ carrier (poly-ring R). local.coeff y 0 = x}) ‘ carrier R)⟩
  unfolding FactRing-def A-RCOSETS-def RCOSETS-def r-coset-def
  using UN-singleton by auto
  show ⟨bij-betw (λx. {y ∈ carrier ((carrier R) [X]). local.coeff y 0 = x}) (carrier
R) (carrier ((carrier R) [X] Quot Idl(carrier R) [X] {X}))⟩
  unfolding bij-betw-def

```

```

apply(safe)
  apply(rule inj-onI)
subgoal proof –
  fix x :: 'a and y :: 'a
  assume a1: x ∈ (carrier R)
  assume a2: y ∈ (carrier R)
  assume {y ∈ carrier ((carrier R) [X]). local.coeff y 0 = x} = {ya ∈ carrier
((carrier R) [X]). local.coeff ya 0 = y}
  then have y = (THE a. ∀ as. as ∈ {as ∈ carrier ((carrier R) [X]). local.coeff
as 0 = x} → local.coeff as 0 = a)
    using a2 The-a-is-a by force
  then show x = y
    using a1 The-a-is-a by auto
qed
proof(subst rule-1)
  fix x xa
  have rule-1:⟨x' ∈ (⋃ xa∈{p ∈ carrier (poly-ring R). local.coeff p 0 = 0}. {xa
⊕poly-ring R [local.coeff x' 0]}) =
(∃ xa. xa ∈ carrier (poly-ring R) ∧ local.coeff xa 0 = 0 ∧ x' = xa ⊕poly-ring R
[local.coeff x' 0])⟩ for x'
    by simp
  assume h1':⟨xa ∈ carrier R⟩
  then show ⟨∃ x∈carrier (poly-ring R).
{y ∈ carrier (poly-ring R). local.coeff y 0 = xa} = (⋃ xa∈Idlpoly-ring R
{X}. {xa ⊕poly-ring R x})⟩
    apply(cases ⟨xa=0⟩)
    apply(rule beXI[where x=⟨[]⟩])
    using gen-ideal-X-iff kcr.r-zero univ-poly-zero apply(safe)[1]
    apply (simp add: univ-poly-zero)+
    apply (simp add: univ-poly-zero-closed)
    apply(rule beXI[where x=⟨[xa]⟩])
    apply(subst gen-ideal-X-iff')
    apply(safe)
    apply(subst rule-1)
    apply (metis coeff.simps(1) coeff-0-of-mult-X diff-le-self kcr.m-closed
normalize-take-is-poly poly-invariant subring var-closed(1))
    apply (metis bot-least insert-subset list.simps(15) poly-add-is-polynomial
polynomial-incl'
set-empty2 subring univ-poly-add univ-poly-carrier)
    apply (metis diff-Suc-1 insert-subset kcr.zero-closed l-zero lead-coeff-simp
length-Cons
list.distinct(1) list.sel(1) list.simps(15) list.size(3) poly-add-coeff poly-
nomial-incl' univ-poly-add univ-poly-zero)
    by (metis Diff-iff const-is-polynomial emptyE insertE univ-poly-carrier)
  next
  show ⟨∧x. x ∈ carrier (poly-ring R Quot Idlpoly-ring R {X})
⇒ x ∈ (λx. {y ∈ carrier (poly-ring R). local.coeff y 0 = x}) ' carrier R⟩
  proof(rule rule-2')
  fix x xa

```

**assume**  $h1: \langle x \in \text{carrier } (\text{poly-ring } R \text{ Quot } \text{Idl}_{\text{poly-ring } R} \{X\}) \rangle \langle xa \in \text{carrier } (\text{poly-ring } R) \rangle$   
**then show**  $\langle (\bigcup x \in \text{Idl}_{\text{poly-ring } R} \{X\}. \{x \oplus_{\text{poly-ring } R} xa\}) \in (\lambda x. \{y \in \text{carrier } (\text{poly-ring } R). \text{local.coeff } y \ 0 = x\}) \text{ 'carrier } R \rangle$   
**apply** (*simp only: image-def, safe*)  
**apply** (*rule bexI [where x = \langle \text{coeff } xa \ 0 \rangle]*)  
**apply** (*safe*)  
**by** (*auto simp: gen-ideal-X-iff coeffs-of-add-poly domain-axioms polynomial-incl'*)  
*(metis coeff.simps(1) coeffs-of-add-poly gen-ideal-X-iff insertI1 kcr.add.inv-closed*  
*kcr.add.inv-solve-right kcr.add.m-comm kcr.l-neg kcr.minus-closed*  
*kcr.minus-eq univ-poly-zero)+*  
**qed**  
**qed**  
**qed**  
**qed**

**lemma** (*in domain*) *RX-imp-RX-over-X*:  
 $\langle \text{noetherian-ring } (\text{carrier } R[X]) \implies \text{noetherian-ring } (\text{carrier } R[X] \text{ Quot } \text{genideal } (\text{carrier } R[X]) \{X\}) \rangle$   
**by** (*meson domain.var-closed(1) domain-axioms empty-subsetI insert-subset noetherian-ring-def*  
*noetherian-ring-imp-quot-noetherian-ring ring.genideal-ideal subring)*

**lemma** (*in domain*) *noetherian-RX-imp-noetherian-R*:  
 $\langle \text{noetherian-ring } ((\text{carrier } R)[X]) \implies \text{noetherian-ring } R \rangle$   
**proof** –  
**assume**  $h1: \langle \text{noetherian-ring } ((\text{carrier } R)[X]) \rangle$   
**have**  $\langle \text{noetherian-ring } (((\text{carrier } R)[X]) \text{ Quot } (\text{genideal } ((\text{carrier } R)[X]) \{X\})) \rangle$   
**using** *RX-imp-RX-over-X h1 by auto*  
**moreover have**  $\langle (((\text{carrier } R)[X]) \text{ Quot } (\text{genideal } ((\text{carrier } R)[X]) \{X\})) \simeq R \rangle$   
**using** *R-isom-RX-X local.ring-axioms ring-iso-sym by blast*  
**ultimately show** *?thesis*  
**using** *local.ring-axioms noetherian-isom-imp-noetherian by blast*  
**qed**

**lemma** *principal-imp-noetherian*:  $\langle \text{principal-domain } R \implies \text{noetherian-ring } R \rangle$   
**proof** –  
**assume**  $h1: \langle \text{principal-domain } R \rangle$   
**then show** *?thesis*  
**apply** (*intro ring.noetherian-ringI*)  
**using** *cring.axioms(1) domain-def principal-domain.axioms(1) apply blast*  
**by** (*metis cring.cgenideal-eq-genideal domain-def empty-subsetI finite.emptyI finite.insertI*  
*insert-subset principal-domain.axioms(1) principal-domain.exists-gen)*  
**qed**

**lemma** (in ring) *coeff-iff-poly-carrier*: $\langle x \in \text{carrier } (\text{poly-ring } R) \implies y \in \text{carrier } (\text{poly-ring } R) \implies (x=y) \iff \text{coeff } x = \text{coeff } y \rangle$   
**by** (auto simp add: *coeff-iff-polynomial-cond univ-poly-carrier*)

**lemma** *zero-is-zero*: $\langle B = B(\text{zero} := \mathbf{0}_B) \rangle$   
**unfolding** *ring-def monoid-def ring-axioms-def abelian-group-def abelian-group-axioms-def abelian-monoid-def comm-monoid-def* **by**(auto)

**lemma** *ring-iso-imp-iso*: $\langle A \simeq B \implies A \cong B \rangle$   
**unfolding** *is-ring-iso-def is-iso-def ring-iso-def iso-def ring-hom-def hom-def* **by**(auto)

**lemma** (in ring) *iso-imp-exist-0*: $\langle R \simeq B \implies \exists x. \text{ring } (B(\text{zero}:=x)) \rangle$   
**proof** –  
**assume** *h1*: $\langle R \simeq B \rangle$   
**have**  $\langle \text{ring } R \rangle$   
**by** (simp add: *local.ring-axioms*)  
**with** *h1* **obtain** *h* **where** *f0*: $\langle h \in \text{ring-hom } R B \wedge \text{bij-betw } h (\text{carrier } R) (\text{carrier } B) \rangle$   
**unfolding** *is-ring-iso-def ring-iso-def* **by** *auto*  
**then** **have** *f1*: $\text{ring } (B(\text{carrier} := h \text{ ` } (\text{carrier } R), \text{zero} := h \mathbf{0}_R))$   
**using** *ring-hom-imp-imp-ring[of ] h1* **unfolding** *ring-iso-def*  
**using** *ring.ring-hom-imp-imp-ring* **by** *blast*  
**moreover** **have** *f2*: $h \text{ ` } (\text{carrier } R) = \text{carrier } B$   
**using** *h1* **unfolding** *ring-iso-def bij-betw-def*  
**by** (simp add: *f0 bij-betw-imp-surj-on*)  
**then** **show** *?thesis* **using** *f1 f2* **by**(auto)  
**qed**

**lemma** (in domain) *noetherian-R-imp-noetherian-UP-R*:  
**assumes** *h1*: $\langle \text{noetherian-ring } R \rangle$   
**shows**  $\langle \text{noetherian-ring } (UP R) \rangle$   
**proof** –  
**interpret** *UPring*: *UP-ring R UP R*  
**by** (simp add: *UP-ring-def local.ring-axioms*)  
**have**  $\langle \text{noetherian-ring } ((\text{carrier } R)[X]) \rangle$   
**using** *noetherian-domain.weak-Hilbert-basis h1*  
**using** *domain-axioms noetherian-domain.intro* **by** *auto*  
**with** *h1* **show** *?thesis*  
**unfolding** *noetherian-domain-def*  
**using**  $\langle \text{noetherian-ring } (\text{poly-ring } R) \rangle$  *noetherian-isom-imp-noetherian h1 UP-ring.UP-ring RX-iso-P*  
**by** *blast*

qed

**lemma** (in domain) noetheriandom-R-imp-noetheriandom-UP-R:

assumes  $h1: \langle \text{noetherian-domain } R \rangle$

shows  $\langle \text{noetherian-domain } (UP\ R) \rangle$

**proof** –

**interpret** UP-dom: UP-domain R UP R

by (simp add: UP-domain.intro domain-axioms)+

**have**  $\langle \text{noetherian-ring } ((\text{carrier } R)[X]) \rangle$

using noetherian-domain.weak-Hilbert-basis h1

by(auto)

**with** h1 **show** ?thesis

**unfolding** noetherian-domain-def

**using** UP-dom.domain-axioms noetherian-R-imp-noetherian-UP-R **by** blast

qed

**lemma** (in cring) Pring-one-index-isom-P:  $\langle (Pring\ R\ \{N\}) \simeq UP\ R \rangle$

**proof** –

**interpret** UPcring: UP-cring R UP R

by (simp add: UP-cring-def is-cring)+

**have**  $\langle IP\text{-to-UP } N \in \text{ring-hom } (Pring\ R\ \{N\})\ (UP\ R) \rangle$

by (simp add: UPcring.IP-to-UP-ring-hom ring-hom-ring.homh)

**then show** ?thesis **unfolding** is-ring-iso-def ring-iso-def

**apply**(subst ex-in-conv[symmetric])

**apply**(rule exI[**where**  $x = \langle IP\text{-to-UP } N \rangle$ ])

**unfolding** bij-betw-def **apply**(safe)

**apply** (simp add: UPcring.IP-to-UP-ring-hom-inj)

**apply** (simp add: IP-to-UP-closed is-cring)

**by** (metis UPcring.IP-to-UP-inv UPcring.UP-to-IP-closed image-eqI)

qed

**lemma** (in cring) P-isom-Pring-one-index:  $\langle UP\ R \simeq (Pring\ R\ \{N\}) \rangle$

**proof** –

**interpret** UPcring: UP-cring R UP R

by (simp add: UP-cring-def is-cring)+

**interpret** crR: cring Pring R {N}

by (simp add: Pring-is-cring is-cring)

**show** ?thesis

**using** cring.Pring-one-index-isom-P crR.ring-axioms ring-iso-sym is-cring **by**

fastforce

qed

**lemma** (in domain) P-iso-RX:  $\langle UP\ R \simeq ((\text{carrier } R)[X]) \rangle$

**proof** –

**interpret** d: domain (carrier R)[X]

by (simp add: subring univ-poly-is-domain)

**have**  $\langle (\text{carrier } R)[X] \simeq UP\ R \rangle$

**using** RX-iso-P UP-ring-def local.ring-axioms **by** blast

```

then show ?thesis
  using d.ring-axioms ring-iso-sym by blast
qed

```

```

lemma (in domain) IP-noeth-imp-R-noeth:⟨noetherian-ring (Pring R {a}) ⟹
noetherian-ring R⟩
proof –
  assume h1:⟨noetherian-ring (Pring R {a})⟩
  have ⟨(Pring R {a}) ≃ ((carrier R)[X])⟩
  by (meson Pring-one-index-isom-P domain.P-iso-RX domain-axioms ring-iso-trans)

  then have ⟨noetherian-ring ((carrier R)[X])⟩
  using domain.univ-poly-is-ring domain-axioms h1 noetherian-isom-imp-noetherian
  subring by blast
  then show ?thesis
  using noetherian-RX-imp-noetherian-R by fastforce
qed

```

```

lemma (in domain) R-iso-UPR-quot-X:⟨R ≃ (UP R) Quot (cgenideal (UP R) (λi.
if i=1 then 1 else 0))⟩
proof –
  interpret UP-r: UP-ring R UP R
  by (simp add: UP-ring-def local.ring-axioms)+
  have f0:⟨coeff ∈ ring-iso (poly-ring R) (UP R)⟩
  using coeff-iso-RX-P by blast
  have ⟨X ∈ carrier (poly-ring R)⟩ ⟨(λi. if i = 1 then 1 else 0) ∈ carrier (UP R)⟩
  ⟨cring (poly-ring R)⟩ ⟨cring (UP R)⟩
  apply (simp add: subring var-closed(1))
  apply(force simp:UP-def up-def)
  apply (simp add: subring univ-poly-is-cring)
  by (simp add: UP-cring.UP-cring UP-cring.intro is-cring)
  then have ⟨(carrier R[X]) Quot (cgenideal (poly-ring R) X) ≃(UP R) Quot
(cgenideal (UP R) (λi. if i=1 then 1 else 0))⟩
  using Quot-iso-cgen[of X ⟨poly-ring R⟩ ⟨(λi. if i=1 then 1 else 0)⟩ ⟨(UP R)⟩
coeff] X-has-correp
  f0 by fastforce
  then show ?thesis
  using domain.R-isom-RX-X domain-axioms gen-is-cgen ring-iso-trans by force
qed

```

end

## 5 The Hilbert Basis theorem for Indexed Polynomials Rings

theory Hilbert-Basis



**imports** *Weak-Hilbert-Basis*

**begin**

## 5.1 The isomorphism between $A[X_0..X_n]$ and $A[X_0..X_{n-1}][X_n]$

This part until *var\_factor\_iso* is due to Aaron Crighton

**lemma** *ring-iso-memI'*:

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

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

**assumes**  $\bigwedge x. x \in \text{carrier } R \implies g (f x) = x$

**assumes**  $\bigwedge x. x \in \text{carrier } S \implies f (g x) = x$

**shows**  $f \in \text{ring-iso } R \ S$

$g \in \text{ring-iso } S \ R$

**proof** –

**show**  $0: f \in \text{ring-iso } R \ S$

**unfolding** *ring-iso-def mem-Collect-eq*

**apply**(*rule conjI*, *rule assms(1)*, *rule bij-betwI*[*of - - g*])

**using** *assms ring-hom-memE* **by** *auto*

**show**  $g \in \text{ring-iso } S \ R$

**unfolding** *ring-iso-def mem-Collect-eq*

**apply**(*rule conjI*, *rule assms(2)*, *rule bij-betwI*[*of - - f*])

**using** *assms ring-hom-memE* **by** *auto*

**qed**

**lemma**(*in cring*) *var-factor-inverse*:

**assumes**  $I = J0 \cup J1$

**assumes**  $J1 \subseteq I$

**assumes**  $J1 \cap J0 = \{\}$

**assumes**  $\psi1 = (\text{var-factor-inv } I \ J0 \ J1)$

**assumes**  $\psi0 = (\text{var-factor } I \ J0 \ J1)$

**assumes**  $P \in \text{carrier } (\text{Pring } (\text{Pring } R \ J0) \ J1)$

**shows**  $\psi0 (\psi1 P) = P$

**proof**(*induct rule: ring.Pring-car-induct''*[*of Pring R J0 - J1*])

**case** *1*

**then show** *?case*

**using** *Pring-is-ring* **by** *blast*

**next**

**case** *2*

**then show** *?case*

**using** *assms(6)* **by** *force*

**next**

**case** (*3 c*)

**interpret** *pring-cring: cring Pring R J0*

**using** *Pring-is-cring is-cring* **by** *auto*

**interpret** *Rcring: cring R*

**using** *is-cring* **by** *auto*

**have**  $0: \text{ring-hom-ring } (\text{Pring } (\text{Pring } R \ J0) \ J1) \ (\text{Pring } R \ I) \ \psi1$

```

    by (simp add: assms(1) assms(3) assms(4) var-factor-inv-morphism(1))
  have 1: ring-hom-ring (Pring R I) (Pring (Pring R J0) J1)  $\psi_0$ 
    by (simp add: assms(1) assms(3) assms(5) var-factor-morphism'(1))
  have 2:  $\psi_0 \circ \psi_1 \in \text{ring-hom} (Pring (Pring R J0) J1) (Pring (Pring R J0) J1)$ 

    using 0 1 ring-hom-trans[of  $\psi_1$  Pring (Pring R J0) J1 Pring R I  $\psi_0$  Pring
(Pring R J0) J1]
      ring-hom-ring.homh[of Pring R I Pring (Pring R J0) J1  $\psi_0$ ]
      ring-hom-ring.homh[of Pring (Pring R J0) J1 Pring R I  $\psi_1$ ]
    by blast
  then show ?case using assms
    by (simp add: 3 var-factor-inv-morphism(3) var-factor-morphism'(3))
next
case (4 p q)
interpret pring-cring: cring Pring R J0
  using Pring-is-cring is-cring by auto
interpret Rcring: cring R
  using is-cring by auto
have 0: ring-hom-ring (Pring (Pring R J0) J1) (Pring R I)  $\psi_1$ 
  by (simp add: assms(1) assms(3) assms(4) var-factor-inv-morphism(1))
have 1: ring-hom-ring (Pring R I) (Pring (Pring R J0) J1)  $\psi_0$ 
  by (simp add: assms(1) assms(3) assms(5) var-factor-morphism'(1))
have 2:  $\psi_0 \circ \psi_1 \in \text{ring-hom} (Pring (Pring R J0) J1) (Pring (Pring R J0) J1)$ 

    using 0 1 ring-hom-trans[of  $\psi_1$  Pring (Pring R J0) J1 Pring R I  $\psi_0$  Pring
(Pring R J0) J1]
      ring-hom-ring.homh[of Pring R I Pring (Pring R J0) J1  $\psi_0$ ]
      ring-hom-ring.homh[of Pring (Pring R J0) J1 Pring R I  $\psi_1$ ]
    by blast
  from 4 show ?case
proof-
  fix p q
  assume A:  $p \in \text{carrier} (Pring (Pring R J0) J1)$ 
     $q \in \text{carrier} (Pring (Pring R J0) J1)$ 
     $\psi_0 (\psi_1 p) = p$ 
     $\psi_0 (\psi_1 q) = q$ 
  show  $\psi_0 (\psi_1 (p \oplus_{Pring (Pring R J0) J1} q)) = p \oplus_{Pring (Pring R J0) J1} q$ 
    using A 2 ring-hom-add[of  $\psi_0 \circ \psi_1$  Pring (Pring R J0) J1 Pring (Pring R
J0) J1 p q]
      comp-apply[of  $\psi_0$   $\psi_1$ ]
    by (simp add: pring-cring.Pring-add pring-cring.Pring-car)
  qed
next
case (5 p i)
interpret pring-cring: cring Pring R J0
  using Pring-is-cring is-cring by auto
interpret Rcring: cring R
  using is-cring by auto
have 0: ring-hom-ring (Pring (Pring R J0) J1) (Pring R I)  $\psi_1$ 

```

**by** (*simp add: assms(1) assms(3) assms(4) var-factor-inv-morphism(1)*)  
**have** 1: *ring-hom-ring* (*Pring R I*) (*Pring (Pring R J0) J1*)  $\psi_0$   
**by** (*simp add: assms(1) assms(3) assms(5) var-factor-morphism'(1)*)  
**have** 2:  $\psi_0 \circ \psi_1 \in \text{ring-hom}$  (*Pring (Pring R J0) J1*) (*Pring (Pring R J0) J1*)  
  
**using** 0 1 *ring-hom-trans*[of  $\psi_1$  *Pring (Pring R J0) J1 Pring R I*  $\psi_0$  *Pring (Pring R J0) J1*]  
*ring-hom-ring.homh*[of *Pring R I Pring (Pring R J0) J1*  $\psi_0$ ]  
*ring-hom-ring.homh*[of *Pring (Pring R J0) J1 Pring R I*  $\psi_1$ ]  
**by** *blast*  
**from** 5 **show** ?*case*  
**proof**–  
**fix**  $p \ i$  **assume**  $A: p \in \text{carrier}$  (*Pring (Pring R J0) J1*)  
 $\psi_0 (\psi_1 p) = p$   
 $i \in J1$   
**show**  $\psi_0 (\psi_1 (p \otimes_{\text{Pring (Pring R J0) J1}} \text{pvar (Pring R J0) } i)) =$   
 $p \otimes_{\text{Pring (Pring R J0) J1}} \text{pvar (Pring R J0) } i$   
**proof**–  
**have**  $A1: \psi_0 (\psi_1 (\text{pvar (Pring R J0) } i)) = \text{pvar (Pring R J0) } i$   
**by** (*metis A(3) assms(1) assms(2) assms(3) assms(4) assms(5)*)  
*var-factor-inv-morphism(2) var-factor-morphism'(2)*)  
**then show** ?*thesis*  
**using** 2  $A$  *ring-hom-mult*[of  $\psi_0 \circ \psi_1$  (*Pring (Pring R J0) J1*)] 2  
*Pring-car comp-apply*[of  $\psi_0 \psi_1$ ]  
**by** (*metis pring-crng.Pring-car pring-crng.Pring-var-closed*)  
**qed**  
**qed**  
**qed**

**lemma**(*in cring*) *var-factor-iso*:  
**assumes**  $I = J0 \cup J1$   
**assumes**  $J1 \subseteq I$   
**assumes**  $J1 \cap J0 = \{\}$   
**assumes**  $\psi_1 = (\text{var-factor-inv } I \ J0 \ J1)$   
**assumes**  $\psi_0 = (\text{var-factor } I \ J0 \ J1)$   
**shows**  $\psi_0 \in \text{ring-iso}$  (*Pring R I*) (*Pring (Pring R J0) J1*)  
 $\psi_1 \in \text{ring-iso}$  (*Pring (Pring R J0) J1*)(*Pring R I*)  
**proof**–  
**have** 1:  $\psi_0 \in \text{ring-hom}$  (*Pring R I*) (*Pring (Pring R J0) J1*)  
 $\psi_1 \in \text{ring-hom}$  (*Pring (Pring R J0) J1*) (*Pring R I*)  
 $\bigwedge x. x \in \text{carrier}$  (*Pring R I*)  $\implies \psi_1 (\psi_0 x) = x$   
 $\bigwedge x. x \in \text{carrier}$  (*Pring (Pring R J0) J1*)  $\implies \psi_0 (\psi_1 x) = x$   
**using** *assms var-factor-inv-inverse*[of  $I \ J0 \ J1 \ \psi_1$ ] *var-factor-inverse*[of  $I \ J0 \ J1 \ \psi_1$ ]  
**by** (*auto simp add: var-factor-inv-morphism(1) cring.var-factor-morphism'(1)*)  
*is-crng*  
*ring-hom-ring.homh*)  
**show**  $\psi_0 \in \text{ring-iso}$  (*Pring R I*) (*Pring (Pring R J0) J1*)

```

     $\psi1 \in \text{ring-iso } (\text{Pring } (\text{Pring } R \ J0) \ J1) \ (\text{Pring } R \ I)$ 
    using 1 ring-iso-memI[of  $\psi0 \ \text{Pring } R \ I \ \text{Pring } (\text{Pring } R \ J0) \ J1 \ \psi1$ ]
    by auto
qed

```

```

lemma (in cring) is-iso-Prings:
  assumes h1: $I = J0 \cup J1$ 
  assumes h2: $J1 \subseteq I$ 
  assumes h3: $J1 \cap J0 = \{\}$ 
  shows  $(\text{Pring } (\text{Pring } R \ J0) \ J1) \simeq (\text{Pring } R \ I)$  and  $(\text{Pring } R \ I) \simeq (\text{Pring } (\text{Pring } R \ J0) \ J1)$ 
proof -
  show  $\langle (\text{Pring } (\text{Pring } R \ J0) \ J1) \simeq (\text{Pring } R \ I) \rangle$ 
    unfolding is-ring-iso-def
    using h2 var-factor-iso[of  $I \ J0 \ J1 \ \langle \text{var-factor-inv } I \ J0 \ J1 \rangle \ \langle \text{var-factor } I \ J0 \ J1 \rangle$ ]
    using h1 h3 by auto
  show  $\langle (\text{Pring } R \ I) \simeq (\text{Pring } (\text{Pring } R \ J0) \ J1) \rangle$ 
    unfolding is-ring-iso-def
    using h2 var-factor-iso[of  $I \ J0 \ J1 \ \langle \text{var-factor-inv } I \ J0 \ J1 \rangle \ \langle \text{var-factor } I \ J0 \ J1 \rangle$ ]
    using h1 h3 by auto
qed

```

## 5.2 Preliminaries lemmas

```

lemma (in cring) poly-no-var:
  assumes  $\langle x \in ((\text{carrier } R) \ [\mathcal{X}_{\{\}}]) \wedge xa \neq \{\#\} \rangle$ 
  shows  $\langle x \ xa = \mathbf{0} \rangle$ 
  apply(rule ring.Pring-car-induct''[of  $R \ x \ \langle \{\} \rangle$ ])
  apply (simp add: local.ring-axioms)
  apply (simp add: Pring-car assms)
  unfolding indexed-const-def using assms
  by(auto simp add: Pring-add indexed-padd-def)

```

```

lemma (in cring) R-isom-P-mt: $\langle R \simeq \text{Pring } R \ \{\} \rangle$ 
proof -
  interpret cringP: cring Pring R  $\{\}$ 
  by (simp add: Pring-is-cring is-cring)
  have f0: $\langle \text{bij-betw indexed-const } (\text{carrier } R) \ (\text{carrier } (\text{Pring } R \ \{\})) \rangle$ 
  proof(unfold bij-betw-def inj-on-def, safe)
    fix x y
    assume h1: $\langle x \in \text{carrier } R \rangle \langle y \in \text{carrier } R \rangle \langle \text{indexed-const } x = \text{indexed-const } y \rangle$ 
    show  $\langle \text{indexed-const } x = \text{indexed-const } y \implies x = y \rangle$ 
    by (metis indexed-const-def)
  next
  fix x xa
  assume h1: $\langle xa \in \text{carrier } R \rangle$ 
  show  $\langle \text{indexed-const } xa \in \text{carrier } (\text{Pring } R \ \{\}) \rangle$ 

```

```

    by (simp add: h1 indexed-const-closed)
next
fix x::⟨'f multiset ⇒ 'a⟩
assume h1:⟨x ∈ carrier (Pring R { })⟩
then show ⟨x ∈ indexed-const ' carrier R⟩
  unfolding image-def apply(safe)
  apply(rule beXI[where x=⟨x {#}⟩])
  unfolding indexed-const-def
  by (auto simp:fun-eq-iff Pring-def poly-no-var)
qed
show ?thesis
  unfolding is-ring-iso-def ring-iso-def
  apply(subst ex-in-conv[symmetric])
  unfolding ring-hom-def
  apply(rule exI[where x=indexed-const])
  apply(safe)
    apply (simp add: indexed-const-closed)
    apply (simp add: indexed-const-mult)
  using cringP.indexed-padd-const
  apply (simp add: Pring-add indexed-padd-const)
  apply (simp add: Pring-one)
  by(simp add:f0)
qed

```

### 5.3 Hilbert Basis theorem

We show after this Hilbert basis theorem, based on Indexed Polynomials in HOL-Algebra and its extension in *PadicFields*

**theorem** (in domain) *Hilbert-basis*:

```

  assumes h1:⟨noetherian-ring R⟩ and h2:⟨finite I⟩
  shows ⟨noetherian-ring (Pring R I)⟩
proof(induct rule :finite.induct[OF h2])
  case 1
  interpret cringP: cring Pring R { }
    by (simp add: Pring-is-cring is-cring)
  show ?case
    using R-isom-P-mt cringP.ring-axioms h1 noetherian-isom-imp-noetherian by
  auto
next
  case (2 A a)
  have f0:⟨noetherian-ring (Pring R A)⟩
    using 2 by blast
  have f1:⟨cring (Pring R A)⟩
    using Pring-is-cring is-cring by auto
  interpret UPcring: UP-cring Pring R A UP (Pring R A)
    by (simp add: UP-cring.intro f1)+
  have f2:⟨Pring (Pring R A) {a} ≃ UP (Pring R A)⟩
    using cring.Pring-one-index-isom-P UP-cring-def f1
    by (simp add: UPcring.R.Pring-one-index-isom-P)

```

```

then have f3:⟨noetherian-ring (UP (Pring R A))⟩
  using Pring-is-domain domain.noetherian-R-imp-noetherian-UP-R f0 by blast
have f7:⟨cring (Pring (Pring R A) {a})⟩
  by (simp add: UPcring.R.Pring-is-cring f1)
then have ⟨UP (Pring R A) ≃ Pring (Pring R A) {a}⟩
  by (simp add: cring-def f2 ring-iso-sym)
have f6:⟨noetherian-ring (Pring (Pring R A) {a})⟩
  using ⟨UP (Pring R A) ≃ Pring (Pring R A) {a}⟩ cring.axioms(1) f3
  f7 noetherian-isom-imp-noetherian by auto
have f10:⟨a∉A ⇒ Pring (Pring R A) {a} ≃ (Pring R (insert a A))⟩
  apply(rule cring.is-iso-Prings(1))
  by (simp add: is-cring)+
have f11:⟨ring (Pring R (insert a A))⟩
  by (simp add: Pring-is-ring)
then show ?case
  apply(cases ⟨a∈A⟩)
  using f0
  apply (simp add: insert-absorb)
  using noetherian-isom-imp-noetherian[of ⟨Pring (Pring R A) {a}⟩
    ⟨(Pring R (insert a A))⟩] f10 f11 f6 by(simp)

```

qed

```

lemma (in domain) R-noetherian-implies-IP-noetherian:
  assumes h1:⟨noetherian-ring R⟩
  shows ⟨noetherian-ring (Pring R {0..N::nat})⟩
  using Hilbert-basis h1 by blast

```

```

lemma (in domain) IP-noetherian-implies-R-noetherian:
  assumes h1:⟨noetherian-ring (Pring R I)⟩ and h2:⟨finite I⟩
  shows ⟨noetherian-ring R⟩
proof(insert h1, induct rule:finite.induct[OF h2])
  case 1
  interpret cringP: cring Pring R {}
  by (simp add: Pring-is-cring is-cring)
  have ⟨Pring R {} ≃ R⟩
  using local.ring-axioms R-isom-P-mt ring-iso-sym by blast
  then show ?case
  using 1 local.ring-axioms noetherian-isom-imp-noetherian by blast

```

next

```

case (2 A a)
have f1:⟨cring (Pring R A)⟩
  using Pring-is-cring is-cring by auto
interpret UPcring: UP-cring Pring R A UP (Pring R A)
  by (simp add: UP-cring.intro f1)+
interpret dom: domain (Pring R (A))
  using Pring-is-domain by blast
have f2:⟨Pring (Pring R A) {a} ≃ UP (Pring R A)⟩
  using cring.Pring-one-index-isom-P UP-cring-def f1
  by (simp add: UPcring.R.Pring-one-index-isom-P)

```

```

{assume h2:⟨a∉A⟩
  then have ⟨(Pring (Pring R (A)) {a}) ≃ (Pring R (insert a A))⟩
    by (simp add: cring.is-iso-Prings(1) is-cring)
  then have ⟨noetherian-ring (Pring (Pring R (A)) {a})⟩
    using 2.premis UPcring.R.Pring-is-ring
      noetherian-isom-imp-noetherian ring-iso-sym by blast
  then have ⟨noetherian-ring (Pring R (A))⟩
    by (simp add: dom.IP-noeth-imp-R-noeth)
  then have ⟨noetherian-ring R⟩
    using 2.hyps(2) by blast}note a-not-in=this
then show ?case apply(cases ⟨a∈A⟩)
  using 2
  apply (simp add: insert-absorb)
  using a-not-in by blast
qed

```

end

## 6 The Hilbert Basis theorem for Formal Power Series

**theory** *Formal-Power-Series-Ring*

**imports**

*HOL-Library.Extended-Nat*  
*HOL-Computational-Algebra.Formal-Power-Series*  
*HOL-Algebra.Module*  
*HOL-Algebra.Ring-Divisibility*

**begin**

We define the ring of formal power series over a domain (idom) as a record to match HOL-Algebra definitions. We then show that it is a domain for addition and multiplication. This is immediate with the existing theory from HOL-Analysis.

We then proceed to show the theorem similar to Hilbert's basis theorem but for the ring of Formal power series.

### 6.1 Preliminaries definition and lemmas

**context**

**fixes**  $R::\langle'a::\{idom\} ring\rangle$  (**structure**)  
**defines**  $R::\langle R \equiv (\text{carrier} = UNIV, \text{monoid.mult} = (*), \text{one} = 1, \text{zero} = 0, \text{add} = (+))\rangle$   
**begin**

```

lemma ring-R:⟨ring R⟩
  apply(unfold-locales)
  using add.right-inverse
  by (auto simp add: R mult.assoc ab-semigroup-add-class.add-ac(1)
      add.left-commute Units-def add.commute ring-class.ring-distrib(2)
      ring-class.ring-distrib(1) exI[of - - x for x])

```

```

lemma domain-R:⟨domain R⟩
  apply(rule domainI)
  apply(rule cringI)
  apply (simp add: ring.is-abelian-group ring-R)
  apply (metis Groups.mult-ac(2) R monoid.monoid-comm-monoidI
      monoid.simps(1) ring.is-monoid ring-R)
  apply (simp add: ring.ring-simprules(13) ring-R)
  apply (simp add: R)
  by (simp add: R)

```

**definition**

```

FPS-ring :: 'a::{idom} fps ring
where FPS-ring = (⟨carrier = UNIV, monoid.mult = (*), one = 1, zero = 0,
  add = (+)⟩)

```

```

lemma ring-FPS:⟨ring FPS-ring⟩
  apply(rule ringI)
  apply(rule abelian-groupI)
  apply (simp-all add: FPS-ring-def ab-semigroup-add-class.add-ac(1)
      add.left-commute add.commute)
  apply (metis ab-group-add-class.ab-left-minus add.commute)
  apply(rule monoidI)
  by(simp-all add: FPS-ring-def mult.assoc ab-semigroup-add-class.add-ac(1)
      add.left-commute add.commute ring-class.ring-distrib(2) ring-class.ring-distrib(1))

```

```

lemma cring-FPS:⟨cring FPS-ring⟩
  apply(rule cringI)
  apply (simp add: ring.is-abelian-group ring-FPS)
  apply(rule comm-monoidI)
  apply (simp add: ring.ring-simprules(5) ring-FPS)
  apply (simp add: ring.ring-simprules(6) ring-FPS)
  apply (simp add: ring.ring-simprules(11) ring-FPS)
  apply (simp add: ring.ring-simprules(12) ring-FPS)
  apply (simp add: FPS-ring-def)
  by (simp add: ring.ring-simprules(13) ring-FPS)

```

```

lemma domain-FPS:⟨domain FPS-ring⟩
  apply(rule domainI)
  apply (simp add: cring-FPS)
  apply (simp add: FPS-ring-def)

```



by (simp add: FPS-ring-def)

valuation over FPS<sub>r</sub>ing

**definition** v-subdegree :: ('a::idom) fps ⇒ enat **where**  
v-subdegree f = (if f = 0 then ∞ else subdegree f)

**definition** valuation::('a::idom) fps ⇒ enat (ν) **where**  
⟨ν x ≡ Sup {enat k | k. x ∈ cgenideal FPS-ring (fps-X<sup>^</sup>k)}⟩

**lemma** fps-X-pow-k-ideal-iff:⟨cgenideal FPS-ring (fps-X<sup>^</sup>k) = {x. v-subdegree x ≥ k}⟩

**proof**(induct k)

  case 0

**then show** ?case **unfolding** cgenideal-def  
  **using** enat-def zero-enat-def  
  **by** (simp add: FPS-ring-def)

**next**

  case (Suc k)

**have** ⟨x ∈ carrier FPS-ring ⇒ v-subdegree (x\*fps-X<sup>^</sup>r) ≥ r⟩ **for** r x  
  **apply**(cases ⟨x=0⟩)  
  **unfolding** v-subdegree-def **by**(auto)

**then show** ?case **unfolding** cgenideal-def v-subdegree-def FPS-ring-def  
  **apply**(safe)  
  **apply**(auto simp:FPS-ring-def) [1]  
  **by** (metis (mono-tags, opaque-lifting) UNIV-I enat-ord-simps(1) fps-shift-times-fps-X-power  
  monoid.select-convs(1) mult-zero-left partial-object.select-convs(1))

**qed**

**lemma** valuation-miscs-1:**assumes** h1:⟨f ∈ carrier FPS-ring⟩  
**shows** ⟨(valuation f) = (∞::enat) ⟷ f = 0⟩  
**apply**(safe)  
**unfolding** valuation-def **apply**(subst (asm) fps-X-pow-k-ideal-iff)  
**apply** (smt (verit, best) Sup-least-infinity-ileE mem-Collect-eq v-subdegree-def)  
**apply**(subst fps-X-pow-k-ideal-iff)  
**unfolding** v-subdegree-def  
**unfolding** enat-def **apply**(clarsimp)  
**by** (smt (verit, ccfv-threshold) Suc-ile-eq Sup-le-iff enat.exhaust enat-def enat-ord-simps(2)  
  mem-Collect-eq nat-less-le order.refl)

**lemma** valuation-miscs-0:  
**shows** ⟨valuation f = Inf {enat n | n. fps-nth f n ≠ 0}⟩

**proof**(cases ⟨f=0⟩)

  case 1:True

**have** f1:⟨valuation f = ∞⟩  
  **using** 1 valuation-miscs-1  
  **by** (simp add: FPS-ring-def)

**have** f0:⟨{enat n | n. fps-nth f n ≠ 0} = {}⟩  
  **by** (simp add: 1)

```

show ?thesis
  apply(subst f0)
  unfolding Inf-enat-def using f1 by(auto)
next
case 2:False
have f0:⟨fps-nth f n ≠ 0 ⟹ f ∉ cgenideal FPS-ring (fps-X^(Suc n))⟩ for n
  apply(subst fps-X-pow-k-ideal-iff)
  unfolding v-subdegree-def
  using not-less-eq-eq subdegree-leI by auto
then have ⟨f ∉ cgenideal FPS-ring (fps-X^(n)) ⟹ ∀ i ≥ n. f ∉ cgenideal FPS-ring
(fps-X^(i))⟩
  for n
  by (simp add: 2 fps-X-pow-k-ideal-iff v-subdegree-def)
with f0 have f2:⟨fps-nth f n ≠ 0 ⟹ valuation f ≤ n⟩ for n
  unfolding valuation-def
  by (smt (verit, del-insts) Sup-le-iff enat-ord-simps(1) mem-Collect-eq not-less-eq-eq)
then have ⟨valuation f = v-subdegree f⟩
  by (smt (verit, best) 2 Orderings.order-eq-iff Sup-le-iff fps-X-pow-k-ideal-iff
mem-Collect-eq v-subdegree-def valuation-def)
then show ?thesis unfolding v-subdegree-def subdegree-def
  using 2 enat-def
  by (smt (z3) 2 LeastI-ex cInf-eq-minimum enat-def f2 fps-nonzero-nth mem-Collect-eq)

```

qed

```

lemma valuation-miscs-3:⟨valuation f = v-subdegree f⟩
proof(cases ⟨f=0⟩)
case 1:True
have f1:⟨valuation f = ∞⟩
  using 1 valuation-miscs-1
  by (simp add: FPS-ring-def)
show ?thesis
  by (metis 1 Formal-Power-Series-Ring.v-subdegree-def f1)
next
case 2:False
have f0:⟨fps-nth f n ≠ 0 ⟹ f ∉ cgenideal FPS-ring (fps-X^(Suc n))⟩ for n
  apply(subst fps-X-pow-k-ideal-iff)
  unfolding v-subdegree-def
  using not-less-eq-eq subdegree-leI by auto
then have ⟨f ∉ cgenideal FPS-ring (fps-X^(n)) ⟹ ∀ i ≥ n. f ∉ cgenideal FPS-ring
(fps-X^(i))⟩
  for n
  by (simp add: 2 fps-X-pow-k-ideal-iff v-subdegree-def)
with f0 have f2:⟨fps-nth f n ≠ 0 ⟹ valuation f ≤ n⟩ for n
  unfolding valuation-def
  by (smt (verit, del-insts) Sup-le-iff enat-ord-simps(1) mem-Collect-eq not-less-eq-eq)
then show ⟨valuation f = v-subdegree f⟩
  by (smt (verit, best) 2 Orderings.order-eq-iff Sup-le-iff fps-X-pow-k-ideal-iff
mem-Collect-eq v-subdegree-def valuation-def)

```

qed

**lemma** *triangular-ineq-v*: $\langle \text{valuation } (f + g) \geq \min (\text{valuation } f) (\text{valuation } g) \rangle$   
**apply**(*subst* (1 2 3) *valuation-miscs-3*)  
**unfolding** *v-subdegree-def*  
**by** (*simp add: subdegree-add-ge'*)

**lemma** *triang-eq-v*:**assumes** *h1*: $\langle \text{valuation } f \neq \text{valuation } g \rangle$   
**shows**  $\langle \text{valuation } (f+g) = \min (\text{valuation } f) (\text{valuation } g) \rangle$   
**proof** –  
**have** *f0*: $\langle \text{valuation } (f+g) \geq \min (\text{valuation } f) (\text{valuation } g) \rangle$   
**by** (*simp add:triangular-ineq-v FPS-ring-def*)  
**have**  $\langle \text{valuation } (f+g) \leq \min (\text{valuation } f) (\text{valuation } g) \rangle$   
**apply**(*subst* (1 2 3) *valuation-miscs-3*) **unfolding** *min-def v-subdegree-def*  
**by** (*smt (verit, ccfv-threshold) Suc-le-eq add-cancel-right-left add-diff-cancel-right'*  
  
*add-eq-0-iff2 diff-zero enat-ord-simps(2) enat-ord-simps(3) h1 not-less-eq-eq*  
*order-le-less*  
*subdegree-add-eq1 subdegree-add-eq2 subdegree-uminus v-subdegree-def valuation-miscs-3*)  
**then show** *?thesis using f0*  
**by** *order*  
qed

**lemma** *prod-triang-v*: $\langle \text{valuation } (f*g) = \text{valuation } f + \text{valuation } g \rangle$   
**apply**(*subst* (1 2 3) *valuation-miscs-3*)  
**unfolding** *v-subdegree-def by(auto)*

## 6.2 Premisses for noetherian ring proof

**definition** *subdeg-poly-set*: $\langle \text{subdeg-poly-set } S k = \{a. a \in S \wedge \text{subdegree } a = k\} \cup \{0\} \rangle$

**definition** *sublead-coeff-set*: $\langle 'b::\{\text{zero}\} \text{ fps set} \Rightarrow \text{nat} \Rightarrow 'b \text{ set} \rangle$   
**where**  $\langle \text{sublead-coeff-set } S k \equiv \{ \text{fps-nth } a (\text{subdegree } a) \mid a. a \in \text{subdeg-poly-set } S k \} \rangle$

**lemma** *ideal-nonempty*: $\langle \text{ideal } I \text{ FPS-ring} \Longrightarrow I \neq \{\} \rangle$   
**by** (*metis FPS-ring-def UNIV-I empty-iff ideal.axioms(2)*  
*partial-object.select-conv(1) ring.quotient-eq-iff-same-a-r-cos*)

**lemma** *mult-X-in-ideal*: $\langle \text{ideal } I \text{ FPS-ring} \Longrightarrow \forall x \in I. \text{fps-X} * x \in I \rangle$   
**unfolding** *ideal-def ideal-axioms-def*  
**by** (*simp add: FPS-ring-def*)

**lemma** *non-empty-sublead*: $\langle \text{ideal } I \text{ FPS-ring} \Longrightarrow \text{sublead-coeff-set } I k \neq \{\} \rangle$   
**unfolding** *sublead-coeff-set-def subdeg-poly-set by(auto)*

**lemma** *inv-unique*: $\langle \forall x \in \text{carrier FPS-ring}. \exists !y. x + y = 0 \rangle$   
**by** (*metis add.right-inverse add-diff-cancel-left'*)

**lemma** *inv-same-degree:assumes*  $h:\langle x \in \text{carrier FPS-ring} \rangle$   
**shows**  $\langle \text{subdegree } (\text{inv\_add-monoid FPS-ring } x) = \text{subdegree } x \rangle$   
**by** (*metis FPS-ring-def ring-FPS abelian-group.a-group add-eq-0-iff2 group.l-inv*  
 $h$   
 $\text{monoid.select-convs}(1)$   $\text{monoid.select-convs}(2)$   $\text{partial-object.select-convs}(1)$   
*ring-def*  
 $\text{ring-record-simps}(11)$   $\text{ring-record-simps}(12)$   $\text{subdegree-uminus}$ )

**lemma** *inv-subdegree-is-inv: assumes*  $h:\langle x \in \text{carrier FPS-ring} \rangle$   
**shows**  $\langle \text{fps-nth } (\text{inv\_add-monoid FPS-ring } x) (\text{subdegree } x) =$   
 $(\text{inv\_add-monoid } R (\text{fps-nth } x (\text{subdegree } x))) \rangle$   
**unfolding** *a-inv-def*  
**by** (*metis FPS-ring-def ring-FPS R UNIV-I a-inv-def*  
 $\text{fps-add-nth}$   $\text{partial-object.select-convs}(1)$   
 $\text{ring.ring-simprules}(17)$   $\text{ring.ring-simprules}(9)$   $\text{ring-R}$   
 $\text{ring-record-simps}(12)$ )

**lemma** *subdeg-inv-in-sublead:*  
**assumes**  $h1:\langle \text{ideal } I \text{ FPS-ring} \rangle$  **and**  $h2:\langle a \in \text{sublead-coeff-set } I \ k \rangle$   
**shows**  $\langle \text{inv\_add-monoid } R \ a \in \text{sublead-coeff-set } I \ k \rangle$   
**proof** –  
**have**  $f0:\langle x \in I \implies \text{inv\_add-monoid FPS-ring } x \in I \rangle$  **for**  $x$   
**by** (*meson additive-subgroup-def h1 ideal.axioms(1) subgroup.m-inv-closed*)  
**then have**  $f1:\langle x \in I \implies \text{inv\_add-monoid FPS-ring } x \in \text{subdeg-poly-set } I (\text{subdegree } x) \rangle$  **for**  $x$   
**unfolding** *subdeg-poly-set using*  $\text{UnCI } h1$  *ideal.Icarr[of I FPS-ring x]* *inv-same-degree[of x]*  
 $\text{mem-Collect-eq}$  **by** (*auto*)  
**have**  $f2:\langle x \in I \implies (\text{inv\_add-monoid } R (\text{fps-nth } x (\text{subdegree } x)))$   
 $\in \text{sublead-coeff-set } I (\text{subdegree } x) \rangle$   
**for**  $x$   
**unfolding** *sublead-coeff-set-def*  
**using**  $f1$  *[of x]*  $h1$  *ideal.Icarr[of I FPS-ring x]* *inv-same-degree[of x]*  
 $\text{inv-subdegree-is-inv[of x]}$  *mem-Collect-eq*  
**by force**  
**have**  $\langle 0 \in I \rangle$   
**by** (*metis FPS-ring-def additive-subgroup.zero-closed h1 ideal.axioms(1) ring.simps(1)*)  
**then have**  $f3:\langle a \neq 0 \implies \exists x \in I. a = \text{fps-nth } x (\text{subdegree } x) \wedge \text{subdegree } x = k \rangle$   
**using**  $h2$  **unfolding** *sublead-coeff-set-def subdeg-poly-set*  
**by** (*auto*)  
**then obtain**  $x$  **where**  $\langle a \neq 0 \implies x \in I \wedge a = \text{fps-nth } x (\text{subdegree } x) \wedge \text{subdegree } x = k \rangle$  **by blast**  
**then have**  $f5:\langle a \neq 0 \implies \text{inv\_add-monoid } R \ a = \text{fps-nth } (\text{inv\_add-monoid FPS-ring } x) (\text{subdegree } x) \rangle$   
**by** (*metis FPS-ring-def UNIV-I inv-subdegree-is-inv partial-object.select-convs(1)*)  
**then have**  $f6:\langle a \neq 0 \implies \text{fps-nth } (\text{inv\_add-monoid FPS-ring } x) (\text{subdegree } x) \in \text{sublead-coeff-set } I \ k \rangle$   
**using**  $\langle a \neq 0 \implies x \in I \wedge a = \text{fps-nth } x (\text{subdegree } x) \wedge \text{subdegree } x = k \rangle$   $f2$

```

by force
  then show ?thesis
    apply(cases ⟨a=0⟩)
    apply (metis R a-inv-def h2 ring.minus-zero ring-R ring-record-simps(11))
    using f5 by presburger
qed

```

```

lemma mult-stable-sublead:
  assumes h1:⟨ideal I FPS-ring⟩
    and h2:⟨a ∈ sublead-coeff-set I k⟩
    and h3:⟨b ∈ sublead-coeff-set I k⟩
  shows ⟨a ⊗R b ∈ sublead-coeff-set I k⟩
proof -
  have ⟨0∈I⟩
  by (metis FPS-ring-def additive-subgroup.zero-closed h1 ideal.axioms(1) ring.simps(1))
  {assume h4:⟨a≠0⟩ and h5:⟨b≠0⟩
    then have f3:⟨∃ x∈I. a = fps-nth x (subdegree x) ∧ subdegree x = k⟩
      using h2 unfolding sublead-coeff-set-def subdeg-poly-set
      by(auto)
    then obtain x where f0:⟨x∈I ∧ a = fps-nth x (subdegree x) ∧ subdegree x =
k⟩ by blast
    then have ⟨fps-const b ∈ carrier FPS-ring⟩
      by (simp add: FPS-ring-def)
    then have ⟨fps-const b*x ∈ I⟩
      by (metis FPS-ring-def f0 h1 ideal-axioms-def ideal-def monoid.simps(1))
    then have ⟨fps-nth (fps-const b * x) k = a*b⟩
      by (simp add: f0)
    then have ⟨subdegree (fps-const b * x) = k⟩
      using f0 h4 h5 by force
    then have ⟨a ⊗R b ∈ sublead-coeff-set I k⟩
      unfolding sublead-coeff-set-def subdeg-poly-set FPS-ring-def
      using R ⟨fps-const b * x ∈ I⟩ ⟨fps-nth (fps-const b * x) k = a * b⟩ by force
  }note proof-2=this
  then show ?thesis
    apply(cases ⟨a=0 ∨ b=0⟩)
    using R h2 h3 by auto
qed

```

```

lemma add-stable-sublead:
  assumes h1:⟨ideal I FPS-ring⟩
    and h2:⟨a ∈ sublead-coeff-set I k⟩
    and h3:⟨b ∈ sublead-coeff-set I k⟩
  shows ⟨a ⊗add-monoid R b ∈ sublead-coeff-set I k⟩
proof -
  have f0:⟨0∈I⟩
  by (metis FPS-ring-def additive-subgroup.zero-closed h1 ideal.axioms(1) ring.simps(1))
  have p2:⟨a=-b ⟹ a + b ∈ sublead-coeff-set I k⟩
    unfolding sublead-coeff-set-def subdeg-poly-set by(auto)
  {assume h4:⟨a≠0⟩ and h5:⟨b≠0⟩ and h6:⟨a≠- b⟩

```

```

then have f3:⟨ $\exists x \in I. a = \text{fps-nth } x \text{ (subdegree } x) \wedge \text{subdegree } x = k$ ⟩
  using h2 unfolding sublead-coeff-set-def subdeg-poly-set
  by(auto)
then obtain x where f2:⟨ $x \in I \wedge a = \text{fps-nth } x \text{ (subdegree } x) \wedge \text{subdegree } x =$ 
k⟩ by blast
have f4:⟨ $\exists x \in I. b = \text{fps-nth } x \text{ (subdegree } x) \wedge \text{subdegree } x = k$ ⟩
  using f0 h3 h4 h5 unfolding sublead-coeff-set-def subdeg-poly-set
  by(auto)
then obtain y where f1:⟨ $y \in I \wedge b = \text{fps-nth } y \text{ (subdegree } y) \wedge \text{subdegree } y =$ 
k⟩ by blast
then have ⟨ $x + y \in I$ ⟩ using h1 unfolding ideal-def
  using f2 additive-subgroup.a-closed[of I FPS-ring x y]
  by (simp add: FPS-ring-def)
have f4:⟨ $\text{fps-nth } (x+y) k = a + b$ ⟩
  by (simp add: f1 f2)
have ⟨ $\forall i < k. \text{fps-nth } (x+y) i = 0$ ⟩
  by (simp add: f1 f2 nth-less-subdegree-zero)
then have f5:⟨ $\text{subdegree } (x + y) = k$ ⟩
  by (metis ⟨ $\text{fps-nth } (x + y) k = a + b$ ⟩ eq-neg-iff-add-eq-0 h6 subdegreeI)
then have ⟨ $a+b \in \text{sublead-coeff-set } I k$ ⟩
  using f4 f5 unfolding sublead-coeff-set-def subdeg-poly-set
  using ⟨ $x + y \in I$ ⟩ by force
note proof-1=this
then show ?thesis
  apply(cases ⟨ $a=0 \vee b = 0 \vee a = -b$ ⟩)
  using R h2 h3 p2 proof-1 by auto
qed

```

**lemma** outer-stable-sublead:

```

assumes h1:⟨ideal I FPS-ring⟩ and h2:⟨ $a \in \text{sublead-coeff-set } I k$ ⟩ and h3:⟨ $b \in$ 
carrier R⟩
shows ⟨ $b \otimes a \in \text{sublead-coeff-set } I k$ ⟩
proof –
have ⟨ $0 \in I$ ⟩
  by (metis FPS-ring-def additive-subgroup.zero-closed h1 ideal.axioms(1) ring.simps(1))
then have p2:⟨ $0 \in \text{sublead-coeff-set } I k$ ⟩ unfolding sublead-coeff-set-def subdeg-poly-set
by(auto)
{assume h4:⟨ $a \neq 0$ ⟩ and h5:⟨ $b \neq 0$ ⟩
  then have f3:⟨ $\exists x \in I. a = \text{fps-nth } x \text{ (subdegree } x) \wedge \text{subdegree } x = k$ ⟩
    using h2 unfolding sublead-coeff-set-def subdeg-poly-set
    by(auto)
  then obtain x where f0:⟨ $x \in I \wedge a = \text{fps-nth } x \text{ (subdegree } x) \wedge \text{subdegree } x =$ 
k⟩ by blast
  then have ⟨ $\text{fps-const } b \in \text{carrier FPS-ring}$ ⟩
    by (simp add: FPS-ring-def)
  then have ⟨ $\text{fps-const } b * x \in I$ ⟩
    by (metis FPS-ring-def f0 h1 ideal-axioms-def ideal-def monoid.simps(1))
  then have ⟨ $\text{fps-nth } (\text{fps-const } b * x) k = a * b$ ⟩
    by (simp add: f0)
}

```

```

then have ⟨subdegree (fps-const b * x) = k⟩
  using f0 h4 h5 by force
then have ⟨b ⊗R a ∈ sublead-coeff-set I k⟩
  unfolding sublead-coeff-set-def subdeg-poly-set FPS-ring-def
  using R ⟨fps-const b * x ∈ I⟩ ⟨fps-nth (fps-const b * x) k = a * b⟩ by force
}note proof-2=this
then show ?thesis
  apply(cases ⟨a=0 ∨ b=0⟩)
  using R h2 h3 proof-2 p2 by auto
qed

```

```

lemma sublead-ideal:⟨ideal I FPS-ring ⇒ ideal (sublead-coeff-set I k) R⟩
  apply(rule idealI)
  apply(simp add:ring-R)
  apply(rule group.subgroupI)
  using abelian-group.a-group ring.is-abelian-group ring-R apply fastforce
  apply (simp add: R)
  apply(simp add:non-empty-sublead)
  using subdeg-inv-in-sublead apply blast
  using add-stable-sublead apply force
  apply (simp add: outer-stable-sublead mult.commute)
  by (metis Groups.mult-ac(2) R monoid.simps(1) outer-stable-sublead)

```

```

lemma order-sublead:
  assumes h1:⟨J1 ⊆ J2⟩ and h2:⟨ideal J1 FPS-ring⟩ and h3:⟨ideal J2 FPS-ring⟩
  shows ⟨sublead-coeff-set J1 k ⊆ sublead-coeff-set J2 k⟩
  unfolding sublead-coeff-set-def subdeg-poly-set
  using h1 by blast

```

```

lemma sup-sublead-stable-add:⟨ideal I FPS-ring ⇒
  a ∈ ⋃ (range (sublead-coeff-set I)) ⇒
  b ∈ ⋃ (range (sublead-coeff-set I))
  ⇒ a ⊗add-monoid R b ∈ ⋃ (range (sublead-coeff-set I))⟩
proof -
  have f2:⟨x≠0 ∧ x∈subdeg-poly-set I k ⇒ subdegree x = k⟩ for x k
    unfolding subdeg-poly-set by auto
  then have f1:⟨x≠0 ∧ x∈subdeg-poly-set I k ⇒ fps-nth x k ∈ sublead-coeff-set I
  k⟩ for x k
    unfolding sublead-coeff-set-def by blast
  assume h1:⟨ideal I FPS-ring⟩ ⟨a ∈ ⋃ (range (sublead-coeff-set I))⟩
  ⟨b ∈ ⋃ (range (sublead-coeff-set I))⟩
  then obtain x x' k k'
    where f0:⟨a = fps-nth x k ∧ x∈subdeg-poly-set I k ∧ b = fps-nth x' k' ∧
  x'∈subdeg-poly-set I k'⟩
  unfolding sublead-coeff-set-def apply(safe)
  by (metis (mono-tags, lifting) Un-def mem-Collect-eq subdeg-poly-set)
  have p1:⟨k=k' ⇒ a≠0 ⇒ b≠0 ⇒ a∈sublead-coeff-set I k ∧ b ∈ sublead-coeff-set
  I k⟩
  using h1 f0 f1 fps-nonzero-nth by blast

```

**have**  $f3: \langle k < k' \implies a \neq 0 \implies b \neq 0 \implies \text{fps-}X^{\wedge}(k'-k)*x \in I \rangle$   
**by** (*metis* (*no-types*, *lifting*) *Formal-Power-Series-Ring.FPS-ring-def UNIV-I Un-iff*  
*additive-subgroup.zero-closed f0 h1(1) ideal-axioms-def ideal-def mem-Collect-eq monoid.simps(1)*  
*partial-object.select-conv(1) ring.simps(1) singletonD subdeg-poly-set*)  
**then have**  $f4: \langle k < k' \implies a \neq 0 \implies b \neq 0 \implies \text{subdegree}(\text{fps-}X^{\wedge}(k'-k)*x) = k' \rangle$   
**by** (*metis*  $f2$  *add-diff-inverse-nat f0 fps-nonzero-nth fps-subdegree-mult-fps-X-power(1)*  
*less-numeral-extra(3) nat-diff-split-asm zero-less-diff*)  
**then have**  $f5: \langle k < k' \implies a \neq 0 \implies b \neq 0 \implies (\text{fps-}X^{\wedge}(k'-k)*x) \in \text{subdeg-poly-set } I \ k' \rangle$   
**unfolding** *subdeg-poly-set using f3 by auto*  
**then have**  $f6: \langle k < k' \implies a \neq 0 \implies b \neq 0 \implies \text{fps-nth}((\text{fps-}X^{\wedge}(k'-k)*x) + x') \ k' \in \bigcup (\text{range}(\text{sublead-coeff-set } I)) \rangle$   
**by** (*metis*  $R$  *UNIV-I UN-iff add-stable-sublead f0 f1 fps-add-nth fps-mult-fps-X-power-nonzero(1)*  
*fps-zero-nth h1(1) monoid.simps(1) ring-record-simps(12)*)  
**have**  $f7: \langle b \neq 0 \implies k < k' \implies a = -b \implies \exists r. 0 \in \text{sublead-coeff-set } I \ r \rangle$   
**by** (*metis*  $R$  *additive-subgroup.zero-closed h1(1) ideal-def ring.simps(1) sublead-ideal*)  
**have**  $f8: \langle a \neq 0 \implies b \neq 0 \implies k < k' \implies a \neq -b \implies \exists r. a + b \in \text{sublead-coeff-set } I \ r \rangle$   
**by** (*metis* *UN-E add-diff-cancel-left' add-less-same-cancel2 diff-add-inverse2 f0 f6*  
*fps-X-power-mult-nth fps-add-nth less-imp-add-positive not-less-zero*)  
**then have**  $p2: \langle a \neq 0 \implies b \neq 0 \implies k < k' \implies \exists r. a \oplus b \in \text{sublead-coeff-set } I \ r \rangle$   
**apply** (*cases*  $\langle a = -b \rangle$ )  
**by** (*auto simp: R FPS-ring-def f7*)  
**have**  $f3': \langle k' < k \implies a \neq 0 \implies b \neq 0 \implies \text{fps-}X^{\wedge}(k-k')*x' \in I \rangle$   
**by** (*metis* (*no-types*, *lifting*) *Formal-Power-Series-Ring.FPS-ring-def UNIV-I Un-iff*  
*additive-subgroup.zero-closed f0 h1(1) ideal-axioms-def ideal-def mem-Collect-eq monoid.simps(1)*  
*partial-object.select-conv(1) ring.simps(1) singletonD subdeg-poly-set*)  
**then have**  $f4': \langle k' < k \implies a \neq 0 \implies b \neq 0 \implies \text{subdegree}(\text{fps-}X^{\wedge}(k-k')*x') = k \rangle$   
**by** (*metis*  $f2$  *add-diff-inverse-nat f0 fps-nonzero-nth fps-subdegree-mult-fps-X-power(1)*  
*less-numeral-extra(3) nat-diff-split-asm zero-less-diff*)  
**then have**  $f5': \langle k' < k \implies a \neq 0 \implies b \neq 0 \implies (\text{fps-}X^{\wedge}(k-k')*x') \in \text{subdeg-poly-set } I \ k \rangle$   
**unfolding** *subdeg-poly-set using f3' by auto*  
**then have**  $f6': \langle k' < k \implies a \neq 0 \implies b \neq 0 \implies \text{fps-nth}((\text{fps-}X^{\wedge}(k-k')*x') + x) \ k \in \bigcup (\text{range}(\text{sublead-coeff-set } I)) \rangle$   
**by** (*metis*  $R$  *UNIV-I UN-iff add-stable-sublead f0 f1 fps-add-nth fps-mult-fps-X-power-nonzero(1)*  
*fps-zero-nth h1(1) monoid.simps(1) ring-record-simps(12)*)  
**have**  $f7': \langle b \neq 0 \implies k' < k \implies a = -b \implies \exists r. 0 \in \text{sublead-coeff-set } I \ r \rangle$



**by** (*metis*  $R$  *additive-subgroup.zero-closed*  $h1(1)$  *ideal-def* *ring.simps(1)* *sublead-ideal*)  
**have**  $f8': \langle a \neq 0 \implies b \neq 0 \implies k' < k \implies a \neq -b \implies \exists r. a + b \in \text{sublead-coeff-set } I \ r \rangle$   
**by** (*metis* (*no-types, lifting*) *UN-iff*  $f8$  *add.commute* *add-diff-cancel-right'* *add-diff-inverse-nat*)  
  
 $f0$   $f4'$   $f6'$  *fps-X-power-mult-nth* *fps-add-nth* *not-less-zero* *nth-subdegree-zero-iff* *subdegree-0*)  
**then have**  $p3: \langle a \neq 0 \implies b \neq 0 \implies k' < k \implies \exists r. a \oplus b \in \text{sublead-coeff-set } I \ r \rangle$   
**apply** (*cases*  $\langle a = -b \rangle$ )  
**by** (*auto* *simp:R* *FPS-ring-def*  $f7'$ )  
**have** *cases:  $\langle k = k' \vee k < k' \vee k' < k \rangle$*   
**by** *auto*  
**then show** *?thesis*  
**apply** (*cases*  $\langle a = 0 \vee b = 0 \rangle$ )  
**using**  $R$   $h1(2)$   $h1(3)$  **apply** *force*  
**using** *Formal-Power-Series-Ring.add-stable-sublead*  $R$   $h1(1)$   $p1$   $p2$   $p3$  **by** (*force*)

qed

**lemma** *sup-sublead-ideal:  $\langle \text{ideal } I \text{ FPS-ring} \implies \text{ideal } (\bigcup k. \text{sublead-coeff-set } I \ k) \ R \rangle$*   
**apply** (*rule* *idealI*)  
**apply** (*simp* *add: ring-R*)  
**apply** (*rule* *group.subgroupI*)  
**using** *abelian-group.a-group* *ring.is-abelian-group* *ring-R* **apply** *blast*  
**apply** (*simp* *add: R*)  
**using** *non-empty-sublead* **apply** *force*  
**using** *subdeg-inv-in-sublead* **apply** *force*  
**using** *sup-sublead-stable-add* **apply** *force*  
**apply** (*metis* *UN-iff* *outer-stable-sublead*)  
**by** (*metis* *UN-iff* *ideal.I-r-closed* *sublead-ideal*)

**lemma** *Sub-subdeg-eq-ideal:  $\langle \text{ideal } J \text{ FPS-ring} \implies (\bigcup k. \text{subdeg-poly-set } J \ k) = J \rangle$*   
**unfolding** *subdeg-poly-set* **apply** (*safe*)  
**apply** (*metis* *Formal-Power-Series-Ring.FPS-ring-def* *additive-subgroup.zero-closed* *ideal.axioms(1)* *ring.simps(1)*)  
**by** *auto*

**lemma** *eq-subdeg:*  
**assumes**  $h1: \langle J1 \subseteq J2 \rangle$   
**and**  $h3: \langle \text{ideal } J1 \text{ FPS-ring} \rangle$  **and**  $h4: \langle \text{ideal } J2 \text{ FPS-ring} \rangle$   
**shows**  $\langle J1 = J2 \iff (\forall k. \text{subdeg-poly-set } J1 \ k = \text{subdeg-poly-set } J2 \ k) \rangle$   
**proof** –  
**have**  $\langle \forall k. \text{subdeg-poly-set } J1 \ k = \text{subdeg-poly-set } J2 \ k \implies (\bigcup k. \text{subdeg-poly-set } J1 \ k) = (\bigcup k. \text{subdeg-poly-set } J2 \ k) \rangle$

```

    by auto
  then have f0:⟨∀ k. subdeg-poly-set J1 k = subdeg-poly-set J2 k ⟹ J1 = J2⟩
    by (metis Sub-subdeg-eq-ideal h3 h4)
  have f1:⟨J1 = J2 ⟹ ∀ k. subdeg-poly-set J1 k = subdeg-poly-set J2 k⟩
    unfolding subdeg-poly-set by auto
  then show ?thesis using f0 f1 by auto
qed

```

```

lemma included-sublead:⟨ideal I FPS-ring ⟹ sublead-coeff-set I k ⊆ sublead-coeff-set
I (k+1)⟩
  unfolding sublead-coeff-set-def subdeg-poly-set
proof (safe)
  fix x a
  assume ⟨ideal I local.FPS-ring⟩
    ⟨a ∈ I⟩
    ⟨k = subdegree a ⟩
  show ⟨∃ aa. fps-nth a (subdegree a) = fps-nth aa (subdegree aa)
    ∧ aa ∈ {aa ∈ I. subdegree aa = subdegree a + 1} ∪ {0}⟩
    apply (rule exI [where x = ⟨fps-X * a⟩])
    by (simp add: subdegree-eq-0-iff ⟨a ∈ I⟩ ⟨ideal I local.FPS-ring⟩ mult-X-in-ideal) +
next
  show ⟨∃ a. fps-nth 0 (subdegree 0) = fps-nth a (subdegree a)
    ∧ a ∈ {a ∈ I. subdegree a = k + 1} ∪ {0}⟩
    by auto
qed

```

```

lemma included-sublead-gen: assumes ⟨ideal I FPS-ring⟩ ⟨k ≤ k'⟩
  shows ⟨sublead-coeff-set I k ⊆ sublead-coeff-set I (k')⟩
  using assms
  apply (induct ⟨k' - k⟩)
  apply simp
  by (metis Suc-eq-plus1 included-sublead lift-Suc-mono-le)

```

```

lemma sup-sublead:
  assumes h1:⟨ideal I FPS-ring⟩
    and h2:⟨noetherian-ring R⟩
  shows ⟨(⋃ {sublead-coeff-set I k | k. k ∈ UNIV}) ∈ {sublead-coeff-set I k | k. k ∈ UNIV}⟩
  apply (rule noetherian-ring.ideal-chain-is-trivial [OF h2, of ⟨{sublead-coeff-set I
k | k. k ∈ UNIV}⟩])
  apply blast
  unfolding subset-chain-def using included-sublead-gen
  by (auto simp add: h1 sublead-ideal) (meson h1 in-mono linorder-linear)

```

```

lemma subdeg-inf-imp-s-tendsto-zero:
  fixes s::⟨nat ⇒ 'a::⟨idom⟩ fps⟩
  assumes g2:⟨strict-mono (λ n. subdegree (s n))⟩
  shows ⟨s ⟶ 0⟩
proof -
  have g1:⟨(λ x. 1/x) ⟶ 0⟩

```

```

    using lim-1-over-n by force
  have ⟨ $\forall n. \exists k. n < \text{subdegree } (s \ k)$ ⟩
    by (metis dual-order.strict-trans g2 gt-ex linorder-not-le
        nat-neq-iff strict-mono-imp-increasing)
  have r1:⟨ $r > 0 \implies (n::\text{nat}) > \log 2 (1/r) \implies (1/2^n < r) \iff 2^n > 1/r$ ⟩ for
n r
    by(auto simp:field-simps)
  have r2:⟨ $r > 0 \implies (n::\text{nat}) > \log 2 (1/r) \implies 2^n > 2^{\text{powr } (\log 2 (1/r))}$ ⟩ for
n r
    by (simp add: log-less-iff powr-realpow)
  then have r3:⟨ $r > 0 \implies (n::\text{nat}) > \log 2 (1/r) \implies 2^{\text{powr } (\log 2 (1/r))} =$ 
 $1/r$ ⟩ for r n
    by auto
  then have r4:⟨ $r > 0 \implies (n::\text{nat}) > \log 2 (1/r) \implies 2^n > 1/r$ ⟩ for r n
    using ⟨ $\bigwedge r n. \llbracket 0 < r; \log 2 (1/r) < \text{real } n \rrbracket \implies 2^{\text{powr } \log 2 (1/r)} < 2^n$ ⟩
    by force
  then have r5:⟨ $r > 0 \implies (n::\text{nat}) > \log 2 (1/r) \implies 1/2^n < r$ ⟩ for r n
    using ⟨ $\bigwedge r n. \llbracket 0 < r; \log 2 (1/r) < \text{real } n \rrbracket \implies (1/2^n < r) = (1/r <$ 
 $2^n)$ ⟩ by blast
  have ⟨ceiling (r::real)  $\geq r$ ⟩ for r
    by simp
  then have r6:⟨ $r > 0 \implies (\text{ceiling } (\log 2 (1/r))) \geq \log 2 (1/r)$ ⟩ for r
    by auto
  then have r7:⟨ $r > 0 \implies (n::\text{nat}) > (\text{ceiling } (\log 2 (1/r))) \implies 1/2^n < r$ ⟩ for
r n
    by (metis ⟨ $\bigwedge r n. \llbracket 0 < r; \log 2 (1/r) < \text{real } n \rrbracket \implies 1/2^n < r$ ⟩
        ceiling-less-cancel ceiling-of-nat)
  have r8:⟨ $r > 0 \implies \exists n::\text{nat}. n > (\log 2 (1/r))$ ⟩ for r
    by (simp add: reals-Archimedean2)
  have r9:⟨ $\forall (r::\text{real}) > 0. \exists n0. \forall n \geq n0. 1/2^n < r$ ⟩
  proof(safe)
    fix r::real
    assume ⟨ $r > 0$ ⟩
    then obtain n::nat where ⟨ $n > (\log 2 (1/r))$ ⟩ using r8 by blast
    show ⟨ $\exists n0. \forall n \geq n0. 1/2^n < r$ ⟩
      apply(rule exI[where x=n])
      using ⟨ $0 < r$ ⟩ ⟨ $\log 2 (1/r) < \text{real } n$ ⟩ r5 by auto
  qed
  show t2:⟨ $s \longrightarrow 0$ ⟩
  proof(rule metric-LIMSEQ-I)
    fix r::real
    assume ⟨ $0 < r$ ⟩
    then obtain n where ⟨ $n > 0 \wedge 1/2^n < r$ ⟩ using r9
      by (metis gr0I less-or-eq-imp-le zero-less-numeral)
    then have ⟨ $\forall k \geq n. \text{inverse } (2^k) < r$ ⟩
      by (smt (verit, ccfv-threshold) inverse-eq-divide
          inverse-less-iff-less power-increasing-iff zero-less-power)
    then obtain n1 where ⟨ $1/2^{(\text{subdegree } (s \ n1))} < r \wedge n1 > 0$ ⟩
      by (metis ⟨ $\forall n. \exists k. n < \text{subdegree } (s \ k)$ ⟩ bot-nat-0.not-eq-extremum di-

```

```

vide-inverse
  mult-1 nle-le not-less-iff-gr-or-eq order-less-le-trans)
  then have ⟨dist (s n1) 0 < r⟩
    by (simp add: ⟨0 < r⟩ dist-fps-def inverse-eq-divide)
  then show ⟨∃ no. ∀ n ≥ no. dist (s n) 0 < r⟩
    apply (intro exI [where x = n1])
    apply (safe) using g2
    unfolding dist-fps-def strict-mono-def
    using power-strict-increasing-iff [of 2 ⟨subdegree (s n1)⟩] inverse-eq-divide
inverse-le-iff-le
  by (smt (verit) ⟨0 < r⟩ ⟨1 / 2 ^ subdegree (s n1) < r ∧ 0 < n1⟩
    diff-zero le-eq-less-or-eq power-less1-D)
qed
qed

```

```

lemma idl-sum: ⟨finite A ⟹ ideal {x. ∃ s. x = (∑ i ∈ {0..<card A}. s i * from-nat-into
A i)} R⟩ for A
proof (rule idealI)
  assume ⟨finite A⟩
  show ⟨ring R⟩
    using ring-R by (simp)
  show ⟨subgroup {x. ∃ s. x = (∑ i = 0..<card A. s i * from-nat-into A i)}
(add-monoid R)⟩
  proof (rule group.subgroupI)
    show ⟨Group.group (add-monoid R)⟩
      using abelian-group.a-group ring.is-abelian-group ring-R by blast
    show ⟨{x. ∃ s. x = (∑ i = 0..<card A. s i * from-nat-into A i)} ⊆ carrier
(add-monoid R)⟩
      by (simp add: R)
    show ⟨{x. ∃ s. x = (∑ i = 0..<card A. s i * from-nat-into A i)} ≠ {}⟩
      by blast
  next
    fix a assume ⟨a ∈ {x. ∃ s. x = (∑ i = 0..<card A. s i * from-nat-into A i)}⟩
    then show ⟨inv add-monoid R a ∈ {x. ∃ s. x = (∑ i = 0..<card A. s i *
from-nat-into A i)}⟩
      proof (safe)
        fix s
        have p6: ⟨(THE y. (∑ i = 0..<card A. s i * from-nat-into A i) + y = 0 ∧ y
+
(∑ i = 0..<card A. s i * from-nat-into A i) = 0)
= (∑ i = 0..<card A. - (s i * from-nat-into A i))⟩ for A::⟨'a set⟩ and s
          using theI [of ⟨λ y. (∑ i = 0..<card A. s i * from-nat-into A i) + y = 0 ∧
y +
(∑ i = 0..<card A. s i * from-nat-into A i) = 0⟩]
          by (smt (verit, best) add.commute add.right-inverse add-left-imp-eq sum.cong
sum-negf)

```

```

    assume ⟨a = (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i)⟩
    then show ⟨∃ sa. invadd-monoid R (∑ i = 0.. $\text{card } A$ . s i *
from-nat-into A i) = (∑ i = 0.. $\text{card } A$ . sa i * from-nat-into A i)⟩
      apply(intro exI[where x=⟨-s⟩])
      by(auto simp add:m-inv-def p6 R)
    qed
  next
  fix a b
  assume ⟨a ∈ {x. ∃ s. x = (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i)}⟩
    ⟨b ∈ {x. ∃ s. x = (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i)}⟩
  then show ⟨a ⊗add-monoid R b ∈ {x. ∃ s. x = (∑ i = 0.. $\text{card } A$ . s i *
from-nat-into A i)}⟩
    proof(safe)
      fix s sa
      assume ⟨a = (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i)⟩
        ⟨b = (∑ i = 0.. $\text{card } A$ . sa i * from-nat-into A i)⟩
      then show ⟨∃ sb. (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i) ⊗add-monoid R
(∑ i = 0.. $\text{card } A$ . sa i * from-nat-into A i) = (∑ i = 0.. $\text{card } A$ . sb i *
from-nat-into A i)⟩
        apply(intro exI[where x=⟨λi. s i + sa i⟩])
        by(simp add:R comm-semiring-class.distrib sum.distrib)
      qed
    qed
  next
  fix a x
  assume ⟨finite A⟩ ⟨a ∈ {x. ∃ s. x = (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A
i)}⟩ ⟨x ∈ carrier R⟩
  then show ⟨x ⊗ a ∈ {x. ∃ s. x = (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i)}⟩
    ⟨a ⊗ x ∈ {x. ∃ s. x = (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i)}⟩
  proof(safe)
    fix s
    assume ⟨a = (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i)⟩
    then show
      ⟨∃ sa. x ⊗ (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i) = (∑ i = 0.. $\text{card } A$ .
sa i * from-nat-into A i)⟩
      apply(intro exI[where x=⟨(λi. x * s i)⟩])
      by(simp add:R comm-semiring-class.distrib sum.distrib mult.assoc sum-distrib-left)

    show
      ⟨∃ sa. (∑ i = 0.. $\text{card } A$ . s i * from-nat-into A i) ⊗ x = (∑ i = 0.. $\text{card } A$ .
sa i * from-nat-into A i)⟩
      apply(intro exI[where x=⟨(λi. x * s i)⟩])
      by(simp add:R comm-semiring-class.distrib sum.distrib mult.assoc
sum-distrib-left mult.left-commute mult commute)
    qed
  qed

```

**lemma** *genideal-sum-rep*:

⟨finite A ⟹ genideal R A = {x. ∃ s. x=(∑ i∈{0.. $\text{card } A$ }. s i \* from-nat-into

```

A i)} for A
proof(subst set-eq-subset, rule conjI)
  assume hr:⟨finite A⟩
  then have unq:⟨x∈A ⇒ ∃!i. i<card A ∧ from-nat-into A i = x⟩ for x
    using bij-betw-from-nat-into-finite[of A, OF ⟨finite A⟩]
    unfolding bij-betw-def inj-on-def
    by (smt (verit, ccfv-threshold) ⟨bij-betw (from-nat-into A) {..<card A} A⟩
        bij-betw-iff-bijections lessThan-iff)
  have ⟨A≠{⟩ ⇒ (card ({0..<card A} ∩ {i. from-nat-into A i = x})) = 1⟩ if
hh:⟨x∈A⟩ for x
  proof(rule ccontr)
    assume hhh:⟨card ({0..<card A} ∩ {i. from-nat-into A i = x}) ≠ 1⟩ ⟨A≠{⟩
    then have jm:
      ⟨card ({0..<card A} ∩ {i. from-nat-into A i = x}) > 1 ⇒ ∃ i1 i2. i1≠i2 ∧
i1 < card A
∧ i2 < card A ∧ from-nat-into A i1 = from-nat-into A i2 ∧ from-nat-into A i1 =
x⟩
    by (smt (verit, ccfv-SIG) Int-Collect One-nat-def atLeastLessThan-iff card-le-Suc0-iff-eq
        finite-Int finite-atLeastLessThan linorder-not-less n-not-Suc-n)
    then have ⟨card ({0..<card A} ∩ {i. from-nat-into A i = x}) > 1⟩ using hhh
hr
    by (metis (mono-tags, lifting) Int-def atLeastLessThan-iff card-eq-0-iff emptyE
        finite-Int
        finite-atLeastLessThan le0 less-one linorder-neqE-nat mem-Collect-eq that
        unq)
    then show False using jm unq[OF hh] by(auto)
  qed
  then have ⟨A⊆{x. ∃ s. x = (∑ i = 0..<card A. s i * from-nat-into A i)}⟩
  proof(safe)
    fix x
    assume hhhh:⟨(∧x. x ∈ A ⇒ A ≠ {⟩ ⇒ card ({0..<card A} ∩ {i. from-nat-into
A i = x}) = 1)⟩
      ⟨x∈A⟩
    then have ⟨of-nat (card ({0..<card A} ∩ {xa. from-nat-into A xa = x})) = 1⟩
      by (metis One-nat-def card.empty less-nat-zero-code of-nat-1 unq)
    with hhhh show ⟨∃ s. x = (∑ i = 0..<card A. s i * from-nat-into A i)⟩
      apply(cases ⟨x=0⟩)
      apply(rule exI[where x=⟨λi. 0⟩])
      apply(simp)
      apply(rule exI[where x=⟨λi. if from-nat-into A i = x then 1 else 0⟩])
      apply(subst if-distrib[where f=⟨λx. x*a⟩ for a])
      apply(subst sum.If-cases)
      by(simp)+
  qed
  then show ⟨Idl A ⊆ {x. ∃ s. x = (∑ i = 0..<card A. s i * from-nat-into A i)}⟩
    unfolding genideal-def using idl-sum[OF hr] by(auto)
  show ⟨{x. ∃ s. x = (∑ i = 0..<card A. s i * from-nat-into A i)} ⊆ Idl A ⟩
  proof(safe)
    fix x and s::⟨nat⇒'a⟩

```

```

have a:⟨∀ i < card A. from-nat-into A i ∈ A⟩
  by (metis card.empty from-nat-into less-nat-zero-code)
then have b:⟨∀ i. s i ∈ carrier R⟩
  using R by auto
then have ⟨∀ i < card A. s i * from-nat-into A i ∈ genideal R A⟩
  using ring.genideal-ideal[OF ring-R, of A] ideal.I-l-closed[of - R ]
  by (metis R a ideal-def monoid.simps(1) partial-object.select-convs(1)
      ring.genideal-self subsetD subset-UNIV)
have ff:⟨A ⊆ carrier R⟩ by (simp add:R)
then have ⟨∀ i < n. g i ∈ genideal R A ⟹ (∑ i ∈ {0..<n}. g i) ∈ genideal R
A⟩
  for g:⟨nat ⇒ 'a⟩ and n
  apply(induct n)
  apply (metis R additive-subgroup.zero-closed atLeastLessThan-iff
      ideal-def not-less-zero ring.genideal-ideal ring.simps(1) ring-R sum.neutral)
  using ring.genideal-ideal[OF ring-R, of A, OF ⟨A ⊆ carrier R⟩]
      additive-subgroup.a-closed[of ⟨genideal R A⟩ R - -] unfolding ideal-def
by(auto simp:R)
  then show ⟨(∑ i = 0..<card A. s i * from-nat-into A i) ∈ Idl A⟩
  using ⟨∀ i < card A. s i * from-nat-into A i ∈ Idl A⟩ by presburger
qed
qed

```

**lemma** *fps-sum-rep-nth'*:

```

fps-nth (sum (λi. fps-const(a i)*fps-X^i) {0..m}) n = (if n ≤ m then a n else 0)
  by (simp add: fps-sum-nth if-distrib cong del: if-weak-cong)

```

**lemma** *abs-tndsto*: **shows**  $\langle (\lambda n. (\sum i \leq n. \text{fps-const } (s i) * \text{fps-X}^i)::'a \text{ fps}) \longrightarrow \text{Abs-fps } s \rangle$

(is  $\langle ?s \longrightarrow ?a \rangle$ )

**proof** –

**have**  $\exists n0. \forall n \geq n0. \text{dist } (?s n) ?a < r$  **if**  $r > 0$  **for**  $r$

**proof** –

**obtain**  $n0$  **where**  $n0: (1/2)^{n0} < r$   $n0 > 0$

**using** *reals-power-lt-ex*[OF  $\langle r > 0 \rangle$ , of 2] **by** *auto*

**show** *?thesis*

**proof** –

**have**  $\text{dist } (?s n) ?a < r$  **if**  $nn0: n \geq n0$  **for**  $n$

**proof** –

**from that have**  $thnn0: (1/2)^n \leq (1/2)^{n0}$  **for**  $n \geq n0$

**by** (*simp add: field-split-simps*)

**show** *?thesis*

**proof** (*cases ?s n = ?a*)

**case** *True*

**then show** *?thesis*

**unfolding** *metric-space.dist-eq-0-iff*

**using**  $\langle r > 0 \rangle$  **by** (*simp del: dist-eq-0-iff*)

**next**

```

case False
from False have dth:  $\text{dist } (?s \ n) \ ?a = (1/2)^{\wedge} \text{subdegree } (?s \ n - \ ?a)$ 
  by (simp add: dist-fps-def field-simps)
from False have kn:  $\text{subdegree } (?s \ n - \ ?a) > n$ 
  apply (intro subdegree-greaterI) apply (simp-all add: fps-sum-rep-nth')
  by (metis (full-types) atLeast0AtMost fps-sum-rep-nth')
then have  $\text{dist } (?s \ n) \ ?a < (1/2)^{\wedge} n$ 
  by (simp add: field-simps dist-fps-def)
also have  $\dots \leq (1/2)^{\wedge} n0$ 
  using nn0 by (simp add: field-split-simps)
also have  $\dots < r$ 
  using n0 by simp
  finally show ?thesis .
qed
qed
then show ?thesis by blast
qed
qed
then show ?thesis
  unfolding lim-sequentially by blast
qed

lemma add-stable-FPS-ring:  $\langle \text{ideal } I \text{ FPS-ring} \implies a \in I \implies b \in I \implies a + b \in I \rangle$ 
  unfolding FPS-ring-def
  by (metis additive-subgroup.a-closed ideal.axioms(1) ring-record-simps(12))

lemma abs-tndsto-le: shows  $\langle (\lambda n. (\sum i < n. \text{fps-const } (s \ i) * \text{fps-}X^{\wedge}i)::'a \ \text{fps})$ 
   $\longrightarrow \text{Abs-fps } s \rangle$ 
  using LIMSEQ-lessThan-iff-atMost abs-tndsto by blast

lemma bij-betw-strict-mono:
  assumes  $\langle \text{strict-mono } (f::\text{nat} \Rightarrow \text{nat}) \rangle$ 
  shows  $\langle \text{bij-betw } f \ \text{UNIV} \ (f' \ \text{UNIV}) \rangle$ 
  by (simp add: assms bij-betw-imageI strict-mono-on-imp-inj-on)

lemma no-i-inf-0:  $\langle \text{strict-mono } (f::\text{nat} \Rightarrow \text{nat}) \implies i < f \ 0 \implies \neg(\exists j. f \ j = i) \rangle$ 
  by (auto simp add: strict-mono-less)

lemma inter-mt:  $\langle \text{strict-mono } (f::\text{nat} \Rightarrow \text{nat}) \implies \{.. < f \ 0\} \cap \text{range } f = \{\} \rangle$ 
  by (metis Int-emptyI lessThan-iff no-i-inf-0 rangeE)

lemma range-inter-f:  $\langle \text{strict-mono } (f::\text{nat} \Rightarrow \text{nat}) \implies \{.. < f \ n\} \cap \text{range } f = f' \{0.. < n\} \rangle$ 
  apply (induct n)
  apply (simp add: inter-mt)
  by (auto simp: strict-mono-less strict-monoD)

lemma simp-rule-sum:  $\langle \text{strict-mono } (f::\text{nat} \Rightarrow \text{nat}) \implies (\sum i \in \{.. < f \ (\text{Suc } n)\}. \text{if } i$ 

```



$\in \text{range } f$   
 then  $(s ((\text{inv-into UNIV } f) i)) * \text{fps-}X^{\wedge}i \text{ else } 0) = (\sum_{i \in \{..<f n\}}. (\text{if } i \in \text{range } f \text{ then } (s ((\text{inv-into UNIV } f) i)) * \text{fps-}X^{\wedge}i \text{ else } 0)) + (s ((\text{inv-into UNIV } f) (f n))) * \text{fps-}X^{\wedge}(f n)$   
**proof** –  
 assume  $h1 : \langle \text{strict-mono } f \rangle$   
 have  $f0 : \langle \forall i \in \{f n .. <f (Suc n)\}. (\text{if } i \in \text{range } f \text{ then } (s ((\text{inv-into UNIV } f) i)) * \text{fps-}X^{\wedge}i \text{ else } 0) = 0 \rangle$   
 by  $(\text{metis greaterThanLessThan-iff } h1 \text{ not-less-eq rangeE strict-mono-less})$   
 then have  $s : \langle \{..<f (Suc n)\} = \{..f n\} \cup \{f n .. <f (Suc n)\} \rangle$   
 by  $(\text{metis } h1 \text{ ivl-disj-un-one(1) strict-mono-Suc-iff})$   
 show  $?thesis$   
 apply  $(\text{subst } s)$   
 apply  $(\text{subst sum.union-disjoint})$   
 apply  $(\text{auto})[3]$   
 using  $f0$  apply  $(\text{simp})$   
 by  $(\text{smt } (\text{verit}, \text{ccfv-SIG}) \text{ lessThan-Suc-atMost rangeI sum.lessThan-Suc})$   
**qed**

**lemma rewriting-sum: assumes**  $\langle \text{strict-mono } (f :: \text{nat} \Rightarrow \text{nat}) \rangle$   
**shows**  $\langle (\sum_{i < n}. \text{fps-const } (s i) * \text{fps-}X^{\wedge}(f i)) = (\sum_{i \in \{..<f n\}}. (\text{if } i \in \text{range } f \text{ then } \text{fps-const } (s (\text{inv-into UNIV } f) i)) * \text{fps-}X^{\wedge}i \text{ else } 0) \rangle$   
**proof**  $(\text{induct } n)$   
 case 0  
 then show  $?case$   
 by  $(\text{simp add: assms inter-mt sum.If-cases})$   
**next**  
 case  $(\text{Suc } n)$   
 then show  $?case$   
 apply  $(\text{subst simp-rule-sum})$   
 by  $(\text{auto simp: assms strict-mono-on-imp-inj-on})$   
**qed**

**lemma exists-seq:**  $\langle \text{strict-mono } (f :: \text{nat} \Rightarrow \text{nat}) \rangle \implies$   
 $\exists s. (\sum_{i \in \{..<f n\}}. (\text{if } i \in \text{range } f \text{ then } \text{fps-const } (s' (\text{inv-into UNIV } f) i)) * \text{fps-}X^{\wedge}i \text{ else } 0)$   
 $= (\sum_{i \in \{..<f n\}}. \text{fps-const } (s i) * \text{fps-}X^{\wedge}i)$   
**apply**  $(\text{rule exI}[\text{where } x = \lambda i. (\text{if } i \in \text{range } f \text{ then } (s' ((\text{inv-into UNIV } f) i)) \text{ else } 0) \ ]])$   
**using**  $\text{rewriting-sum}$   
**by**  $(\text{smt } (\text{verit}, \text{best}) \text{ fps-const-0-eq-0 lambda-zero sum.cong})$

**lemma exists-seq':**  $\langle \text{strict-mono } (f :: \text{nat} \Rightarrow \text{nat}) \rangle \implies$   
 $\exists s. (\sum_{i < n}. \text{fps-const } (s' i) * (\text{fps-}X^{\wedge} : : 'a \text{ fps})^{\wedge}(f i)) =$   
 $(\sum_{i \in \{..<f n\}}. \text{fps-const } (s i) * \text{fps-}X^{\wedge}i)$

```

apply(subst rewriting-sum[])
using exists-seq[of f ‹ $\lambda i. (s' i)$ ›]
by(auto)

```

```

lemma exists-seq-all:‹strict-mono (f::nat⇒nat) ⇒⇒
  ∃ s. ∀ n. (∑ i∈{.. $f n$ }. (if i ∈ range f then fps-const (s' (inv-into UNIV f i))
  *fps-X∧i else 0))
  = (∑ i∈{.. $f n$ }. fps-const (s i) *fps-X∧i)›
  apply(rule exI[where x=‹ $\lambda i. (if i ∈ range f then (s' ((inv-into UNIV f) i)) else 0)$ ›])
  using rewriting-sum
  by (smt (verit, best) fps-const-0-eq-0 lambda-zero sum.cong)

```

```

lemma exists-seq-all':‹strict-mono (f::nat⇒nat) ⇒⇒
  ∃ s. ∀ n. (∑ i<n. fps-const (s' i) *fps-X∧(f i)) =
  (∑ i∈{.. $f n$ }. fps-const (s i) *fps-X∧i)›
  apply(subst rewriting-sum)
  using exists-seq-all[of f ‹ $\lambda i. (s' i)$ ›]
  by(auto)

```

```

lemma tendsto-f-seq:assumes ‹strict-mono (f::nat⇒nat)›
  shows ‹ $(\lambda n. (\sum i \in \{.. $f n$ \}. fps-const (s i) *fps-X∧i)::'a fps) \longrightarrow Abs-fps (\lambda i. s i)$ ›
  using fps-notation LIMSEQ-subseq-LIMSEQ[OF abs-tndsto-le[of s], of f] assms
  by(auto simp:o-def)

```

```

lemma LIMSEQ-add-fps:
  fixes x y :: 'a::idom fps
  assumes f:f \longrightarrow x and g:(g \longrightarrow y)
  shows (( $\lambda x. f x + g x$ ) \longrightarrow x + y)
proof –
  from f have ‹ $\forall e>0. \exists n. \forall j \geq n. dist (f j) x < e/2$ ›
    using lim-sequentially
    using half-gt-zero by blast
  from g have f0:‹ $\forall e>0. \exists n. \forall j \geq n. dist (g j) y < e/2$ ›
    using lim-sequentially half-gt-zero by blast
  have f4:‹ $dist (f j - x) 0 = dist (f j) x$ › for j
    unfolding dist-fps-def by(auto)
  have f5:‹ $dist (g j - y) 0 = dist (g j) y$ › for j
    by (metis diff-0-right dist-fps-def eq-iff-diff-eq-0)
  then have f0':‹ $dist (f j + g j) (x + y) = dist (f j - x + g j - y) 0$ › for j
    unfolding dist-fps-def
    by (auto simp add: add.commute add-diff-eq diff-diff-eq2)
  have f1:‹ $dist (f j - x + g j - y) 0 \leq \max (dist (f j - x) 0) (dist (g j - y) 0)$ ›
for j

```

```

unfolding dist-fps-def apply(auto simp:le-max-iff-disj field-simps)[1]
  by (metis (no-types, lifting) add-diff-add eq-iff-diff-eq-0 min-le-iff-disj subde-
gree-add-ge')
  then have f2: $\langle \text{dist } (f\ j - x + g\ j - y)\ 0 \leq \text{dist } (f\ j - x)\ 0 + \text{dist } (g\ j - y)\ 0 \rangle$ 
  for j
  by (smt (verit) zero-le-dist)
  from f0 have f3: $\langle \forall e > 0. \exists n. \forall j \geq n. \text{dist } (f\ j)\ x + \text{dist } (g\ j)\ y < e/2 + e/2 \rangle$ 
  by (metis  $\langle \forall e > 0. \exists n. \forall j \geq n. \text{dist } (f\ j)\ x < e / 2 \rangle$  add-strict-mono le-trans
linorder-le-cases)
  then show ?thesis
  unfolding LIMSEQ-def
  by (metis f0' f2 f4 f5 field-sum-of-halves order-le-less-trans)
qed

```

**lemma** *LIMSEQ-cmult-fps*:

```

fixes x y :: 'a::idom fps'
assumes f:f  $\longrightarrow$  x
shows ( $\lambda x. c * f\ x \longrightarrow c*x$ )
proof -
  from f have  $\langle \forall e > 0. \exists n. \forall j \geq n. \text{dist } (f\ j)\ x < e \rangle$ 
  using lim-sequentially
  using half-gt-zero by blast
  have  $\langle \text{dist } (c*f\ j - c*x)\ 0 = \text{dist } (c*(f\ j))\ (c*x) \rangle$  for j
  unfolding dist-fps-def by auto
  have  $\langle \forall i \leq n. \text{fps-nth } (f\ j)\ i = \text{fps-nth } x\ i \implies$ 
   $(\sum i = 0..n. \text{fps-nth } c\ i * \text{fps-nth } (f\ j)\ (n - i)) = (\sum i = 0..n. \text{fps-nth } c\ i * \text{fps-nth } x\ (n - i)) \rangle$ 
  for j n
  using diff-le-self by presburger
  have  $\langle c \neq 0 \implies \text{dist } (f\ j)\ x \geq \text{dist } (c*f\ j)\ (c*x) \rangle$  for j
  proof(cases  $\langle x = f\ j \rangle$ )
  case True
  then show ?thesis
  unfolding dist-fps-def subdegree-def
  by(auto)
  next
  case False
  then have rule-su: $\langle (\text{LEAST } n. \text{fps-nth } (f\ j)\ n \neq \text{fps-nth } x\ n) \leq$ 
   $(\text{LEAST } n. \text{fps-nth } (c * f\ j)\ n \neq \text{fps-nth } (c * x)\ n)$ 
   $\implies c \neq 0 \implies \text{dist } (c * f\ j)\ (c * x) \leq \text{dist } (f\ j)\ x \rangle$ 
  unfolding dist-fps-def subdegree-def by(auto)
  have f0: $\langle n < (\text{LEAST } n. \text{fps-nth } (f\ j)\ n \neq \text{fps-nth } x\ n) \implies$ 
   $(\sum i = 0..n. \text{fps-nth } c\ i * \text{fps-nth } (f\ j)\ (n - i)) = (\sum i = 0..n. \text{fps-nth } c\ i * \text{fps-nth } x\ (n - i)) \rangle$ 
  for n
  by (metis (mono-tags, lifting) less-imp-diff-less not-less-Least)
  have f1: $\langle \forall n. (\sum i = 0..n. \text{fps-nth } c\ i * \text{fps-nth } (f\ j)\ (n - i))$ 
   $= (\sum i = 0..n. \text{fps-nth } c\ i * \text{fps-nth } x\ (n - i)) \implies$ 

```

```

 $x \neq f j \implies c \neq 0 \implies (LEAST\ n.\ fps\text{-}nth\ (f\ j)\ n \neq fps\text{-}nth\ x\ n) \leq (LEAST\ n.\$ 
 $False)$ 
(is  $\langle ?P \implies ?R1 \implies ?R2 \implies ?R3 \rangle$ )
proof -
  assume  $a1: c \neq 0$ 
  assume  $a2: x \neq f j$ 
  assume  $\forall n. (\sum i = 0..n. fps\text{-}nth\ c\ i * fps\text{-}nth\ (f\ j)\ (n - i)) =$ 
 $(\sum i = 0..n. fps\text{-}nth\ c\ i * fps\text{-}nth\ x\ (n - i))$  (is  $?P$ )
  then show  $(LEAST\ n.\ fps\text{-}nth\ (f\ j)\ n \neq fps\text{-}nth\ x\ n) \leq (LEAST\ n.\ False)$ 
    using  $a2\ a1$  by (metis (no-types) fps-ext fps-mult-nth mult-cancel-left)
  qed
  have  $f2: \langle x \neq f j \implies c \neq 0 \implies (LEAST\ n.\ fps\text{-}nth\ (f\ j)\ n \neq fps\text{-}nth\ x\ n)$ 
 $\leq (LEAST\ n.\ (\sum i = 0..n. fps\text{-}nth\ c\ i * fps\text{-}nth\ (f\ j)\ (n - i)) \neq$ 
 $(\sum i = 0..n. fps\text{-}nth\ c\ i * fps\text{-}nth\ x\ (n - i))) \rangle$  (is  $\langle ?R1 \implies ?R2 \implies ?P1 \rangle$ )
  proof (cases  $\langle ?P \rangle$ )
    case True
      assume  $\langle x \neq f j \rangle \langle c \neq 0 \rangle$ 
      then show  $?thesis$  using True  $f1$  by (auto)
    next
      case False
        assume  $a1: c \neq 0$ 
        assume  $a2: x \neq f j$ 
        show  $?thesis$ 
        proof (insert False  $a1\ a2$ , rule ccontr)
          fix  $n$ 
          assume  $**:\langle \neg ?P \rangle$ 
          assume  $**:\langle \neg ?P1 \rangle$ 
          (is  $\langle \neg ?a \leq ?b \rangle$ )
          then have  $\langle fps\text{-}nth\ (f\ j)\ ?b = fps\text{-}nth\ (f\ j)\ ?b \rangle$ 
            by blast
          also have  $\langle (\sum i = 0..?b. fps\text{-}nth\ c\ i * fps\text{-}nth\ (f\ j)\ (?b - i))$ 
 $\neq (\sum i = 0..?b. fps\text{-}nth\ c\ i * fps\text{-}nth\ x\ (?b - i)) \rangle$ 
            using  $*$ 
            by (smt (verit, best)  $**\ LeastI\ sum.cong$ )
          thus False
          using  $*\ f0\ linorder\text{-}not\text{-}le$  by blast
        qed
      qed
    from False show  $?thesis$ 
    unfolding dist-fps-def subdegree-def
    by (simp add:  $f2\ fps\text{-}mult\text{-}nth$ )
  qed
  then show  $?thesis$ 
  unfolding LIMSEQ-def
  by (metis  $\langle \forall e > 0. \exists n. \forall j \geq n. dist\ (f\ j)\ x < e \rangle\ dist\text{-}self\ lambda\text{-}zero\ or\text{-}der\text{-}le\text{-}less\text{-}trans$ )
qed

```

### 6.3 The Hilbert Basis theorem

**theorem** *Hilbert-basis-FPS*:  
**assumes**  $h2:\langle \text{noetherian-ring } R \rangle$   
**shows**  $\langle \text{noetherian-ring FPS-ring} \rangle$   
**proof**(*rule ring.noetherian-ringI*)  
**show**  $\text{fst}:\langle \text{ring FPS-ring} \rangle$   
**by** (*simp add: ring-FPS*)  
**fix**  $I$   
**assume**  $h1:\langle \text{ideal } I \text{ FPS-ring} \rangle$   
**show**  $\langle \exists A \subseteq \text{carrier FPS-ring. finite } A \wedge I = \text{Idd}_{\text{FPS-ring}} A \rangle$   
**proof**(*cases*  $\langle I = \{0\} \vee I = \text{carrier FPS-ring} \rangle$ )  
**case** *True*  
**then show** *?thesis* **apply**(*safe*)  
**apply**(*rule exI[where x= $\{0\}$ ]*)  
**apply**(*simp add: genideal-def*)  
**using**  $h1$  *ideal.Icarr* **apply** *fastforce*  
**apply**(*rule exI[where x= $\{1\}$ ]*)  
**using** *ideal.I-l-closed* **by**(*fastforce simp:FPS-ring-def genideal-def*)  
**next**  
**case** *False*  
**have**  $f0:\langle \text{subset.chain } \{I. \text{ideal } I R\} \{(\text{sublead-coeff-set } I k) \mid k. k \in \text{UNIV}\} \rangle$   
**unfolding** *subset-chain-def* **using** *included-sublead-gen[OF h1]* *sublead-ideal[OF h1]*  
**by** (*smt (verit, ccfv-threshold) mem-Collect-eq nle-le subsetI*)  
**have**  $f2:\langle (\text{sublead-coeff-set } I k) \mid k. k \in \text{UNIV} \rangle \neq \{ \}$   
**using**  $h1$  **by**(*auto*)  
**have**  $\langle \text{genideal } R S = \text{genideal } R (S \cup \{0\}) \rangle$  **for**  $S$   
**unfolding** *genideal-def*  
**by** (*metis R Un-insert-right additive-subgroup.zero-closed ideal.axioms(1) insert-subset ring.simps(1) sup-bot-right*)  
**have**  $\langle (\bigcup k. \text{sublead-coeff-set } I k) \in \{ \text{sublead-coeff-set } I m \mid m. m \in \text{UNIV} \} \rangle$   
**by** (*smt (verit, best) Collect-cong full-SetCompr-eq h1 h2 image-iff mem-Collect-eq sup-sublead*)  
**then have**  $\langle \exists m. (\bigcup k. \text{sublead-coeff-set } I k) = \text{sublead-coeff-set } I m \rangle$  **by** *auto*  
**then obtain**  $m$  **where**  $f60:\langle (\bigcup k. \text{sublead-coeff-set } I k) = \text{sublead-coeff-set } I m \wedge m > 0 \rangle$   
**by** (*metis Formal-Power-Series-Ring.included-sublead-gen UNIV-I UN-upper bot-nat-0.extremum dual-order.eq-iff h1 less-Suc0 neq0-conv*)  
**have**  $\langle \forall k \geq m. \text{sublead-coeff-set } I m = \text{sublead-coeff-set } I k \rangle$   
**using** *Formal-Power-Series-Ring.included-sublead-gen f60 h1* **by** *auto*  
**from**  $f2$  **have**  $\langle \exists S. \forall k. \text{finite } (S k) \wedge (\text{sublead-coeff-set } I k) = \text{genideal } R (S k) \rangle$   
**using**  $h2$   $h1$  *sublead-ideal[OF h1]* **unfolding** *noetherian-ring-def noetherian-ring-axioms-def*  
**by** *meson*  
**then obtain**  $S$  **where**  $f4:\langle \forall k. \text{finite } (S k) \wedge (\text{sublead-coeff-set } I k) = \text{genideal } R (S k) \rangle$  **by** *blast*  
**then have**  
 $\langle \exists S. \forall k. \text{finite } (S k) \wedge 0 \notin S k \wedge (\text{sublead-coeff-set } I k) = \text{genideal } R (S k) \wedge$

```

(∀ k ≥ m. S k = S m)
  apply(intro exI[where x = ⟨(λ k. if k ≤ m then S k - {0} else S m - {0})⟩])
  by (smt (verit, ccfv-threshold) Diff-iff Un-Diff-cancel Un-commute ⟨∧ S. Idl
S = Idl (S ∪ {0})⟩
    ⟨∀ k ≥ m. sublead-coeff-set I m = sublead-coeff-set I k⟩ finite-Diff nle-le
singletonI)
  then obtain S' where f5:
    ⟨∀ k. finite (S' k) ∧ 0 ∉ S' k ∧ (sublead-coeff-set I k) = genideal R (S' k) ∧
(∀ k ≥ m. S' k = S' m)⟩
  by blast
  have *: ⟨∀ x ∈ (S' j). ∃ y ∈ I. subdegree y = j ∧ fps-nth y j = x⟩ for j
  proof (safe)
    fix x
    assume h3: ⟨x ∈ S' j⟩
    then have ⟨x ∈ sublead-coeff-set I j⟩
      using f5 unfolding genideal-def by (auto)
    then show ⟨∃ y ∈ I. subdegree y = j ∧ fps-nth y j = x⟩
      unfolding sublead-coeff-set-def subdeg-poly-set using f5
      using h3 by auto
  qed
  define f where ⟨f = (λ j x. (SOME y. y ∈ I ∧ subdegree y = j ∧ fps-nth y j =
x))⟩
  define B where ⟨B = (λ j. {f j x | x. x ∈ S' j})⟩
  have ⟨bij-betw (f k) (S' k) (B k)⟩ for k
    apply(rule bij-betwI[where g = ⟨λ x. fps-nth x k⟩])
    using B-def apply blast
    using f5 B-def image-def f-def Pi-def apply(safe)
      apply (smt (verit, del-insts) * someI-ex)
      apply (smt (verit, del-insts) * f-def someI-ex)
    unfolding f-def B-def image-def
    apply(safe)
    by (smt (verit, ccfv-threshold) * Eps-cong someI-ex)
  then have f6: ⟨card (B j) = card (S' j)⟩ for j
    by (metis bij-betw-same-card)
  have f7: ⟨bij-betw (from-nat-into (B k)) ({0..<card (B k)}) (B k)⟩ for k
    by (simp add: B-def atLeast0LessThan bij-betw-from-nat-into-finite f5)
  have f8: ⟨bij-betw (from-nat-into (S' k)) ({0..<card (S' k)}) (S' k)⟩ for k
    by (simp add: B-def atLeast0LessThan bij-betw-from-nat-into-finite f5)
  have ⟨∀ x ∈ S' k. ∃ y ∈ B k. x = fps-nth y k⟩ for k
    using f5 unfolding B-def f-def sublead-coeff-set-def subdeg-poly-set
    apply(safe)
    by (smt (verit, ccfv-threshold) * mem-Collect-eq someI-ex)
  have f30: ⟨∀ x ∈ B k. ∃ y ∈ S' k. y = fps-nth x k⟩ for k
    using f5 unfolding B-def f-def sublead-coeff-set-def subdeg-poly-set
    apply(safe)
    by (smt (verit, ccfv-threshold) * mem-Collect-eq someI-ex)
  have ⟨∀ i < card (B k). ∃! n. n < card (B k) ∧ fps-nth (from-nat-into (B k) n) k
= from-nat-into (S' k) i⟩
    for k

```

```

proof(safe)
  fix i
  assume  $\langle i < \text{card } (B \ k) \rangle$ 
  then have  $\langle \text{from-nat-into } (S' \ k) \ i \in S' \ k \rangle$ 
    by (metis card.empty f6 from-nat-into less-nat-zero-code)
  then show  $\langle \exists n < \text{card } (B \ k). \text{fps-nth } (\text{from-nat-into } (B \ k) \ n) \ k = \text{from-nat-into } (S' \ k) \ i \rangle$ 
    by (smt (verit, ccfv-threshold)  $\langle \bigwedge k. \forall x \in S' \ k. \exists y \in B \ k. x = \text{fps-nth } y \ k \rangle$ 
      atLeastLessThan-iff bij-betw-iff-bijections f7)
  next
  have  $f9: \langle h \in B \ k \wedge g \in B \ k \implies h \neq g \iff \text{fps-nth } h \ k \neq \text{fps-nth } g \ k \rangle$  for  $k \ h \ g$ 
    using f5 unfolding B-def f-def sublead-coeff-set-def subdeg-poly-set ap-
ply(safe)
    by (smt (verit) * someI-ex)+
    fix i n y
    assume  $\langle i < \text{card } (B \ k) \rangle \langle n < \text{card } (B \ k) \rangle$ 
     $\langle \text{fps-nth } (\text{from-nat-into } (B \ k) \ n) \ k = \text{from-nat-into } (S' \ k) \ i \rangle \langle y < \text{card } (B \ k) \rangle$ 
     $\langle \text{fps-nth } (\text{from-nat-into } (B \ k) \ y) \ k = \text{from-nat-into } (S' \ k) \ i \rangle$ 
    then have  $\langle (\text{from-nat-into } (B \ k) \ n) = (\text{from-nat-into } (B \ k) \ y) \rangle$ 
    using f9
    by (metis card.empty from-nat-into less-nat-zero-code)
    then show  $\langle n = y \rangle$  using f7 unfolding bij-betw-def inj-on-def
    by (metis (no-types, opaque-lifting)  $\langle n < \text{card } (B \ k) \rangle \langle y < \text{card } (B \ k) \rangle$ 
      atLeastLessThan-iff bij-betw-iff-bijections f7 zero-le)
  qed
  then have  $\langle \exists p. (\forall i < \text{card } (S' \ k). \text{fps-nth } (\text{from-nat-into } (B \ k) \ (p \ i)) \ k = \text{from-nat-into } (S' \ k) \ i) \rangle$ 
    for k
    apply(intro exI[where  $x = \langle \lambda i. \text{THE } n. n < \text{card } (B \ k) \wedge \text{fps-nth } (\text{from-nat-into } (B \ k) \ (n)) \ k = \text{from-nat-into } (S' \ k) \ i \rangle$ ])
    apply(safe)
    by (smt (z3) f6 theI')
  then obtain p
    where  $f11: \langle (\forall i < \text{card } (S' \ k). \text{fps-nth } (\text{from-nat-into } (B \ k) \ (p \ k \ i)) \ k = \text{from-nat-into } (S' \ k) \ i) \rangle$ 
    for k
    by metis
    have  $\langle x \neq y \implies x \in S' \ j \wedge y \in S' \ j \implies (\text{SOME } y. y \in I \wedge \text{subdegree } y = j \wedge \text{fps-nth } y \ j = x) \neq (\text{SOME } y'. y' \in I \wedge \text{subdegree } y' = j \wedge \text{fps-nth } y' \ j = y) \rangle$  for  $x \ y \ j$ 
    using *
    apply(safe)
    by (smt (verit, best) someI-ex)
  then have  $\langle \forall x \ y. x \in S' \ j \wedge y \in S' \ j \wedge x \neq y \implies f \ j \ x \neq f \ j \ y \rangle$  for j
    unfolding f-def by(auto)
  have  $f10: \langle \text{finite } (B \ j) \rangle$  for j
    using f6 f5 B-def
    using  $\langle \bigwedge k. \text{bij-betw } (f \ k) \ (S' \ k) \ (B \ k) \rangle$  bij-betw-finite by blast

```

```

from idl-sum
  have  $\langle \forall x \in \text{sublead-coeff-set } I \ m. (\exists s. x = (\sum_{i \in \{0..<\text{card } (S \ m)\}} s \ i \ * \text{from-nat-into } (S \ m) \ i)) \rangle$ 
  using f4 genideal-sum-rep by blast
  have  $\langle \text{genideal } R \ \{\} = \{0\} \rangle$ 
  unfolding genideal-def
  proof(safe)
    fix x
    assume 1:  $\langle x \in \bigcap \{I. \text{ideal } I \ R \wedge \{\} \subseteq I\} \rangle \langle x \notin \{\} \rangle$ 
    have  $\langle \text{ideal } (\{0\}) \ R \rangle$ 
    using R ring.zeroideal ring-R by fastforce
    then have  $\langle x \in \{0\} \rangle$  using 1 by auto
    then show  $\langle x = 0 \rangle$  by auto
  next
    fix X
    assume 2:  $\langle \text{ideal } X \ R \rangle \langle \{\} \subseteq X \rangle$ 
    then show  $\langle 0 \in X \rangle$ 
    using R additive-subgroup.zero-closed ideal.axioms(1) by fastforce
  qed
  have  $\langle \text{sublead-coeff-set } I \ m \neq \{0\} \implies S \ m \neq \{\} \rangle$ 
  using  $\langle \text{Idl } \{\} = \{0\} \rangle$  f4 by force
  define I' where  $\langle I' \equiv \text{genideal FPS-ring } (\bigcup_{k \leq m}. B \ k) \rangle$ 
  have f62:  $\langle (\bigcup_{k \leq m}. B \ k) \subseteq I \wedge \text{finite } (\bigcup_{k \leq m}. B \ k) \rangle$ 
  apply(rule conjI)
  using B-def f-def f10 apply(auto simp:image-def * some-eq-ex)[I]
  apply (smt (verit, del-insts) * some-eq-ex)
  using f10 apply(induct m) by(auto)
  then have  $\langle I' \subseteq I \rangle$ 
  unfolding I'-def
  by (meson Formal-Power-Series-Ring.ring-FPS h1 ring.genideal-minimal)
  have  $\langle \forall k \geq m. S' \ m = S' \ k \rangle$ 
  using f5 by blast
  have eq-fps-S':  $\langle \{ \text{fps-nth } f \ k \mid f \in B \ k \} = S' \ k \rangle$  for k
  unfolding B-def f-def apply(safe)
  using B-def f30 f-def apply blast
  using B-def  $\langle \bigwedge k. \forall x \in S' \ k. \exists y \in B \ k. x = \text{fps-nth } y \ k \rangle$  f-def by blast
  {
    fix f m'
    assume h9:  $\langle f \neq 0 \rangle \langle f \in I \rangle \langle \text{subdegree } f \leq m \rangle \langle f \notin I' \rangle \langle \text{subdegree } f = m' \rangle$ 
    with h9 have  $\langle \text{fps-nth } f \ m' \in \text{sublead-coeff-set } I \ m' \rangle$ 
    unfolding sublead-coeff-set-def subdeg-poly-set
    by blast
    then have  $\langle \exists s. \text{fps-nth } f \ m' = (\sum_{k=0..<\text{card } (S' \ m')}. (s \ k) \ * \text{from-nat-into } (S' \ m') \ k) \rangle$ 
    using f5
    using genideal-sum-rep by blast
    then obtain s where f12:  $\langle \text{fps-nth } f \ m' = (\sum_{k=0..<\text{card } (S' \ m')}. (s \ k) \ * \text{from-nat-into } (S' \ m') \ k) \rangle$ 
    by blast
  }

```



```

then have f21:⟨(∑ k=0.. $\text{card } (S' m')$ . (s k)*from-nat-into (S' m') k)
= fps-nth (∑ k=0.. $\text{card } (B m')$ . (fps-const (s k))*from-nat-into (B m') (p m'
k)) m'⟩
  using f11
  apply(subst fps-sum-nth)
  apply(subst fps-mult-left-const-nth)
  using f6 by fastforce
then have f14:
  ⟨fps-nth f m' = fps-nth (∑ k=0.. $\text{card } (B m')$ . (fps-const (s k))*from-nat-into
(B m') (p m' k)) m'⟩
  using f11 f12 by auto
then have
  ⟨fps-nth ((f - (∑ k=0.. $\text{card } (B m')$ . (fps-const (s k))*from-nat-into (B
m') (p m' k)))) m' = 0⟩
  by auto
have f22:⟨(from-nat-into (B m') (p m' ka)) ∈ B m'⟩ for ka
  by (metis atLeastLessThan0 card.empty f12 f6 from-nat-into h9(1)
h9(5) nth-subdegree-zero-iff sum.empty)
then have f13:⟨∀ k<m'. fps-nth (from-nat-into (B m') (p m' ka)) k = 0⟩
for ka
  unfolding B-def f-def using f5 unfolding sublead-coeff-set-def sub-
deg-poly-set
  by (smt (verit, best) * h9 mem-Collect-eq nth-less-subdegree-zero someI-ex)
then have
  ⟨∀ ka<m'. fps-nth (∑ k=0.. $\text{card } (B m')$ . (fps-const (s k))*from-nat-into (B
m') (p m' k)) ka = 0⟩
  apply(subst fps-sum-nth)
  apply(subst fps-mult-left-const-nth) apply(safe)
  apply(subst f13) by(auto)
then have f18:⟨ka≤m' ⇒ (fps-nth (f - (∑ k=0.. $\text{card } (B m')$ .
(fps-const (s k))*from-nat-into (B m') (p m' k))) ka = 0⟩ for ka
  apply(cases ⟨ka=m'⟩)
  using f14 apply fastforce
  using nth-less-subdegree-zero
  using h9(5) by force
have ⟨from-nat-into (B m') (p m' k) ∈ I'⟩ for k
  using f22 unfolding I'-def genideal-def
  using h9(3) h9(5) by blast
then have f23:⟨(fps-const (s k))*from-nat-into (B m') (p m' k) ∈ I'⟩ for k
by (metis FPS-ring-def I'-def UNIV-I ideal.I-l-closed monoid.select-convs(1)

partial-object.select-convs(1) ring.genideal-ideal ring-FPS subset-UNIV)
have f24:⟨(∑ k=0.. $\text{card } (B m')$ . (fps-const (s k))*from-nat-into (B m') (p m' k)) ∈
I'⟩ for r using f22
  apply(induct r)
  apply (metis (full-types) Formal-Power-Series-Ring.FPS-ring-def I'-def
additive-subgroup.zero-closed atLeastLessThan0 ideal-def partial-object.select-convs(1)

ring.genideal-ideal ring.simps(1) ring-FPS sum.empty top-greatest)

```

```

apply(subst sum.atLeast0-lessThan-Suc)
unfolding I'-def
by (metis (no-types, lifting) Formal-Power-Series-Ring.FPS-ring-def I'-def
    additive-subgroup.a-closed f23 ideal.axioms(1) partial-object.select-convs(1)

    ring.genideal-ideal ring-FPS ring-record-simps(12) subset-UNIV)
then have f26:
  ⟨f - (∑ k=0.. $\text{card } (B \ m')$ . (fps-const (s k))*from-nat-into (B m') (p m'
k))) = 0 ⟹ False⟩
  using h9 by auto
  have ⟨subdegree (f - (∑ k=0.. $\text{card } (B \ m')$ . (fps-const (s k))*from-nat-into
(B m') (p m' k))) > m'⟩
  using f26
  by (smt (verit, ccfv-SIG)f18 enat-ord-code(4) enat-ord-simps(1)
    linorder-not-less nth-subdegree-zero-iff)
then have ⟨∃ g ∈ I'. subdegree (f + g) > m' ∧ (f + g) ≠ 0⟩
  using f26
  apply(intro be_xI[where x = ⟨- (∑ k=0.. $\text{card } (B \ m')$ . (fps-const (s k))*
from-nat-into (B m') (p m' k))⟩)]
  using f26 f24 apply(safe)
  apply(auto)[2]
  by (metis (no-types, lifting) Formal-Power-Series-Ring.FPS-ring-def
    Formal-Power-Series-Ring.ring-FPS I'-def UNIV-I ideal.I-l-closed
monoid.select-convs(1)
    mult-1s(3) partial-object.select-convs(1) ring.genideal-ideal subset-UNIV)
} note first = this
{
fix f
assume h10:⟨f ≠ 0⟩ ⟨f ∈ I⟩ ⟨f ∉ I'⟩ ⟨subdegree f < m⟩
have ⟨∃ g ∈ I'. subdegree (f + g) > subdegree f ∧ f + g ≠ 0 ⟩
  using first[OF h10(1) h10(2) - h10(3)]
  using h10(4) nat-less-le
  by blast
have ⟨∃ g ∈ I'. subdegree (f + g) ≥ m' ∧ f + g ≠ 0⟩ if hh:⟨m' ≤ m⟩ for m'
  using hh h10 proof(induct m' arbitrary:f)
  case 0
  then show ?case
    using first less-or-eq-imp-le by blast
next
case (Suc m')
then obtain g where g1:⟨g ∈ I' ∧ subdegree (f + g) ≥ m' ∧ f + g ≠ 0⟩
  using first order-less-imp-le
  by (metis less-Suc-eq-le nle-le)
{assume hh1:subdegree (f + g) < Suc m'
  with g1 have g2:⟨subdegree (f + g) = m'⟩
  by auto
  have g3:⟨f + g ∈ I⟩
  by (metis Formal-Power-Series-Ring.FPS-ring-def Suc.premis(3)
    ⟨I' ⊆ I⟩ additive-subgroup.a-closed g1 h1 ideal.axioms(1)

```

```

      in-mono ring-record-simps(12))
  have g4: ⟨f+g ∉ I'⟩
  proof(rule ccontr)
    assume ⟨¬f+g ∉ I'⟩
    have ⟨-g ∈ I'⟩
      using g1 unfolding I'-def
    by (metis (no-types, lifting) FPS-ring-def Formal-Power-Series-Ring.genideal-sum-rep
      Formal-Power-Series-Ring.idl-sum UNIV-I f62 ideal.I-l-closed
      monoid.select-convs(1)
      mult-1s(3) partial-object.select-convs(1))
    then have ⟨f+g-g ∈ I'⟩
      by (metis (no-types, lifting) Formal-Power-Series-Ring.FPS-ring-def
      Formal-Power-Series-Ring.genideal-sum-rep Formal-Power-Series-Ring.idl-sum
      I'-def
      Suc.prem(4) f62 ⟨¬f+g ∉ I'⟩ add.commute additive-subgroup.a-closed
      ideal.axioms(1)
      minus-add-cancel ring-record-simps(12))
    then have ⟨f ∈ I'⟩ by auto
    then show False
      using Suc.prem(4) by auto
  qed
  have g5: ⟨subdegree (f+g) ≤ m⟩
    by (simp add: Suc.prem(1) Suc-leD g2)
  then obtain g' where ⟨g' ∈ I' ∧ subdegree (f + g + g') > subdegree (f+g)
  ∧ f+g+g' ≠ 0⟩
    using first[OF - g3 g5 g4, of m']
    using g1 g2 by blast
  then have ⟨subdegree (f+g+g') ≥ Suc m'⟩
    using Suc-le-eq g2 by blast
  then have ⟨∃ g' ∈ I'. subdegree (f + g + g') ≥ Suc m' ∧ f+g+g' ≠ 0⟩
    using ⟨g' ∈ I' ∧ subdegree (f + g) < subdegree (f + g + g') ∧ f + g +
  g' ≠ 0⟩ by blast
  }note proof1=this
  then obtain g' where ttt: ⟨subdegree (f+g) < Suc m' ⟹ g' ∈ I' ∧
  subdegree (f + g) < subdegree (f + g + g') ∧ f + g + g' ≠ 0⟩
    using order-less-le-trans by blast
  show ?case apply(cases ⟨subdegree (f+g) ≥ Suc m'⟩)
    using g1 apply blast
    using proof1
    apply(intro beI[where x=⟨g + g'⟩])
    apply (metis Suc-leI ab-semigroup-add-class.add-ac(1)
      g1 le-less-Suc-eq linorder-not-less ttt)
    unfolding I'-def
  by (metis (no-types, lifting) FPS-ring-def Formal-Power-Series-Ring.genideal-sum-rep
      Formal-Power-Series-Ring.idl-sum I'-def ⟨⋃ (B ' {..m}) ⊆ I ∧ finite
      (⋃ (B ' {..m}))⟩
      additive-subgroup.a-closed g1 ideal.axioms(1) linorder-not-less ring-record-simps(12)
      ttt)

```

```

qed
} note snd=this
{
  fix f m'
  assume h9:⟨f ≠ 0⟩ ⟨f ∈ I⟩ ⟨subdegree f ≥ m⟩ ⟨subdegree f = m'⟩ ⟨f ∉ I'⟩
  then have ⟨fps-nth f m' ∈ sublead-coeff-set I m'⟩
    unfolding sublead-coeff-set-def subdeg-poly-set by auto
  then have f28:⟨fps-nth f m' ∈ sublead-coeff-set I m'⟩
    ⟨sublead-coeff-set I m' = genideal R (S' m)⟩
  using ⟨∀ k ≥ m. sublead-coeff-set I m = sublead-coeff-set I k⟩
    h9 less-or-eq-imp-le apply blast
  using f5 by (metis h9(3-4))
  then have ⟨∃ s. fps-nth f m' = (∑ k=0..<card (S' m). (s k)*from-nat-into
(S' m) k)⟩
    using genideal-sum-rep f5 by blast
  then obtain s where f12:⟨fps-nth f m' = (∑ k=0..<card (S' m). (s
k)*from-nat-into (S' m) k)⟩
    by blast
  then have f21:⟨(∑ k=0..<card (S' m). (s k)*from-nat-into (S' m) k)
=fps-nth (∑ k=0..<card (B m). (fps-const (s k))*from-nat-into (B m) (p m k))
m⟩
    using f11
    apply (subst fps-sum-nth)
    apply (subst fps-mult-left-const-nth)
    using f6 by fastforce
  then have f14:
    ⟨fps-nth f m' = fps-nth (∑ k=0..<card (B m). (fps-const (s k))*from-nat-into
(B m) (p m k)) m⟩
    using f11 f12 by auto
  have f22:⟨(from-nat-into (B m) (p m ka)) ∈ B m⟩ for ka
    by (metis atLeastLessThan0 card.empty f12 f6 from-nat-into h9(1) h9(4)
nth-subdegree-zero-iff sum.empty)
  then have ⟨subdegree (from-nat-into (B m) (p m k)) = m⟩ for k
    unfolding B-def f-def sublead-coeff-set-def subdeg-poly-set
    by (smt (verit, best) * h9 mem-Collect-eq nth-less-subdegree-zero someI-ex)
  then have f32:⟨subdegree ((fps-X^(m'-m))*from-nat-into (B m) (p m k)) =
m'⟩ for k
    using fps-subdegree-mult-fps-X-power(1)
    by (metis f30 f22 f5 h9(3) h9(4) le-add-diff-inverse nth-subdegree-zero-iff)
  have f31:
    ⟨fps-nth ((fps-X^(m'-m))*from-nat-into (B m) (p m k)) m' = fps-nth
(from-nat-into (B m) (p m k)) m⟩
    for k
    by (metis diff-diff-cancel diff-le-self fps-X-power-mult-nth h9(3) h9(4)
linorder-not-less)
  then have f31b:⟨fps-nth (fps-const (s k)*(fps-X^(m'-m))*from-nat-into (B
m) (p m k)) m' =
fps-nth (fps-const (s k)*from-nat-into (B m) (p m k)) m⟩ for k
    by (simp add: mult.assoc)

```

```

then have f33:⟨fps-nth f m' = fps-nth (∑ k=0.. $\text{card } (B m)$ ). (fps-const (s
k))*fps-X $\wedge$ (m'-m))*
from-nat-into (B m) (p m k) m'⟩
apply(subst fps-sum-nth)
apply(subst f31b)
apply(subst fps-mult-left-const-nth)
by (simp add: f14 fps-sum-nth)
then have f36:⟨fps-nth ((f - (∑ k=0.. $\text{card } (B m)$ ). (fps-const (s k))*fps-X $\wedge$ (m'-m))*
from-nat-into (B m) (p m k))) m' = 0⟩
by auto
have ⟨from-nat-into (B m) (p m k) ∈ I'⟩ for k
using f22 unfolding I'-def genideal-def
using h9 by blast
then have f23:⟨fps-const (s k)*from-nat-into (B m) (p m k) ∈ I'⟩ for k
by (metis FPS-ring-def I'-def UNIV-I ideal.I-l-closed monoid.select-convs(1)

partial-object.select-convs(1) ring.genideal-ideal ring-FPS subset-UNIV)
then have f23:⟨fps-const (s k)*fps-X $\wedge$ (m'-m)*from-nat-into (B m) (p m
k) ∈ I'⟩ for k
by (metis (no-types, lifting) FPS-ring-def Formal-Power-Series-Ring.genideal-sum-rep

Formal-Power-Series-Ring.idl-sum I'-def UNIV-I ⟨ $\bigwedge$ k. from-nat-into (B
m) (p m k) ∈ I'⟩
f62 ideal.I-l-closed monoid.select-convs(1) partial-object.select-convs(1))
have f24:⟨(∑ k=0.. $\text{card } (B m)$ ). (fps-const (s k))*fps-X $\wedge$ (m'-m)*from-nat-into (B m)
(p m k) ∈ I'⟩
for r
using f22
apply(induct r)
apply(simp)
apply (metis (full-types) Formal-Power-Series-Ring.FPS-ring-def I'-def
additive-subgroup.zero-closed
ideal-def partial-object.select-convs(1) ring.genideal-ideal ring.simps(1)
ring-FPS top-greatest)
apply(subst sum.atLeast0-lessThan-Suc)
unfolding I'-def
by (metis (no-types, lifting) Formal-Power-Series-Ring.FPS-ring-def I'-def
additive-subgroup.a-closed f23 ideal.axioms(1) partial-object.select-convs(1)

ring.genideal-ideal ring-FPS ring-record-simps(12) subset-UNIV)
then have f26:
⟨f - (∑ k=0.. $\text{card } (B m)$ ). (fps-const (s k))*fps-X $\wedge$ (m'-m)*from-nat-into
(B m) (p m k)) = 0 ⟹ False⟩
(is ⟨f - ?A = 0 ⟹ False⟩ using h9 by auto
have ⟨ $\forall i < m'$ . fps-nth ((fps-const (s k))*fps-X $\wedge$ (m'-m))*from-nat-into (B
m) (p m k)) i = 0⟩
for k
using f32
by (metis ab-semigroup-mult-class.mult-ac(1) fps-mult-nth-outside-subdegrees(2))

```

```

then have f34:⟨∀ i < m'. fps-nth (?A) i = 0⟩
  apply(subst fps-sum-nth)
  by(auto)
then have f35:⟨subdegree ?A = m'⟩
  by (metis (no-types, lifting) f33 h9(1) h9(4) nth-subdegree-nonzero subde-
greeI)
have ⟨x ∈ I' ⇒ -x ∈ I'⟩ for x
  unfolding I'-def FPS-ring-def
  by (metis (no-types, lifting) Formal-Power-Series-Ring.genideal-sum-rep
Formal-Power-Series-Ring.idl-sum UNIV-I f62
ideal.I-l-closed monoid.select-convs(1) mult-1s(3) partial-object.select-convs(1))

have f39:⟨subdegree (- ?A) ≥ m⟩
  using subdegree-uminus[of ?A] f35 h9 by argo
have ⟨subdegree (f - (∑ k=0..<card (B m).
(fps-const (s k))* (fps-X^(m'-m))*from-nat-into (B m) (p m k))) > m'⟩
  using h9 f33
  by (metis (no-types, lifting) f36 f35 diff-zero f26 fps-sub-nth le-neq-implies-less

nth-subdegree-nonzero subdegree-leI)
then have ⟨∃ g. ∃ s. g = - (∑ k=0..<card (B m). (fps-const (s k))* (fps-X^(m'-m))*
from-nat-into (B m) (p m k)) ∧ subdegree (f + g) > m' ∧ (f + g) ≠ 0 ∧ g ∈ I' ∧
subdegree (g) ≥ m⟩
  using f26 f24 ⟨∧ x. x ∈ I' ⇒ -x ∈ I'⟩ f39
  by (metis (no-types, lifting) add-uminus-conv-diff)
note thrd=this
have in-I':⟨x ∈ I' ⇒ -x ∈ I'⟩ for x
  unfolding I'-def FPS-ring-def
  by (metis (no-types, lifting) Formal-Power-Series-Ring.genideal-sum-rep
Formal-Power-Series-Ring.idl-sum UNIV-I ⟨∪ (B ' {..m}) ⊆ I ∧ finite
(∪ (B ' {..m}))⟩
ideal.I-l-closed monoid.select-convs(1) mult-1s(3) partial-object.select-convs(1))

have ⟨I ⊆ I'⟩
proof(safe, rule ccontr)
  fix f
  assume h10:⟨f ∈ I⟩ ⟨f ∉ I'⟩
  then have f40:⟨f ≠ 0⟩
    by (metis FPS-ring-def I'-def additive-subgroup.zero-closed ideal.axioms(1)
partial-object.select-convs(1) ring.genideal-ideal ring.simps(1) ring-FPS
subset-UNIV)
  have ⟨∃ g ∈ I'. subdegree (f + g) ≥ m ∧ f + g ≠ 0⟩
    using snd[OF f40 h10 ]
  by (metis Formal-Power-Series-Ring.FPS-ring-def Formal-Power-Series-Ring.ring-FPS
I'-def
add.right-neutral additive-subgroup.zero-closed f40 ideal.axioms(1)
linorder-not-less
order-refl partial-object.select-convs(1) ring.genideal-ideal ring.simps(1)
subset-UNIV)

```

```

then obtain  $g$  where  $f41:\langle g \in I' \wedge \text{subdegree}(f + g) \geq m \wedge f + g \neq 0 \rangle$  by blast

then have  $\text{hyps}:\langle f + g \neq 0 \rangle \langle f + g \in I \rangle \langle \text{subdegree}(f + g) \geq m \rangle \langle (f + g) \notin I' \rangle$ 
proof –
  show  $\langle f + g \neq 0 \rangle \langle m \leq \text{subdegree}(f + g) \rangle$  using  $f41$  by auto
  have  $\langle g \in I \rangle$ 
    using  $\langle I' \subseteq I \rangle$   $f41$  by blast
  then show  $\langle f + g \in I \rangle$ 
    by (metis FPS-ring-def additive-subgroup.a-closed h1 h10(1) ideal-def
ring-record-simps(12))
  show  $\langle f + g \notin I' \rangle$ 
  proof(rule ccontr)
    assume  $\langle \neg f + g \notin I' \rangle$ 
    have  $\langle -g \in I' \rangle$ 
      unfolding  $I'$ -def FPS-ring-def
    by (metis Formal-Power-Series-Ring.FPS-ring-def Formal-Power-Series-Ring.ring-R
I'-def
       $f41$  ideal.I-l-closed iso-tuple-UNIV-I monoid.select-convs(1) mult-minus1
      partial-object.select-convs(1) ring.genideal-ideal subset-UNIV)
    then have  $\langle f + g - g \in I' \rangle$ 
      by (metis (no-types, lifting) Formal-Power-Series-Ring.FPS-ring-def
Formal-Power-Series-Ring.genideal-sum-rep Formal-Power-Series-Ring.idl-sum
I'-def
       $f62$   $\langle \neg f + g \notin I' \rangle$  add-stable-FPS-ring uminus-add-conv-diff)
    then have  $\langle f \in I' \rangle$  by auto
    then show False
      using  $h10(2)$  by blast
  qed
qed
define  $\text{the-s}$  where  $\langle \text{the-s} \equiv \text{rec-nat}(f + g)$ 
 $(\lambda n \text{ sn}. \text{sn} + (\text{SOME } g. \exists s. g = -(\sum_{k=0..<\text{card}(B \text{ m}). (\text{fps-const}(s \text{ k})) * (\text{fps-X}^\wedge(\text{subdegree}$ 
 $\text{sn} - m)))$ 
 $* \text{from-nat-into}(B \text{ m})(p \text{ m } k))$ 
 $\wedge \text{subdegree}(\text{sn} + g) > \text{subdegree } \text{sn} \wedge (\text{sn} + g) \neq 0 \wedge g \in I' \wedge \text{subdegree } g \geq m) \rangle$ 
have  $\text{subst-rec}:\langle \text{the-s}(\text{Suc } n) = \text{the-s } n + (\text{SOME } g. \exists s. g = -(\sum_{k=0..<\text{card}$ 
 $(B \text{ m}).$ 
 $(\text{fps-const}(s \text{ k})) * (\text{fps-X}^\wedge(\text{subdegree}(\text{the-s}(n)) - m)) * \text{from-nat-into}(B \text{ m})(p \text{ m}$ 
 $k))$ 
 $\wedge \text{subdegree}((\text{the-s}(n)) + g) > \text{subdegree}(\text{the-s}(n)) \wedge ((\text{the-s}(n)) + g) \neq 0$ 
 $\wedge g \in I' \wedge \text{subdegree } g \geq m) \rangle$  for  $n$ 
  unfolding  $\text{the-s-def}$ 
  apply(induct n)
  by (meson old.nat.simps(7)) +
  have  $\text{hyps-the-s}:\langle \text{the-s } n \neq 0 \wedge \text{the-s } n \in I \wedge \text{subdegree}(\text{the-s } n) \geq m \wedge (\text{the-s } n) \notin I' \rangle$ 
for  $n$ 
  proof(induct n)
    case  $0$ 
    then show ?case unfolding  $\text{the-s-def}$  using  $\text{hyps}$  by auto

```

```

next
  case (Suc n)
  then have y1:⟨the-s n ≠ 0⟩ and y2: ⟨the-s n ∈ I⟩ and y3:⟨m ≤ subdegree
(the-s n)⟩
    and y4:⟨the-s n ∉ I'⟩
    by auto
  have f50:⟨∃ g. ∃ s. g = - (∑ k = 0.. $\text{card } (B m)$ . fps-const (s k) * fps-X
^
(subdegree (the-s n) - m) * from-nat-into (B m) (p m k)) ∧ subdegree (the-s n)
<
subdegree (the-s n + g) ∧ the-s n + g ≠ 0 ∧ g ∈ I' ∧ subdegree g ≥ m⟩
  using thrd[OF y1 y2 y3 - y4, of ⟨subdegree (the-s n)⟩] by auto
  let ?g = ⟨(SOME g. ∃ s. g = - (∑ k = 0.. $\text{card } (B m)$ . fps-const (s k) *
fps-X ^
(subdegree (the-s n) - m) * from-nat-into (B m) (p m k)) ∧ subdegree (the-s n)
<
subdegree (the-s n + g) ∧ the-s n + g ≠ 0 ∧ g ∈ I' ∧ subdegree g ≥ m)⟩
  have ⟨the-s (Suc n) ∉ I'⟩
  proof(subst subst-rec, rule ccontr)
    assume h100: ⟨¬the-s n + ?g ∉ I'⟩
    have ⟨?g ∈ I'⟩
      by(smt someI-ex f50)
    then have ⟨-?g ∈ I'⟩
      using in-I' by auto
    then have ⟨the-s n + ?g - ?g ∈ I'⟩
      by (metis (no-types, lifting) Formal-Power-Series-Ring.FPS-ring-def
Formal-Power-Series-Ring.genideal-sum-rep Formal-Power-Series-Ring.idl-sum
I'-def
f62 h100 add-stable-FPS-ring add-uminus-conv-diff)
    then have ⟨the-s n ∈ I'⟩ by auto
    then show False
      using y4 by blast
  qed
  have ⟨the-s (Suc n) ∈ I⟩
  proof(subst subst-rec)
    have ⟨?g ∈ I'⟩
      by(smt someI-ex f50)
    then show ⟨the-s n + ?g ∈ I⟩
      using ⟨I' ⊆ I⟩ add-stable-FPS-ring h1 y2 by blast
  qed
  have f51:⟨the-s (Suc n) ≠ 0⟩
  apply(subst subst-rec)
  by (smt someI-ex f50)
  have ⟨m ≤ subdegree (the-s (Suc n))⟩
  proof(subst subst-rec)
    have ⟨subdegree ?g ≥ m⟩
      by (smt someI-ex f50)
    then show ⟨m ≤ subdegree (the-s n + ?g)⟩
      using f51 apply(subst (asm) subst-rec)

```



by (smt (verit) add-diff-cancel-left' dual-order.trans f50 linorder-le-less-linear  
 subdegree-diff-eq1 subdegree-diff-eq2 y1)

**qed**  
**then show** ?case  
 using ⟨the-s (Suc n) ∈ I⟩ ⟨the-s (Suc n) ∉ I'⟩ f51 **by** blast

**qed**  
**have** f53:⟨∀ n. ∃ g. ∃ s. g = - (∑ k=0.. $\text{card } (B m)$ . (fps-const (s k))\*  
 (fps-X<sup>∧</sup>(subdegree (the-s (n)) - m))\*from-nat-into (B m) (p m k))  
 ∧ subdegree ((the-s (n)) + g) > subdegree (the-s (n))  
 ∧ ((the-s (n)) + g) ≠ 0 ∧ g ∈ I' ∧ subdegree g ≥ m⟩  
 using thrd hyps-thes **by** blast  
**then have** f53':⟨∀ n. ∃ g. ∃ s. the-s (Suc n) = the-s n + g ∧ g = -  
 (∑ k=0.. $\text{card } (B m)$ .  
 (fps-const (s k))\* (fps-X<sup>∧</sup>(subdegree (the-s (n)) - m))\*from-nat-into (B m) (p m  
 k)) ∧  
 subdegree ((the-s (n)) + g) > subdegree (the-s (n)) ∧ ((the-s (n)) + g) ≠ 0 ∧  
 g ∈ I' ∧ subdegree g ≥ m⟩  
**apply**(subst subst-rec)  
**by** (smt (z3) tfl-some)

**from** f53 **have** ⟨subdegree (the-s n) < subdegree (the-s (Suc n))⟩ **for** n  
**apply**(subst subst-rec)  
**by** (smt someI-ex f53 sum.cong)

**then have** f56:⟨strict-mono (λn. subdegree (the-s n))⟩  
**using** strict-mono-Suc-iff **by** blast

**have** f70:⟨strict-mono (λk. subdegree(the-s k) - m)⟩  
**using** f56 **unfolding** strict-mono-def  
**using** diff-less-mono hyps-thes **by** presburger

**let** ?f = ⟨λk. subdegree (the-s k) - m⟩  
**have** ⟨bij-betw ?f UNIV (range ?f)⟩  
**by** (simp add: ⟨strict-mono ?f⟩ bij-betw-imageI strict-mono-on-imp-inj-on)

**from** f56 **have** f80:⟨the-s ⟶ 0⟩  
**using** subdeg-inf-imp-s-tendsto-zero **by** blast

**have** f54:⟨∃ g' s'. ∀ n. g' n = - (∑ k=0.. $\text{card } (B m)$ . (fps-const (s' n k))\*  
 (fps-X<sup>∧</sup>(subdegree (the-s n) - m))\*from-nat-into (B m) (p m k))  
 ∧ subdegree ((the-s n) + (g' n)) > subdegree (the-s n) ∧ ((the-s n) + g' n) ≠ 0 ∧  
 g' n ∈ I'  
 ∧ subdegree (g' n) ≥ m⟩  
**using** f53 **by** meson

**have** ⟨∃ g' s'. ∀ n. the-s (Suc n) = the-s n + g' n ∧ g' n = - (∑ k=0.. $\text{card } (B m)$ .  
 (fps-const (s' n k))\* (fps-X<sup>∧</sup>(subdegree (the-s n) - m))\*from-nat-into (B m) (p m  
 k))  
 ∧ subdegree ((the-s n) + (g' n)) > subdegree (the-s n) ∧ ((the-s n) + g' n) ≠ 0 ∧  
 g' n ∈ I'  
 ∧ subdegree (g' n) ≥ m⟩  
**using** f53' **by** meson

**then obtain** g' s' **where** f55:⟨∀ n. the-s (Suc n) = the-s n + g' n ∧ g' n = -  
 (∑ k=0.. $\text{card } (B m)$ .  
 (fps-const (s' n k))\* (fps-X<sup>∧</sup>(subdegree (the-s n) - m))\*from-nat-into (B m) (p m

$k$ )  
 $\wedge \text{subdegree } ((\text{the-}s \ n) + (g' \ n)) > \text{subdegree } (\text{the-}s \ n) \wedge ((\text{the-}s \ n) + g' \ n) \neq 0 \wedge$   
 $g' \ n \in I'$   
 $\wedge \text{subdegree } (g' \ n) \geq m$   
**by** *blast*  
**then have**  $\langle \forall n. \exists s. \forall k. s' \ n \ k = s \ (\text{subdegree } (\text{the-}s \ n) - m) \ k \rangle$   
**by** *force*  
**have**  $\langle \text{the-}s \ n = f + g + (\sum k < n. (\text{the-}s \ (\text{Suc } k) - \text{the-}s \ k)) \rangle$  **for**  $n$   
**apply** (*induct*  $n$ )  
**apply** (*subst subst-rec* [*rule-format*])  
**apply** (*simp add: the-s-def*)  
**by** *simp*  
**then have**  $t1: \langle f + g = \text{the-}s \ n - (\sum k < n. (\text{the-}s \ (\text{Suc } k) - \text{the-}s \ k)) \rangle$  **for**  $n$   
**by** (*metis* (*no-types, lifting*) *add-diff-cancel-right*)  
**then have**  $\langle f + g = \text{the-}s \ n - (\sum k < n. g' \ k) \rangle$  **for**  $n$   
**by** (*simp add: f55*)  
**then have**  $\langle f + g = \text{the-}s \ n - (\sum k < n. -(\sum i = 0..<\text{card } (B \ m). (\text{fps-const } (s' \ k \ i)))$   
 $(\text{fps-X} \wedge (\text{subdegree } (\text{the-}s \ k) - m)) * \text{from-nat-into } (B \ m) (p \ m \ i))) \rangle$  **for**  $n$   
**by** (*simp add: f55*)  
**then have**  $f87: \langle f + g = \text{the-}s \ n + (\sum k < n. (\sum i = 0..<\text{card } (B \ m). (\text{fps-const } (s' \ k \ i)))$   
 $(\text{fps-X} \wedge (\text{subdegree } (\text{the-}s \ k) - m)) * \text{from-nat-into } (B \ m) (p \ m \ i))) \rangle$   
**for**  $n$   
**by** (*simp add: sum-negf*)  
**then have**  $\langle f + g = \text{the-}s \ n + ((\sum i = 0..<\text{card } (B \ m). \sum k < n. (\text{fps-const } (s' \ k \ i)))$   
 $(\text{fps-X} \wedge (\text{subdegree } (\text{the-}s \ k) - m)) * \text{from-nat-into } (B \ m) (p \ m \ i))) \rangle$  **for**  $n$   
**proof** –  
**assume**  $\bigwedge n. f + g = \text{the-}s \ n + (\sum k < n. \sum i = 0..<\text{card } (B \ m). \text{fps-const } (s' \ k \ i) * \text{fps-X} \wedge (\text{subdegree } (\text{the-}s \ k) - m) * \text{from-nat-into } (B \ m) (p \ m \ i))$   
**then have**  $f + g = \text{the-}s \ n + (\sum n = 0..<n. \sum na = 0..<\text{card } (B \ m). \text{fps-const } (s' \ n \ na) * \text{fps-X} \wedge (\text{subdegree } (\text{the-}s \ n) - m) * \text{from-nat-into } (B \ m) (p \ m \ na))$   
**using** *atLeast0LessThan* **by** *moura*  
**then show** *?thesis*  
**using** *atLeast0LessThan sum.swap* **by** *force*  
**qed**  
**then have**  $f57: \langle f + g = \text{the-}s \ n + ((\sum i = 0..<\text{card } (B \ m). (\sum k < n. (\text{fps-const } (s' \ k \ i)))$   
 $(\text{fps-X} \wedge (\text{subdegree } (\text{the-}s \ k) - m))) * \text{from-nat-into } (B \ m) (p \ m \ i))) \rangle$   
**for**  $n$   
**by** (*auto simp: sum-distrib-right*)  
**have**  $\langle (\lambda n. (f + g)) - \text{the-}s = (\lambda n. (f + g) + (-\text{the-}s \ n)) \rangle$   
**by** (*auto simp: fun-eq-iff*)  
**have**  $\langle - \text{the-}s \longrightarrow 0 \rangle$   
**apply** (*rule metric-LIMSEQ-I*)  
**using** *f80*  
**apply** (*drule metric-LIMSEQ-D*)

```

unfolding dist-fps-def
by fastforce
then have f58: $\langle (\lambda n. (f+g)) - \text{the-}s \longrightarrow f + g \rangle$ 
proof –
  have  $\forall n. f + g + (- \text{the-}s) n = ((\lambda n. f + g) - \text{the-}s) n$ 
  by auto
  then show ?thesis
    by (metis (no-types)  $\langle - \text{the-}s \longrightarrow 0 \rangle$  LIMSEQ-add-fps[of  $\langle (\lambda n. f + g) \rangle$   $\langle f+g \rangle$   $\langle -\text{the-}s \rangle$  0]
      add.right-neutral lim-sequentially tendsto-const)
  qed
  then have  $\langle f+g = \lim (\lambda n. \text{the-}s n + ((\sum i=0..<\text{card } (B m). (\sum k<n. (\text{fps-const } (s' k i))*(\text{fps-X}^\wedge(\text{subdegree } (\text{the-}s k) - m)))*\text{from-nat-into } (B m) (p m i)))) \rangle$ 
  using f57 by auto
  have  $\langle (\lambda n. f+g) - \text{the-}s = (\lambda n. (\sum i=0..<\text{card } (B m). (\sum k<n. (\text{fps-const } (s' k i))*(\text{fps-X}^\wedge(\text{subdegree } (\text{the-}s k) - m)))*\text{from-nat-into } (B m) (p m i)))) \rangle$ 
  using f57 apply (subst fun-eq-iff, safe)
  by (smt (verit, best) add-diff-cancel-left' minus-apply)
  then have  $\langle (\lambda n. (\sum i=0..<\text{card } (B m). (\sum k<n. (\text{fps-const } (s' k i))*(\text{fps-X}^\wedge(\text{subdegree } (\text{the-}s k) - m)))*\text{from-nat-into } (B m) (p m i)))) \longrightarrow f+g \rangle$ 
  (is  $\langle ?S \longrightarrow f+g \rangle$  using f58 by presburger)
  then have f84: $\langle f+g = \lim ?S \rangle$ 
  by (simp add: limI)
  have f63:  $\langle \text{finite } (\bigcup (B \text{ ' } \{..m\})) \rangle$ 
  using f62 by fastforce
  have  $\langle \text{strict-mono } (\lambda k. \text{subdegree } ((\sum i=0..<\text{card } (B m). (\text{fps-const } (s' k i))*(\text{fps-X}^\wedge(\text{subdegree } (\text{the-}s k) - m)))*\text{from-nat-into } (B m) (p m i)))) \rangle$ 
  apply (rule monotone-onI)
  apply (insert f55 f56 hyps-thes)
  by (smt (verit, ccfv-threshold) f87 add.commute add-left-cancel diff-add-cancel
    strict-monoD subdeg-inf-imp-s-tendsto-zero subdegree-diff-eq2 subde-
gree-uminus sum-negf)
  then have  $\langle (\lambda k. (\sum i=0..<\text{card } (B m). (\text{fps-const } (s' k i))*(\text{fps-X}^\wedge(\text{subdegree } (\text{the-}s k) - m)))*\text{from-nat-into } (B m) (p m i)) \longrightarrow 0 \rangle$ 
  using subdeg-inf-imp-s-tendsto-zero by presburger
  define fct where  $\langle \text{fct} = ?f \rangle$ 
  then have f71: $\langle \text{strict-mono } \text{fct} \rangle$  using f70 by auto
  have  $\langle \forall k. \exists s. \forall n. (\sum i<n. (\text{fps-const } (s' i k))*(\text{fps-X}^\wedge(\text{fct } i))) = (\sum i<\text{fct } n. (\text{fps-const } (s i))*(\text{fps-X}^\wedge(i))) \rangle$ 
  using exists-seq-all'[OF f71, of  $\langle \lambda i. s' i k \rangle$  for k]
  by meson
  then obtain s where f72: $\langle \forall n k. (\sum i<n. (\text{fps-const } (s' i k))*(\text{fps-X}^\wedge(\text{fct } i))) =$ 
 $(\sum i<\text{fct } n. (\text{fps-const } (s i k))*(\text{fps-X}^\wedge(i))) \rangle$ 
  by meson

```

```

      then have f85:  $\langle (\lambda n. \sum i < \text{fct } n. (\text{fps-const } (s \ i \ k)) * (\text{fps-X}^\wedge(i))) \longrightarrow$ 
Abs-fps  $\langle (\lambda i. s \ i \ k) \rangle$  for k
      by (simp add: Formal-Power-Series-Ring.tendsto-f-seq f71)
      then have f86:  $\langle (\lambda n. (\sum i < n. (\text{fps-const } (s' \ i \ k)) * (\text{fps-X}^\wedge(\text{subdegree } (\text{the-s } i) - m))))$ 
=  $\langle (\lambda n. \sum i < \text{fct } n. (\text{fps-const } (s \ i \ k)) * (\text{fps-X}^\wedge(i))) \rangle$ 
      for k
      using f72 fct-def by (auto simp: fun-eq-iff)
      then have  $\langle (\lambda n. (\sum k < n. (\text{fps-const } (s' \ k \ i)) * (\text{fps-X}^\wedge(\text{subdegree } (\text{the-s } k) - m))))$ 
 $\longrightarrow$  Abs-fps  $\langle (\lambda k. s \ k \ i) \rangle$  for i
      using f85 by presburger
      then have f82:  $\langle (\lambda n. (\sum i = 0..< r. (\sum k < n. (\text{fps-const } (s' \ k \ i)) * (\text{fps-X}^\wedge(\text{subdegree } (\text{the-s } k) - m))))$ 
*from-nat-into  $\langle (B \ m) \ (p \ m \ i) \rangle \longrightarrow (\sum i = 0..< r. \text{Abs-fps } \langle (\lambda k. s \ k \ i) \rangle * \text{from-nat-into}$ 
 $\langle (B \ m) \ (p \ m \ i) \rangle)$ 
      for r
      proof (induct r)
      case 0
      then show ?case by simp
      next
      case 1: (Suc r)
      have  $\langle (\lambda n. (\sum k < n. \text{fps-const } (s' \ k \ r) * \text{fps-X}^\wedge(\text{subdegree } (\text{the-s } k) - m)))$ 
*
from-nat-into  $\langle (B \ m) \ (p \ m \ r) \rangle \longrightarrow \text{Abs-fps } \langle (\lambda k. s \ k \ r) \rangle * \text{from-nat-into } \langle (B \ m) \ (p \ m \ r) \rangle$ 
      proof -
      have  $\langle (\lambda n. \text{from-nat-into } \langle (B \ m) \ (p \ m \ r) \rangle * (\sum n < n. \text{fps-const } (s' \ n \ r) * \text{fps-X}^\wedge(\text{subdegree } (\text{the-s } n) - m))) \longrightarrow \text{from-nat-into } \langle (B \ m) \ (p \ m \ r) \rangle * \text{Abs-fps}$ 
 $\langle (\lambda n. s \ n \ r) \rangle$ 
      using LIMSEQ-cmult-fps f85 f86 by presburger
      then show ?thesis
      by (simp add: mult.commute)
      qed
      then show ?case
      apply (subst atLeast0-lessThan-Suc)
      by (simp add: 1 LIMSEQ-add-fps add.commute)
      qed
      have f83:  $\langle (\sum i = 0..< r. \text{Abs-fps } \langle (\lambda k. s \ k \ i) \rangle * \text{from-nat-into } \langle (B \ m) \ (p \ m \ i) \rangle) \in$ 
 $I' \rangle$  for r
      proof (induct r)
      case 0
      then show ?case
      by (metis (full-types) FPS-ring-def I'-def add-stable-FPS-ring atLeast-LessThan0 diff-0
diff-add-cancel f53 in-I' partial-object.select-conv(1) ring.genideal-ideal ring-FPS subset-UNIV sum.empty)
      next
      case 1: (Suc r)

```

```

have ⟨ from-nat-into (B m) (p m r) ∈ I' ⟩
  unfolding I'-def genideal-def apply(clarify)
by (metis (no-types, lifting) UN-subset-iff ab-group-add-class.ab-diff-conv-add-uminus
add.right-neutral
      add-diff-cancel-left' atLeastLessThan0 atMost-iff card.empty f53
from-nat-into in-mono less-irrefl-nat order-refl sum.empty)
  with 1 show ?case apply(clarsimp)
by (metis (no-types, lifting) 1 FPS-ring-def Formal-Power-Series-Ring.genideal-sum-rep

      Formal-Power-Series-Ring.idl-sum I'-def UNIV-I
      add-stable-FPS-ring f62 ideal.I-l-closed monoid.select-convs(1) par-
      tial-object.select-convs(1))
qed
then have ⟨ f+g ∈ I' ⟩
proof –
  have  $\bigwedge n. \text{lim } (\lambda na. \sum n = 0..<n. (\sum na < na. \text{fps-const } (s' \text{ na } n) * \text{fps-X}$ 
 $\wedge (\text{subdegree } (\text{the-s na}) - m))$ 
 $* \text{from-nat-into } (B \text{ m}) (p \text{ m } n)) = (\sum n = 0..<n. \text{Abs-fps } (\lambda na. s \text{ na } n) *$ 
from-nat-into (B m) (p m n))
    by (smt (z3) f82 limI)
  then show ?thesis
    using f83 f84 by presburger
qed
then show False
  using hyps(4) by force
qed
then have ⟨ I = I' ⟩
  using ⟨ I' ⊆ I ⟩ by fastforce
then show ⟨  $\exists A \subseteq \text{carrier local.FPS-ring. finite } A \wedge I = \text{Idl}_{\text{local.FPS-ring } A}$  ⟩
  by (metis FPS-ring-def I'-def f62 partial-object.select-convs(1) subset-UNIV)
qed
qed
end
end

```

## 7 The Real Ring definition

**theory** Real-Ring-Definition

**imports**

HOL-Algebra.Module

HOL-Algebra.RingHom

HOL.Real

HOL-Computational-Algebra.Formal-Power-Series

**begin**

Defining real ring for examples on Noetherian Rings.

**definition**

```
REAL :: real ring
where REAL = (|carrier = UNIV, monoid.mult = (*), one = 1, zero = 0, add
= (+)|)
```

**lemma** *REAL-ring*: $\langle$ ring REAL $\rangle$

```
apply(rule ringI)
  apply(rule abelian-groupI)
  by (auto simp:REAL-def monoidI ab-group-add-class.ab-left-minus distrib-right
distrib-left
  intro: exI[of - - x for x])
```

**lemma** *REAL-cring*: $\langle$ cring REAL $\rangle$

```
unfolding cring-def apply(safe)
  apply (simp add: REAL-ring)
  apply(rule comm-monoidI)
  by(auto simp:REAL-def)
```

**lemma** *REAL-field*: $\langle$ field REAL $\rangle$

```
unfolding field-def domain-def field-axioms-def
  apply(safe)
  apply(simp add:REAL-cring)
  unfolding domain-axioms-def
  by(auto simp:REAL-def Units-def mult.commute nonzero-divide-eq-eq)
  (metis mult.commute nonzero-divide-eq-eq)
```

end

## 8 Examples

**theory** *Examples-Noetherian-Rings*

**imports**

```
Hilbert-Basis
Real-Ring-Definition
```

**begin**

### 8.1 Examples of noetherian rings with $\mathbb{Z}$ and $\mathbb{Z}[X]$

**lemma** *INTEG-euclidean-domain*: $\langle$ euclidean-domain INTEG ( $\lambda x.$  nat (*abs*  $x$ )) $\rangle$

```
apply(rule domain.euclidean-domainI)
  unfolding domain-def domain-axioms-def using INTEG-cring apply(simp add:INTEG-def)
  unfolding INTEG-def
  using abs-mod-less div-mod-decomp-int mult.commute
  by (metis Diff-iff INTEG.R.r-null INTEG-def INTEG-mult UNIV-I abs-ge-zero
insert-iff
  mult-zero-left nat-less-eq-zless partial-object.select-convs(1) ring-record-simps(12))
```

```

lemma principal-ideal-INTEG: $\langle ideal\ I\ INTEG \implies principalideal\ I\ INTEG \rangle$ 
proof(rule principalidealI)
  assume  $h:\langle ideal\ I\ INTEG \rangle$ 
  then show  $\langle ideal\ I\ INTEG \rangle$  by(simp)
  {assume  $h1:\langle I \neq \{0\} \rangle$ 
    define  $E$  where  $imp:\langle E \equiv \{nat\ (abs\ x) \mid x.\ x \in I \wedge x \neq 0\} \rangle$ 
    then have  $\langle E \neq \{\} \rangle$ 
      using  $h\ h1\ additive-subgroup.zero-closed$  unfolding ideal-def
      by fastforce
    then have  $\langle \exists n \in E.\ \forall x \in E.\ n \leq x \wedge n > 0 \rangle$ 
      using  $abs-ge-zero\ imp\ zero-less-abs-iff$ 
      by (smt (verit) all-not-in-conv\ exists-least-iff\ gr-zeroI\ leI\ mem-Collect-eq\ nat-0-iff)
    define  $E'$  where  $imp2:\langle E' \equiv \{(abs\ x) \mid x.\ x \in I \wedge x \neq 0\} \rangle$ 
    then have  $\langle bij-betw\ nat\ E'\ E \rangle$ 
      unfolding bij-betw-def
      apply(safe)
      using inj-on-def apply force
      using imp apply blast
      using imp by blast
    then have  $\langle \exists n \in E'.\ \forall x \in E'.\ n \leq x \wedge n > 0 \rangle$ 
      by (smt (verit, best)  $\langle \exists n \in E.\ \forall x \in E.\ n \leq x \wedge 0 < n \rangle$ 
        bij-betw-iff-bijections\ le-nat-iff\ nat-eq-iff2\ nat-le-iff\ zero-less-nat-eq)
    then obtain  $n$  where  $f1:\langle \forall x \in E'.\ n \leq x \wedge n > 0 \wedge n \in E' \rangle$  by blast
    then have  $\langle \exists a \in I.\ abs\ a = n \rangle$ 
      using  $\langle \exists n \in E'.\ \forall x \in E'.\ n \leq x \wedge 0 < n \rangle$  imp2 by blast
    then obtain  $a$  where  $f0:\langle a \in I \wedge abs\ a = n \rangle$  by blast
    then have  $\langle \forall x.\ \exists q\ r.\ x = a * q + r \wedge abs\ r < abs\ a \rangle$ 
      using INTEG-euclidean-domain unfolding euclidean-domain-def
      by (metis  $\langle \exists n \in E'.\ \forall x \in E'.\ n \leq x \wedge 0 < n \rangle$   $\langle \forall x \in E'.\ n \leq x \wedge 0 < n \wedge n \in E' \rangle$ 
        abs-mod-less\ div-mod-decomp-int\ mult.commute\ zero-less-abs-iff)
    then have  $f2:\langle x \in I \wedge r = x - a * q \implies r \in I \rangle$  for  $q\ r\ x$ 
      using  $h$  unfolding ideal-def\ INTEG-def\ additive-subgroup-def\ subgroup-def\ ideal-axioms-def
      ring-def apply(safe, simp)
    by (metis  $\langle a \in I \wedge |a| = n \rangle$  integer-group-def\ inv-integer-group\ uminus-add-conv-diff)
    have  $f3:\langle x \in I \wedge r = x - a * q \implies abs\ r < abs\ a \implies r = 0 \rangle$  for  $r\ q\ x$ 
      apply(frule  $f2$ )
      using imp2\ f1\ f0
      by fastforce
    have  $\langle x \in I \wedge r = x - a * q \implies abs\ r < abs\ a \implies x \in Idl_{INTEG}\ \{a\} \rangle$ 
      for  $x\ r\ q$ 
      apply(frule  $f3$ )
      apply blast
    unfolding genideal-def\ ideal-def\ INTEG-def\ additive-subgroup-def\ subgroup-def\ ideal-axioms-def by(auto)
    then have  $\langle x \in I \implies x \in Idl_{INTEG}\ \{a\} \rangle$  for  $x$ 
      by (metis  $\langle \forall x.\ \exists q\ r.\ x = a * q + r \wedge |r| < |a| \rangle$  add-diff-cancel-left')
  }

```

```

then have  $\langle \text{ideal } I \text{ INTEG} \implies \exists i \in \text{carrier INTEG. } I = \text{Idl}_{\text{INTEG}} \{i\} \rangle$ 
using INTEG.R.cgenideal-eq-genideal INTEG.R.cgenideal-minimal f0 by blast
note non-trivial-ideal=this
show  $\langle \exists i \in \text{carrier INTEG. } I = \text{Idl}_{\text{INTEG}} \{i\} \rangle$ 
apply (cases I={0})
apply (metis INTEG.R.genideal-self
INTEG.R.ring-axioms INTEG-closed h ring.Idl-subset-ideal subsetI sub-
set-antisym)
using non-trivial-ideal h by auto
qed

```

```

lemma INTEG-noetherian-ring:⟨noetherian-ring INTEG⟩
apply (rule ring.noetherian-ringI)
apply (simp add: INTEG.R.ring-axioms)
using principal-ideal-INTEG unfolding principalideal-def
by (meson INTEG-closed finite.emptyI finite-insert principalideal-axioms-def sub-
setI)

```

```

lemma INTEG-noetherian-domain:⟨noetherian-domain INTEG⟩
unfolding noetherian-domain-def
using INTEG-noetherian-ring INTEG-euclidean-domain euclidean-domain.axioms(1)
by blast

```

```

lemma Polynomials-INTEG-noetherian-ring:⟨noetherian-ring (univ-poly INTEG
(carrier INTEG))⟩
by (simp add: INTEG-noetherian-domain noetherian-domain.weak-Hilbert-basis)

```

```

lemma Polynomials-INTEG-noetherian-domain:⟨noetherian-domain (univ-poly IN-
TEG (carrier INTEG))⟩
using INTEG.R.ring-axioms INTEG-noetherian-domain Polynomials-INTEG-noetherian-ring
domain.univ-poly-is-domain noetherian-domain.axioms(2) noetherian-domain.intro
ring.carrier-is-subring by blast

```

## 8.2 Another example with $\mathbb{R}$ and $\mathbb{R}[X]$

```

lemma REAL-noetherian-domain:⟨noetherian-domain REAL⟩
unfolding noetherian-domain-def
by (simp add: REAL-field domain.noetherian-RX-imp-noetherian-R domain.univ-poly-is-principal
field.axioms(1) field.carrier-is-subfield principal-imp-noetherian)

```

```

lemma PolyREAL-noetherian-domain:⟨noetherian-domain (univ-poly REAL (carrier
REAL))⟩
unfolding noetherian-domain-def
by (simp add: REAL-field REAL-noetherian-domain REAL-ring domain.univ-poly-is-domain

```



*field.axioms(1) noetherian-domain.weak-Hilbert-basis ring.carrier-is-subring)*

**end**

## References

- [1] Aaron Crighton *p-adic Fields and p-adic Semialgebraic Sets* , Archive of Formal Proofs, September 2022 [https://www.isa-afp.org/entries/Padic\\_Field.html](https://www.isa-afp.org/entries/Padic_Field.html)
- [2] Stack project <https://stacks.math.columbia.edu/tag/00FM>.
- [3] Vincent Douce <https://agreg-maths.fr/uploads/versions/1458/preview.pdf>.
- [4] Wiedijk's catalogue "Formalizing 100 Theorems" <https://www.cs.ru.nl/~freek/100/>, It appears at position 121 ...