

Fundamental Properties of Valuation Theory and Hensel's Lemma

Hidetsune Kobayashi

September 13, 2023

Abstract

Convergence with respect to a valuation is discussed as convergence of a Cauchy sequence. Cauchy sequences of polynomials are defined. They are used to formalize Hensel's lemma.

Contents

1 Preliminaries	2
1.1 Int and ant (augmented integers)	2
1.2 nsets	7
2 Elementary properties of a valuation	14
2.1 Definition of a valuation	14
2.2 The normal valuation of v	19
2.3 Valuation ring	30
2.4 Ideals in a valuation ring	38
2.4.1 Amin lemma (in Corps)s	51
2.5 pow of vp and <i>n</i> -value – convergence –	52
2.6 Equivalent valuations	65
2.7 Prime divisors	68
2.8 Approximation	71
2.8.1 Representation of an ideal I as a product of prime ideals	125
2.8.2 prime- <i>n</i> -pd	137
2.9 Completion	140
2.9.1 Hensel's theorem	179

```
theory Valuation1
imports Group-Ring-Module.Algebra9
begin

declare ex-image-cong-iff [simp del]
```

Chapter 1

Preliminaries

1.1 Int and ant (augmented integers)

```
lemma int-less-mono:(a::nat) < b ==> int a < int b
apply simp
done

lemma zless-trans:[(i::int) < j; j < k] ==> i < k
apply simp
done

lemma zmult-pos-bignumTr0: $\exists L. \forall m. L < m \longrightarrow z < x + \text{int } m$ 
by (subgoal-tac  $\forall m. (\text{nat}((\text{abs } z) + (\text{abs } x))) < m \longrightarrow z < x + \text{int } m$ ,
blast, rule allI, rule impI, arith)

lemma zle-less-trans:[(i::int)  $\leq j$ ; j < k] ==> i < k
apply (simp add:less-le)
done

lemma zless-le-trans:[(i::int) < j; j  $\leq k] ==> i < k
apply (simp add:less-le)
done

lemma zmult-pos-bignumTr0:< (a::int) ==>
 $\exists l. \forall m. l < m \longrightarrow z < x + (\text{int } m) * a$ 
apply (cut-tac zmult-pos-bignumTr0[of z x])
apply (erule exE)
apply (subgoal-tac  $\forall m. L < m \longrightarrow z < x + \text{int } m * a$ , blast)
apply (rule allI, rule impI)
apply (drule-tac a = m in forall-spec, assumption)
apply (subgoal-tac  $0 \leq \text{int } m$ )
apply (frule-tac a = int m and b = a in pos-zmult-pos, assumption)
apply (cut-tac order-refl[of x])
apply (frule-tac z' = int m and z = int m * a in
zadd-zle-mono[of x x], assumption+)$ 
```

```

apply (rule-tac  $y = x + \text{int } m$  and  $z = x + (\text{int } m)* a$  in
      less-le-trans[of  $z$ ], assumption+)
apply simp
done

lemma ale-shift:[ $(x::\text{ant}) \leq y; y = z \Rightarrow x \leq z$ ]
by simp

lemma aneg-na-0[simp]: $a < 0 \Rightarrow \text{na } a = 0$ 
by (simp add:na-def)

lemma amult-an-an:an ( $m * n$ ) = ( $\text{an } m$ ) * ( $\text{an } n$ )
apply (simp add:an-def)
apply (simp add: of-nat-mult a-z-z)
done

definition
adiv :: [ant, ant]  $\Rightarrow$  ant (infixl adiv 200) where
   $x \text{ adiv } y = \text{ant} ((\text{tna } x) \text{ div } (\text{tna } y))$ 

definition
amod :: [ant, ant]  $\Rightarrow$  ant (infixl amod 200) where
   $x \text{ amod } y = \text{ant} ((\text{tna } x) \text{ mod } (\text{tna } y))$ 

lemma apos-amod-conj: $0 < \text{ant } b \Rightarrow$ 
   $0 \leq (\text{ant } a) \text{ amod } (\text{ant } b) \wedge (\text{ant } a) \text{ amod } (\text{ant } b) < (\text{ant } b)$ 
by (simp add:amod-def tna-ant, simp only:ant-0[THEN sym],
      simp add:aless-zless)

lemma amod-adiv-equality:
   $(\text{ant } a) = (\text{a div } b) *_a (\text{ant } b) + \text{ant} (\text{a mod } b)$ 
apply (simp add:adiv-def tna-ant a-z-z a-zpz asprod-mult)
done

lemma asp-z-Z:z *_a ant  $x \in Z_\infty$ 
by (simp add:asprod-mult z-in-aug-inf)

lemma apos-in-aug-inf: $0 \leq a \Rightarrow a \in Z_\infty$ 
by (simp add:aug-inf-def, rule contrapos-pp, simp+,
      cut-tac minf-le-any[of 0], frule ale-antisym[of 0 -∞],
      assumption+, simp)

lemma amult-1-both:[ $0 < (w::\text{ant}); x * w = 1 \Rightarrow x = 1 \wedge w = 1$ ]
apply (cut-tac mem-ant[of  $x$ ], cut-tac mem-ant[of  $w$ ],
      (erule disjE)+, simp,
      (frule sym, thin-tac  $\infty = 1$ , simp only:ant-1[THEN sym],
      simp del:ant-1))
apply (erule disjE, erule exE, simp,
      (frule sym, thin-tac  $-\infty = 1$ , simp only:ant-1[THEN sym],
      simp del:ant-1))

```

```

simp del:ant-1), simp)
apply (frule sym, thin-tac  $-\infty = 1$ , simp only:ant-1[THEN sym],
      simp del:ant-1)
apply ((erule disjE)+, erule exE, simp,
       frule-tac aless-imp-le[of 0  $-\infty$ ],
       cut-tac minf-le-any[of 0],
       frule ale-antisym[of 0  $-\infty$ ], assumption+,
       simp only:ant-0[THEN sym], simp,
       frule sym, thin-tac  $-\infty = 1$ , simp only:ant-1[THEN sym],
       simp del:ant-1)
apply ((erule disjE)+, (erule exE)+, simp only:ant-1[THEN sym],
       simp del:ant-1 add:a-z-z,
       (cut-tac a = z and b = za in mult.commute, simp,
        cut-tac z = za and z' = z in times-1-both, assumption+),
       simp)
apply (erule exE, simp,
       cut-tac x = z and y = 0 in less-linear, erule disjE, simp,
       frule sym, thin-tac  $-\infty = 1$ , simp only:ant-1[THEN sym],
       simp del:ant-1,
       erule disjE, simp add:ant-0, simp,
       frule sym, thin-tac  $\infty = 1$ , simp only:ant-1[THEN sym],
       simp del:ant-1,
       erule disjE, erule exE, simp,
       frule sym, thin-tac  $\infty = 1$ , simp only:ant-1[THEN sym],
       simp del:ant-1, simp)
done

lemma poss-int-neq-0:0 < (z::int)  $\Rightarrow z \neq 0$ 
by simp

lemma aadd-neg-negg[simp]: $\llbracket a \leq (0::ant); b < 0 \rrbracket \Rightarrow a + b < 0$ 
apply (frule ale-minus[of a 0], simp,
       frule aless-minus[of b 0], simp)
apply (frule aadd-pos-poss[of  $-a - b$ ], assumption+,
       simp add:aminus-add-distrib[THEN sym, of a b],
       frule aless-minus[of 0  $-(a + b)$ ], simp add:a-minus-minus)
done

lemma aadd-two-negg[simp]: $\llbracket a < (0::ant); b < 0 \rrbracket \Rightarrow a + b < 0$ 
by auto

lemma amin-aminTr:(z::ant)  $\leq z' \Rightarrow \text{amin } z w \leq \text{amin } z' w$ 
by (simp add:amin-def)

lemma amin-le1:(z::ant)  $\leq z' \Rightarrow (\text{amin } z w) \leq z'$ 
by (simp add:amin-def, simp add:aneg-le,
       rule impI, frule aless-le-trans[of w z z'], assumption+, simp add:aless-imp-le)

```

```

lemma amin-le2:(z::ant) ≤ z'  $\implies$  (amin w z) ≤ z'
by (simp add:amin-def, rule impI,
      frule ale-trans[of w z z'], assumption+)

lemma Amin-geTr:( $\forall j \leq n. f j \in Z_\infty \wedge (\forall j \leq n. z \leq (f j)) \longrightarrow$ 
                   $z \leq (Amin n f)$ )
apply (induct-tac n)
apply (rule impI, erule conjE, simp)
apply (rule impI, (erule conjE)+,
        cut-tac Nsetn-sub-mem1[of n], simp,
        drule-tac x = Suc n in spec, simp,
        rule-tac z = z and x = Amin n f and y = f(Suc n) in amin-ge1,
        simp+)
done

lemma Amin-ge:[ $\forall j \leq n. f j \in Z_\infty; \forall j \leq n. z \leq (f j)] \implies$ 
                   $z \leq (Amin n f)$ 
by (simp add:Amin-geTr)

definition
Abs :: ant  $\Rightarrow$  ant where
Abs z = (if z < 0 then -z else z)

lemma Abs-pos:0 ≤ Abs z
by (simp add:Abs-def, rule conjI, rule impI,
      cut-tac aless-minus[of z 0], simp,
      assumption,
      rule impI, simp add:aneg-less[of z 0])

lemma Abs-x-plus-x-pos:0 ≤ (Abs x) + x
apply (case-tac x < 0,
      simp add:Abs-def, simp add:aadd-minus-inv)

apply (simp add:aneg-less,
      simp add:Abs-def, simp add:aneg-less[THEN sym, of 0 x],
      simp add:aneg-less[of x 0], simp add:aadd-two-pos)
done

lemma Abs-ge-self:x ≤ Abs x
apply (simp add:Abs-def, rule impI,
      cut-tac ale-minus[of x 0],
      simp add:aminus-0, simp add:aless-imp-le)
done

lemma na-1:na 1 = Suc 0
apply (simp only:ant-1[THEN sym], simp only:na-def,
      simp only:ant-0[THEN sym], simp only:aless-zless[of 1 0],
      simp, subgoal-tac  $\infty \neq 1$ , simp)
apply (simp only:ant-1[THEN sym], simp only:tna-ant,
      simp only:ant-0[THEN sym], simp only:na-def)
done

```

```

rule not-sym, simp only:ant-1[THEN sym], simp del:ant-1)
done

lemma ant-int:ant (int n) = an n
by (simp add:an-def)

lemma int-nat:0 < z ==> int (nat z) = z
by arith

lemma int-ex-nat:0 < z ==>  $\exists n. \text{int } n = z$ 
by (cut-tac int-nat[of z], blast, assumption)

lemma eq-nat-pos-ints:
   $[\text{nat } (z::\text{int}) = \text{nat } (z'::\text{int}); 0 \leq z; 0 \leq z'] \implies z = z'$ 
by simp

lemma a-p1-gt[simp]: $[\![a \neq \infty; a \neq -\infty]\!] \implies a < a + 1$ 
apply (cut-tac aadd-poss-less[of a 1],
       simp add:aadd-commute, assumption+)
apply (cut-tac zposs-aposss[of 1], simp)
done

lemma gt-na-poss:(na a) < m ==> 0 < m
apply (simp add:na-def)
done

lemma azmult-less: $[\![a \neq \infty; na a < m; 0 < x]\!]$ 
   $\implies a < \text{int } m *_a x$ 
apply (cut-tac mem-ant[of a])
apply (erule disjE)
apply (case-tac x =  $\infty$ ) apply simp
apply (subst less-le[of  $-\infty \infty$ ]) apply simp
apply (frule aless-imp-le[of 0 x], frule apos-neq-minf[of x])
apply (cut-tac mem-ant[of x], simp, erule exE, simp)
apply (simp add:asprod-amult a-z-z)
apply (simp, erule exE, simp)

apply (frule-tac a = ant z in gt-na-poss[of - m])
apply (case-tac x =  $\infty$ , simp)
apply (frule aless-imp-le[of 0 x])
apply (frule apos-neq-minf[of x])
apply (cut-tac mem-ant[of x], simp, erule exE,
       simp add:asprod-amult a-z-z)
apply (subst aless-zless)
apply (cut-tac a = ant z in gt-na-poss[of - m], assumption)
apply (smt a0-less-int-conv aposs-na-poss int-less-mono int-nat na-def of-nat-0-le-iff
      pos-zmult-pos tna-ant z-neq-inf)
done

```

lemma *zmult-gt-one*: $\llbracket 2 \leq m; 0 < xa \rrbracket \implies 1 < \text{int } m * xa$
by (*metis ge2-zmult-pos mult.commute*)

lemma *zmult-pos*: $\llbracket 0 < m; 0 < (a:\text{int}) \rrbracket \implies 0 < (\text{int } m) * a$
by (*frule zmult-zless-mono2[of 0 a int m], simp, simp*)

lemma *ant-int-na*: $\llbracket 0 \leq a; a \neq \infty \rrbracket \implies \text{ant}(\text{int}(na)) = a$
by (*frule an-na[of a], assumption, simp add:an-def*)

lemma *zpos-nat*: $0 \leq (z:\text{int}) \implies \exists n. z = \text{int } n$
apply (*subgoal-tac z = int(nat z)*)
apply *blast apply simp*
done

1.2 nsets

lemma *nsetTr1*: $\llbracket j \in \text{nset } a \ b; j \neq a \rrbracket \implies j \in \text{nset } (\text{Suc } a) \ b$
apply (*simp add:nset-def*)
done

lemma *nsetTr2*: $j \in \text{nset } (\text{Suc } a) \ (\text{Suc } b) \implies j - \text{Suc } 0 \in \text{nset } a \ b$
apply (*simp add:nset-def, erule conjE,*
simp add:skip-im-Tr4[of j b])
done

lemma *nsetTr3*: $\llbracket j \neq \text{Suc } (\text{Suc } 0); j - \text{Suc } 0 \in \text{nset } (\text{Suc } 0) \ (\text{Suc } n) \rrbracket$
 $\implies \text{Suc } 0 < j - \text{Suc } 0$
apply (*simp add:nset-def, erule conjE, subgoal-tac j \neq 0,*
rule contrapos-pp, simp+)
done

lemma *Suc-leD1*: $\text{Suc } m \leq n \implies m < n$
apply (*insert lessI[of m],*
rule less-le-trans[of m Suc m n], assumption+)
done

lemma *leI1*: $n < m \implies \neg ((m:\text{nat}) \leq n)$
apply (*rule contrapos-pp, simp+*)
done

lemma *neg-zle*: $\neg (z:\text{int}) \leq z' \implies z' < z$
apply (*simp add: not-le*)
done

lemma *nset-m-m*: $\text{nset } m \ m = \{m\}$
by (*simp add:nset-def,*
rule equalityI, rule subsetI, simp,
rule subsetI, simp)

```

lemma nset-Tr51: $\llbracket j \in \text{nset}(\text{Suc } 0) (\text{Suc}(\text{Suc } n)); j \neq \text{Suc } 0 \rrbracket$ 
 $\implies j - \text{Suc } 0 \in \text{nset}(\text{Suc } 0) (\text{Suc } n)$ 
apply (simp add:nset-def, (erule conjE)+,
      frule-tac  $m = j$  and  $n = \text{Suc}(\text{Suc } n)$  and  $l = \text{Suc } 0$  in diff-le-mono,
      simp)
done

lemma nset-Tr52: $\llbracket j \neq \text{Suc}(\text{Suc } 0); \text{Suc } 0 \leq j - \text{Suc } 0 \rrbracket$ 
 $\implies \neg j - \text{Suc } 0 \leq \text{Suc } 0$ 
by auto

lemma nset-Suc:nset ( $\text{Suc } 0$ ) ( $\text{Suc}(\text{Suc } n)$ ) =
 $\text{nset}(\text{Suc } 0) (\text{Suc } n) \cup \{\text{Suc}(\text{Suc } n)\}$ 
by (auto simp add:nset-def)

lemma AinequalityTr0: $x \neq -\infty \implies \exists L. (\forall N. L < N \longrightarrow$ 
 $(an m) < (x + an N))$ 
apply (case-tac  $x = \infty$ , simp add:an-def)
apply (cut-tac mem-ant[of  $x$ ], simp, erule exE, simp add:an-def a-zpz,
      simp add:aless-zless,
      cut-tac  $x = z$  in zmult-pos-bignumTr0[of int  $m$ ], simp)
done

lemma AinequalityTr: $\llbracket 0 < b \wedge b \neq \infty; x \neq -\infty \rrbracket \implies \exists L. (\forall N. L < N \longrightarrow$ 
 $(an m) < (x + (int N) *_a b))$ 
apply (frule-tac AinequalityTr0[of  $x m$ ],
      erule exE,
      subgoal-tac  $\forall N. L < N \longrightarrow an m < x + (int N) *_a b$ ,
      blast, rule allI, rule impI)
apply (drule-tac  $a = N$  in forall-spec, assumption,
      erule conjE,
      cut-tac  $N = N$  in asprod-ge[of  $b$ ], assumption,
      thin-tac  $x \neq -\infty$ , thin-tac  $b \neq \infty$ , thin-tac  $an m < x + an N$ ,
      simp)
apply (frule-tac  $x = an N$  and  $y = int N *_a b$  and  $z = x$  in aadd-le-mono,
      simp only:aadd-commute[of -  $x$ ])
done

lemma two-inequalities: $\llbracket \forall (n::nat). x < n \longrightarrow P n; \forall (n::nat). y < n \longrightarrow Q n \rrbracket$ 
 $\implies \forall n. (\max x y) < n \longrightarrow (P n) \wedge (Q n)$ 
by auto

lemma multi-inequalityTr0: $(\forall j \leq (n::nat). (x j) \neq -\infty) \longrightarrow$ 
 $(\exists L. (\forall N. L < N \longrightarrow (\forall l \leq n. (an m) < (x l) + (an N))))$ 
apply (induct-tac  $n$ )
apply (rule impI, simp)
apply (rule AinequalityTr0[of  $x 0 m$ ], assumption)

apply (rule impI)

```

```

apply (subgoal-tac  $\forall l. l \leq n \rightarrow l \leq (\text{Suc } n)$ , simp)
apply (erule exE)
apply (frule-tac  $a = \text{Suc } n$  in forall-spec, simp)

apply (frule-tac  $x = x (\text{Suc } n)$  in AinequalityTr0[of - m])
apply (erule exE)
apply (subgoal-tac  $\forall N. (\max L La) < N \rightarrow$ 
 $(\forall l \leq (\text{Suc } n). an m < x l + an N)$ , blast)
apply (rule allI, rule impI, rule allI, rule impI)
apply (rotate-tac 1)
apply (case-tac  $l = \text{Suc } n$ , simp,
drule-tac  $m = l$  and  $n = \text{Suc } n$  in noteq-le-less, assumption+,
drule-tac  $x = l$  and  $n = \text{Suc } n$  in less-le-diff, simp,
simp)
done

lemma multi-inequalityTr1: $\llbracket \forall j \leq (n::nat). (x j) \neq -\infty \rrbracket \implies$ 
 $\exists L. (\forall N. L < N \rightarrow (\forall l \leq n. (an m) < (x l) + (an N)))$ 
by (simp add:multi-inequalityTr0)

lemma gcoeff-multi-inequality: $\llbracket \forall N. 0 < N \rightarrow (\forall j \leq (n::nat). (x j) \neq -\infty \wedge$ 
 $0 < (b N j) \wedge (b N j) \neq \infty) \rrbracket \implies$ 
 $\exists L. (\forall N. L < N \rightarrow (\forall l \leq n. (an m) < (x l) + (\text{int } N) *_a (b N l)))$ 
apply (subgoal-tac  $\forall j \leq n. x j \neq -\infty$ )
apply (frule multi-inequalityTr1[of n x m])
apply (erule exE)
apply (subgoal-tac  $\forall N. L < N \rightarrow$ 
 $(\forall l \leq n. an m < x l + (\text{int } N) *_a (b N l))$ )
apply blast

apply (rule allI, rule impI, rule allI, rule impI,
drule-tac  $a = N$  in forall-spec, simp,
drule-tac  $a = l$  in forall-spec, assumption,
drule-tac  $a = N$  in forall-spec, assumption,
drule-tac  $a = l$  in forall-spec, assumption,
drule-tac  $a = l$  in forall-spec, assumption)
apply (cut-tac  $b = b N l$  and  $N = N$  in asprod-ge, simp, simp,
(erule conjE)+, simp, thin-tac  $x l \neq -\infty$ , thin-tac  $b N l \neq \infty$ )
apply (frule-tac  $x = an N$  and  $y = \text{int } N *_a b N l$  and  $z = x l$  in
aadd-le-mono, simp add:aadd-commute,
rule allI, rule impI,
cut-tac lessI[of (0::nat)],
drule-tac  $a = \text{Suc } 0$  in forall-spec, assumption)
apply simp
done

primrec m-max :: [nat, nat ⇒ nat] ⇒ nat
where
m-max-0: m-max 0 f = f 0

```

| *m-max-Suc*: $m\text{-max } (\text{Suc } n) f = \text{max } (m\text{-max } n f) (f (\text{Suc } n))$

```

lemma m-maxTr:  $\forall l \leq n. (f l) \leq m\text{-max } n f$ 
apply (induct-tac n)
apply simp

apply (rule allI, rule impI)
apply simp
apply (case-tac l = Suc n, simp)
apply (cut-tac m = l and n = Suc n in noteq-le-less, assumption+,
         thin-tac l ≤ Suc n, thin-tac l ≠ Suc n,
         frule-tac x = l and n = Suc n in less-le-diff,
         thin-tac l < Suc n, simp)
apply (drule-tac a = l in forall-spec, assumption)
apply simp
done

lemma m-max-gt:  $l \leq n \implies (f l) \leq m\text{-max } n f$ 
apply (simp add:m-maxTr)
done

lemma ASum-zero:  $(\forall j \leq n. f j \in Z_\infty) \wedge (\forall l \leq n. f l = 0) \longrightarrow \text{ASum } f n = 0$ 
apply (induct-tac n)
apply (rule impI, erule conjE, simp)
apply (rule impI)
apply (subgoal-tac  $(\forall j \leq n. f j \in Z_\infty) \wedge (\forall l \leq n. f l = 0)$ , simp)
apply (simp add:aadd-0-l, erule conjE,
         thin-tac  $(\forall j \leq n. f j \in Z_\infty) \wedge (\forall l \leq n. f l = 0) \longrightarrow \text{ASum } f n = 0$ )
apply (rule conjI)
apply (rule allI, rule impI,
         drule-tac a = j in forall-spec, simp, assumption+)
apply (thin-tac  $\forall j \leq \text{Suc } n. f j \in Z_\infty$ )
apply (rule allI, rule impI,
         drule-tac a = l in forall-spec, simp+)
done

lemma eSum-singleTr:  $(\forall j \leq n. f j \in Z_\infty) \wedge (j \leq n \wedge (\forall l \in \{h. h \leq n\} - \{j\}. f l = 0)) \longrightarrow \text{ASum } f n = f j$ 
apply (induct-tac n)
apply (simp, rule impI, (erule conjE)+)
apply (case-tac j ≤ n)
apply simp
apply (simp add:aadd-0-r)
apply simp
apply (simp add:nat-not-le-less[of j])
apply (frule-tac m = n and n = j in Suc-leI)
apply (frule-tac m = j and n = Suc n in le-antisym, assumption+, simp)

```

```

apply (cut-tac  $n = n$  in ASum-zero [of - f])
apply (subgoal-tac  $(\forall j \leq n. f j \in Z_\infty) \wedge (\forall l \leq n. f l = 0)$ )
apply (thin-tac  $\forall j \leq \text{Suc } n. f j \in Z_\infty,$ 
       $\text{thin-tac } \forall l \in \{h. h \leq \text{Suc } n\} - \{\text{Suc } n\}. f l = 0, \text{ simp only:mp}$ )
apply (simp add:aadd-0-l)

apply (thin-tac  $(\forall j \leq n. f j \in Z_\infty) \wedge (\forall l \leq n. f l = 0) \longrightarrow \text{ASum } f n = 0$ )
apply (rule conjI,
       $\text{thin-tac } \forall l \in \{h. h \leq \text{Suc } n\} - \{\text{Suc } n\}. f l = 0, \text{ simp}$ )
apply (thin-tac  $\forall j \leq \text{Suc } n. f j \in Z_\infty, \text{ simp}$ )
done

lemma eSum-single: $\llbracket \forall j \leq n. f j \in Z_\infty ; j \leq n; \forall l \in \{h. h \leq n\} - \{j\}. f l = 0 \rrbracket$ 
 $\implies \text{ASum } f n = f j$ 
apply (simp add:eSum-singleTr)
done

lemma ASum-eqTr: $\llbracket (\forall j \leq n. f j \in Z_\infty) \wedge (\forall j \leq n. g j \in Z_\infty) \wedge$ 
 $(\forall j \leq n. f j = g j) \longrightarrow \text{ASum } f n = \text{ASum } g n$ 
apply (induct-tac  $n$ )
apply (rule impI, simp)

apply (rule impI, (erule conjE)+)
apply simp
done

lemma ASum-eq: $\llbracket \forall j \leq n. f j \in Z_\infty; \forall j \leq n. g j \in Z_\infty; \forall j \leq n. f j = g j \rrbracket \implies$ 
 $\text{ASum } f n = \text{ASum } g n$ 
by (cut-tac ASum-eqTr[of n f g], simp)

definition
Kronecker-delta :: [nat, nat]  $\Rightarrow$  ant
 $((\delta_{- -}) [70, 71] 70)$  where
 $\delta_i j = (\text{if } i = j \text{ then } 1 \text{ else } 0)$ 

definition
K-gamma :: [nat, nat]  $\Rightarrow$  int
 $((\gamma_{- -}) [70, 71] 70)$  where
 $\gamma_i j = (\text{if } i = j \text{ then } 0 \text{ else } 1)$ 

abbreviation
TRANSPOS  $((\tau_{- -}) [90, 91] 90)$  where
 $\tau_i j == \text{transpos } i j$ 

lemma Kdelta-in-Zinf: $\llbracket j \leq (\text{Suc } n); k \leq (\text{Suc } n) \rrbracket \implies$ 
 $z *_a (\delta_j k) \in Z_\infty$ 
apply (simp add:Kronecker-delta-def)
apply (simp add:z-in-aug-inf Zero-in-aug-inf)

```

```

apply (simp add:asprod-n-0 Zero-in-aug-inf)
done

lemma Kdelta-in-Zinf1:[ $j \leq n; k \leq n \Rightarrow \delta_j k \in Z_\infty$ ]
apply (simp add:Kronecker-delta-def)
apply (simp add:z-in-aug-inf Zero-in-aug-inf)
apply (rule impI)
apply (simp only:ant-1[THEN sym], simp del:ant-1 add:z-in-aug-inf)
done

primrec m-zmax :: [nat, nat  $\Rightarrow$  int]  $\Rightarrow$  int
where
m-zmax-0: m-zmax 0 f = f 0
| m-zmax-Suc: m-zmax (Suc n) f = zmax (m-zmax n f) (f (Suc n))

lemma m-zmax-gt-eachTr:
 $(\forall j \leq n. f j \in Zset) \longrightarrow (\forall j \leq n. (f j) \leq m\text{-}zmax n f)$ 
apply (induct-tac n)
apply (rule impI, rule allI, rule impI, simp)
apply (rule impI)
apply simp
apply (rule allI, rule impI)
apply (case-tac j = Suc n, simp)
apply (simp add:zmax-def)
apply (drule-tac m = j and n = Suc n in noteq-le-less, assumption,
       drule-tac x = j and n = Suc n in less-le-diff, simp)
apply (drule-tac a = j in forall-spec, assumption)
apply (simp add:zmax-def)
done

lemma m-zmax-gt-each:( $\forall j \leq n. f j \in Zset \Rightarrow (\forall j \leq n. (f j) \leq m\text{-}zmax n f)$ )
apply (simp add:m-zmax-gt-eachTr)
done

lemma n-notin-Nset-pred:  $0 < n \Rightarrow \neg n \leq (n - Suc 0)$ 
apply simp
done

lemma Nset-preTr:[ $0 < n; j \leq (n - Suc 0) \Rightarrow j \leq n$ ]
apply simp
done

lemma Nset-preTr1:[ $0 < n; j \leq (n - Suc 0) \Rightarrow j \neq n$ ]
apply simp
done

lemma transpos-noteqTr:[ $0 < n; k \leq (n - Suc 0); j \leq n; j \neq n \Rightarrow j \neq (\tau_j n) k$ ]
apply (rule contrapos-pp, simp+)

```

```
apply (simp add:transpos-def)
apply (case-tac k = j, simp, simp)
apply (case-tac k = n, simp)
apply (simp add:n-notin-Nset-pred)
done
```

Chapter 2

Elementary properties of a valuation

2.1 Definition of a valuation

definition

```
valuation :: [('b, 'm) Ring-scheme, 'b ⇒ ant] ⇒ bool where
valuation K v ↔
  v ∈ extensional (carrier K) ∧
  v ∈ carrier K → Z∞ ∧
  v (0K) = ∞ ∧ (∀x ∈ (carrier K) − {0K}). v x ≠ ∞ ∧
  (∀x ∈ (carrier K). ∀y ∈ (carrier K). v (x ·rK y) = (v x) + (v y)) ∧
  (∀x ∈ (carrier K). 0 ≤ (v x) → 0 ≤ (v (1rK ±K x))) ∧
  (∃x. x ∈ carrier K ∧ (v x) ≠ ∞ ∧ (v x) ≠ 0)
```

lemma (in Corps) invf-closed: $x \in \text{carrier } K - \{0\} \implies x^{-K} \in \text{carrier } K$
by (cut-tac *invf-closed1*[of x], simp, assumption)

lemma (in Corps) valuation-map: $\text{valuation } K v \implies v \in \text{carrier } K \rightarrow Z_\infty$
by (simp add:valuation-def)

lemma (in Corps) value-in-aug-inf: $\llbracket \text{valuation } K v; x \in \text{carrier } K \rrbracket \implies$
 $v x \in Z_\infty$
by (simp add:valuation-def, (erule conjE)+, simp add:funcset-mem)

lemma (in Corps) value-of-zero: $\text{valuation } K v \implies v (0) = \infty$
by (simp add:valuation-def)

lemma (in Corps) val nonzero noninf: $\llbracket \text{valuation } K v; x \in \text{carrier } K; x ≠ 0 \rrbracket \implies (v x) ≠ \infty$
by (simp add:valuation-def)

lemma (in Corps) value-inf-zero: $\llbracket \text{valuation } K v; x \in \text{carrier } K; v x = \infty \rrbracket \implies x = 0$
by (rule contrapos-pp, simp+,

```

frule val-nonzero-noninf[of v x], assumption+, simp)

lemma (in Corps) val-nonzero-z:[valuation K v; x ∈ carrier K; x ≠ 0] ==>
  ∃ z. (v x) = ant z
by (frule value-in-aug-inf[of v x], assumption+,
  frule val-nonzero-noninf[of v x], assumption+,
  cut-tac mem-ant[of v x], simp add:aug-inf-def)

lemma (in Corps) val-nonzero-z-unique:[valuation K v; x ∈ carrier K; x ≠ 0]
  ==> ∃!z. (v x) = ant z
by (rule ex-exI, simp add:val-nonzero-z, simp)

lemma (in Corps) value-noninf-nonzero:[valuation K v; x ∈ carrier K; v x ≠ ∞]
  ==> x ≠ 0
by (rule contrapos-pp, simp+, simp add:value-of-zero)

lemma (in Corps) val1-neq-0:[valuation K v; x ∈ carrier K; v x = 1] ==>
  x ≠ 0
apply (rule contrapos-pp, simp+, simp add:value-of-zero)
apply (simp only:ant-1[THEN sym], cut-tac z-neq-inf[THEN not-sym, of 1], simp)
done

lemma (in Corps) val-Zmin-sym:[valuation K v; x ∈ carrier K; y ∈ carrier K]
  ==> amin (v x) (v y) = amin (v y) (v x)
by (simp add:amin-commute)

lemma (in Corps) val-t2p:[valuation K v; x ∈ carrier K; y ∈ carrier K]
  ==> v (x ·r y) = v x + v y
by (simp add:valuation-def)

lemma (in Corps) val-axiom4:[valuation K v; x ∈ carrier K; 0 ≤ v x] ==>
  0 ≤ v (1r ± x)
by (simp add:valuation-def)

lemma (in Corps) val-axiom5:valuation K v ==>
  ∃ x. x ∈ carrier K ∧ v x ≠ ∞ ∧ v x ≠ 0
by (simp add:valuation-def)

lemma (in Corps) val-field-nonzero:valuation K v ==> carrier K ≠ {0}
by (rule contrapos-pp, simp+,
  frule val-axiom5[of v],
  erule exE, (erule conjE)+, simp add:value-of-zero)

lemma (in Corps) val-field-1-neq-0:valuation K v ==> 1r ≠ 0
apply (rule contrapos-pp, simp+)
apply (frule val-axiom5[of v])
apply (erule exE, (erule conjE)+)
apply (cut-tac field-is-ring,
  frule-tac t = x in Ring.ring-l-one[THEN sym, of K], assumption+,

```

```

simp add:Ring.ring-times-0-x, simp add:value-of-zero)
done

lemma (in Corps) value-of-one:valuation K v  $\implies$  v (1r) = 0
apply (cut-tac field-is-ring, frule Ring.ring-one[of K])
apply (frule val-t2p[of v 1r 1r], assumption+,
       simp add:Ring.ring-l-one, frule val-field-1-neq-0[of v],
       frule val-nonzero-z[of v 1r], assumption+,
       erule exE, simp add:a-zpz)
done

lemma (in Corps) has-val-one-neq-zero:valuation K v  $\implies$  1r  $\neq \mathbf{0}$ 
by (frule value-of-one[of v],
   rule contrapos-pp, simp+, simp add:value-of-zero)

lemma (in Corps) val-minus-one:valuation K v  $\implies$  v (-a 1r) = 0
apply (cut-tac field-is-ring, frule Ring.ring-one[of K],
       frule Ring.ring-is-ag[of K],
       frule val-field-1-neq-0[of v],
       frule aGroup.ag-inv-inj[of K 1r 0], assumption+,
       simp add:Ring.ring-zero, assumption)
apply (frule val-nonzero-z[of v -a 1r],
       rule aGroup.ag-mOp-closed, assumption+, simp add:aGroup.ag-inv-zero,
       erule exE, frule val-t2p [THEN sym, of v -a 1r -a 1r])
apply (simp add:aGroup.ag-mOp-closed[of K 1r],
       simp add:aGroup.ag-mOp-closed[of K 1r],
       frule Ring.ring-inv1-3 [THEN sym, of K 1r 1r], assumption+,
       simp add:Ring.ring-l-one, simp add:value-of-one a-zpz)
done

lemma (in Corps) val-minus-eq:[valuation K v; x  $\in$  carrier K]  $\implies$ 
v (-a x) = v x
apply (cut-tac field-is-ring,
       simp add:Ring.ring-times-minusl[of K x],
       subst val-t2p[of v], assumption+,
       frule Ring.ring-is-ag[of K], rule aGroup.ag-mOp-closed, assumption+,
       simp add:Ring.ring-one, assumption, simp add:val-minus-one,
       simp add:aadd-0-l)
done

lemma (in Corps) value-of-inv:[valuation K v; x  $\in$  carrier K; x  $\neq \mathbf{0}$ ]  $\implies$ 
v (x-K) = - (v x)
apply (cut-tac invf-inv[of x], erule conjE,
       frule val-t2p[of v x-K x], assumption+,
       simp+, simp add:value-of-one, simp add:a-inv)
apply simp
done

lemma (in Corps) val-exp-ring:[ valuation K v; x  $\in$  carrier K; x  $\neq \mathbf{0}$ ]

```

```

 $\implies (\text{int } n) *_a (v x) = v (x^{\text{K } n})$ 
apply (cut-tac field-is-ring,
         induct-tac n, simp add:Ring.ring-r-one, simp add:value-of-one)
apply (drule sym, simp)
apply (subst val-t2p[of v - x], assumption+,
         rule Ring.npClose, assumption+,
         frule val-nonzero-z[of v x], assumption+,
         erule exE, simp add:asprod-mult a-zpz,
         simp add: distrib-right)
done

exponent in a field

lemma (in Corps) val-exp: [valuation K v; x ∈ carrier K; x ≠ 0]  $\implies$ 
 $z *_a (v x) = v (x_K^z)$ 
apply (simp add:npowf-def)
apply (case-tac 0 ≤ z,
         simp, frule val-exp-ring [of v x nat z], assumption+,
         simp, simp)
apply (simp add:zle,
         cut-tac invf-closed1[of x], simp,
         cut-tac val-exp-ring [THEN sym, of v x⁻¹ nat (– z)], simp,
         thin-tac v (x⁻¹ K⁻¹ (nat (– z))) = (– z) *_a v (x⁻¹ K), erule conjE)
apply (subst value-of-inv[of v x], assumption+)
apply (frule val-nonzero-z[of v x], assumption+, erule exE, simp,
         simp add:asprod-mult aminus, simp+)
done

lemma (in Corps) value-zero-nonzero: [valuation K v; x ∈ carrier K; v x = 0]
 $\implies x \neq 0$ 
by (frule value-noninf-nonzero[of v x], assumption+, simp,
       assumption)

lemma (in Corps) v-ale-diff: [valuation K v; x ∈ carrier K; y ∈ carrier K;
x ≠ 0; v x ≤ v y]  $\implies 0 ≤ v(y ·_r x⁻¹ K)$ 
apply (frule value-in-aug-inf[of v x], simp+,
         frule value-in-aug-inf[of v y], simp+,
         frule val-nonzero-z[of v x], assumption+,
         erule exE)
apply (cut-tac invf-closed[of x], simp+,
         simp add:val-t2p,
         simp add:value-of-inv[of v x],
         frule-tac x = ant z in ale-diff-pos[of - v y],
         simp add:diff-ant-def)
apply simp
done

lemma (in Corps) amin-le-plusTr: [valuation K v; x ∈ carrier K; y ∈ carrier K;
v x ≠ ∞; v y ≠ ∞; v x ≤ v y]  $\implies \text{amin} (v x) (v y) \leq v (x ± y)$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag,

```

```

frule value-noninf-nonzero[of v x], assumption+,
frule v-ale-diff[of v x y], assumption+,
cut-tac invf-closed1[of x],
frule Ring.ring-tOp-closed[of K y x^- K], assumption+, simp,
frule Ring.ring-one[of K],
frule aGroup.ag-pOp-closed[of K 1_r y ._r x^- K], assumption+,
frule val-axiom4[of v y ._r ( x^- K)], assumption+
apply (frule aadd-le-mono[of 0 v (1_r ± y ._r x^- K) v x],
      simp add:aadd-0-l, simp add:aadd-commute[of - v x],
      simp add:val-t2p[THEN sym, of v x],
      simp add:Ring.ring-distrib1 Ring.ring-r-one,
      simp add:Ring.ring-tOp-commute[of K x],
      simp add:Ring.ring-tOp-assoc, simp add:linvf,
      simp add:Ring.ring-r-one,
      cut-tac amin-le-l[of v x v y],
      rule ale-trans[of amin (v x) (v y) v x v (x ± y)], assumption+)
apply simp
done

lemma (in Corps) amin-le-plus:[valuation K v; x ∈ carrier K; y ∈ carrier K]
      ⟹ (amin (v x) (v y)) ≤ (v (x ± y))
apply (cut-tac field-is-ring, frule Ring.ring-is-ag)
apply (case-tac v x = ∞ ∨ v y = ∞)
apply (erule disjE, simp,
       frule value-inf-zero[of v x], assumption+,
       simp add:aGroup.ag-l-zero amin-def,
       frule value-inf-zero[of v y], assumption+,
       simp add:aGroup.ag-r-zero amin-def, simp, erule conjE)
apply (cut-tac z = v x and w = v y in ale-linear,
       erule disjE, simp add:amin-le-plusTr,
       frule-tac amin-le-plusTr[of v y x], assumption+,
       simp add:aGroup.ag-pOp-commute amin-commute)
done

lemma (in Corps) value-less-eq:[ valuation K v; x ∈ carrier K; y ∈ carrier K;
      (v x) < (v y)] ⟹ (v x) = (v (x ± y))
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       frule amin-le-plus[of v x y], assumption+,
       frule aless-imp-le[of v x v y],
       simp add: amin-def)
apply (frule amin-le-plus[of v x ± y -_a y],
       rule aGroup.ag-pOp-closed, assumption+,
       rule aGroup.ag-mOp-closed, assumption+,
       simp add:val-minus-eq,
       frule aGroup.ag-mOp-closed[of K y], assumption+,
       simp add:aGroup.ag-pOp-assoc[of K x y],
       simp add:aGroup.ag-r-inv1, simp add:aGroup.ag-r-zero,
       simp add:amin-def)
apply (case-tac ¬ (v (x ±_K y) ≤ (v y)), simp+)

```

done

```

lemma (in Corps) value-less-eq1:valuation K v; x ∈ carrier K; y ∈ carrier K;
  (v x) < (v y) ] ==> v x = v (y ± x)
apply (cut-tac field-is-ring,
  frule Ring.ring-is-ag[of K],
  frule value-less-eq[of v x y], assumption+)
apply (subst aGroup.ag-pOp-commute, assumption+)
done

```

```

lemma (in Corps) val-1px:valuation K v; x ∈ carrier K; 0 ≤ (v (1_r ± x)) ]
  ==> 0 ≤ (v x)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
  frule Ring.ring-one[of K])
apply (rule contrapos-pp, simp+,
  case-tac x = 0_K,
  simp add:aGroup.ag-r-zero, simp add:value-of-zero,
  simp add: aneg-le[of 0 v x],
  frule value-less-eq[of v x 1_r], assumption+,
  simp add:value-of-one)
apply (drule sym,
  simp add:aGroup.ag-pOp-commute[of K x])
done

```

```

lemma (in Corps) val-1mx:valuation K v; x ∈ carrier K;
  0 ≤ (v (1_r ± (-_a x))) ] ==> 0 ≤ (v x)
by (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
  frule val-1px[of v -_a x],
  simp add:aGroup.ag-mOp-closed, assumption, simp add:val-minus-eq)

```

2.2 The normal valuation of v

definition

```

Lv :: [('r, 'm) Ring-scheme , 'r ⇒ ant] ⇒ ant where
Lv K v = AMin {x. x ∈ v ` carrier K ∧ 0 < x}

```

definition

```

n-val :: [('r, 'm) Ring-scheme, 'r ⇒ ant] ⇒ ('r ⇒ ant) where
n-val K v = (λx ∈ carrier K. (THE l. (l * (Lv K v)) = v x))

```

definition

```

Pg :: [('r, 'm) Ring-scheme, 'r ⇒ ant] ⇒ 'r where
Pg K v = (SOME x. x ∈ carrier K − {0_K} ∧ v x = Lv K v)

```

lemma (in Corps) vals-pos-nonempty:valuation K v ==>
 {x. x ∈ v ` carrier K ∧ 0 < x} ≠ {}

using val-axiom5[of v] value-noninf-nonzero[of v] value-of-inv[THEN sym, of v]
by (auto simp: ex-image-cong-iff) (metis Ring.ring-is-ag aGroup.ag-mOp-closed

```

aGroup.ag-pOp-closed aGroup.ag-r-inv1 f-is-ring zero-lt-inf)

lemma (in Corps) vals-pos-LBset:valuation K v ==>
  {x. x ∈ v ` carrier K ∧ 0 < x} ⊆ LBset 1
by (rule subsetI, simp add:LBset-def, erule conjE,
  rule-tac x = x in gt-a0-ge-1, assumption)

lemma (in Corps) Lv-pos:valuation K v ==> 0 < Lv K v
apply (simp add:Lv-def,
frule vals-pos-nonempty[of v],
frule vals-pos-LBset[of v],
simp only:ant-1[THEN sym],
frule AMin[of {x. x ∈ v ` carrier K ∧ 0 < x} 1], assumption+,
erule conjE)
apply simp
done

lemma (in Corps) AMin-z:valuation K v ==>
  ∃ a. AMin {x. x ∈ v ` carrier K ∧ 0 < x} = ant a
apply (frule vals-pos-nonempty[of v],
frule vals-pos-LBset[of v],
simp only:ant-1[THEN sym],
frule AMin[of {x. x ∈ v ` carrier K ∧ 0 < x} 1], assumption+,
erule conjE)
apply (frule val-axiom5[of v],
erule exE, (erule conjE)+,
cut-tac x = v x in aless-linear[of - 0], simp,
erule disjE,
frule-tac x = x in value-noninf-nonzero[of v], assumption+,
frule-tac x1 = x in value-of-inv[THEN sym, of v], assumption+)
apply (frule-tac x = v x in aless-minus[of - 0], simp,
cut-tac x = x in invf-closed1, simp, erule conjE,
frule valuation-map[of v],
frule-tac a = x^-K in mem-in-image[of v carrier K Z∞], simp)
apply (drule-tac a = v (x^-K) in forall-spec, simp,
frule-tac x = x^-K in val-nonzero-noninf[of v],
thin-tac v (x^-K) ∈ v ` carrier K,
thin-tac {x ∈ v ` carrier K. 0 < x} ⊆ LBset 1,
thin-tac AMin {x ∈ v ` carrier K. 0 < x} ∈ v ` carrier K,
thin-tac 0 < AMin {x ∈ v ` carrier K. 0 < x}, simp,
thin-tac v (x^-K) ∈ v ` carrier K,
thin-tac {x ∈ v ` carrier K. 0 < x} ⊆ LBset 1,
thin-tac AMin {x ∈ v ` carrier K. 0 < x} ∈ v ` carrier K,
thin-tac 0 < AMin {x ∈ v ` carrier K. 0 < x}, simp)
apply (rule noninf-mem-Z[of AMin {x ∈ v ` carrier K. 0 < x}],
frule image-sub[of v carrier K Z∞ carrier K],
rule subset-refl)
apply (rule subsetD[of v ` carrier K Z∞
  AMin {x ∈ v ` carrier K. 0 < x}], assumption+)

```

```

apply auto
by (metis (no-types, lifting) aneg-le aug-inf-noninf-is-z image-eqI value-in-aug-inf
z-less-i)

lemma (in Corps) Lv-z:valuation K v  $\implies \exists z. Lv K v = ant z$ 
by (simp add:Lv-def, rule AMin-z, assumption+)

lemma (in Corps) AMin-k:valuation K v  $\implies$ 
 $\exists k \in carrier K - \{\mathbf{0}\}. AMin \{x. x \in v \wedge carrier K \wedge 0 < x\} = v k$ 

apply (frule vals-pos-nonempty[of v],
frule vals-pos-LBset[of v],
simp only:ant-1[THEN sym],
frule AMin[of \{x. x \in v \wedge carrier K \wedge 0 < x\} 1], assumption+,
erule conjE)
apply (thin-tac  $\forall x \in \{x. x \in v \wedge carrier K \wedge 0 < x\}$ .
 $AMin \{x. x \in v \wedge carrier K \wedge 0 < x\} \leq x$ )
apply (simp add:image-def, erule conjE, erule bxE,
thin-tac  $\{x. (\exists xa \in carrier K. x = v xa) \wedge 0 < x\} \subseteq LBset 1$ ,
thin-tac  $\exists x. (\exists xa \in carrier K. x = v xa) \wedge 0 < x$ ,
subgoal-tac  $x \in carrier K - \{\mathbf{0}\}$ , blast,
frule AMin-z[of v], erule exE, simp)
apply (simp add:image-def,
thin-tac  $AMin \{x. (\exists xa \in carrier K. x = v xa) \wedge 0 < x\} = ant a$ ,
rule contrapos-pp, simp+, frule sym, thin-tac v (\mathbf{0}) = ant a,
simp add:value-of-zero)
done

lemma (in Corps) val-Pg: valuation K v  $\implies$ 
 $Pg K v \in carrier K - \{\mathbf{0}\} \wedge v (Pg K v) = Lv K v$ 
apply (frule AMin-k[of v], unfold Lv-def, unfold Pg-def)
apply (rule someI2-ex)
apply (erule bxE, drule sym, unfold Lv-def, blast)
apply simp
done

lemma (in Corps) amin-generateTr:valuation K v  $\implies$ 
 $\forall w \in carrier K - \{\mathbf{0}\}. \exists z. v w = z *_a AMin \{x. x \in v \wedge carrier K \wedge 0 < x\}$ 
apply (frule vals-pos-nonempty[of v],
frule vals-pos-LBset[of v],
simp only:ant-1[THEN sym],
frule AMin[of \{x. x \in v \wedge carrier K \wedge 0 < x\} 1], assumption+,
frule AMin-z[of v], erule exE, simp,
thin-tac  $\exists x. x \in v \wedge carrier K \wedge 0 < x$ ,
(erule conjE)+, rule ballI, simp, erule conjE,
frule-tac  $x = w$  in val-nonzero-noninf[of v], assumption+,
frule-tac  $x = w$  in value-in-aug-inf[of v], assumption+,
simp add:aug-inf-def,
cut-tac  $a = v w$  in mem-ant, simp, erule exE,

```

```

cut-tac  $a = z$  and  $b = a$  in amod-adiv-equality)
apply (case-tac  $z \bmod a = 0$ , simp add:ant-0 aadd-0-r, blast,
      thin-tac  $\{x. x \in v \text{ carrier } K \wedge 0 < x\} \subseteq LBset 1$ ,
      thin-tac  $v w \neq \infty$ , thin-tac  $v w \neq -\infty$ )

apply (frule AMin-k[of v], erule bxE,
       drule sym,
       drule sym,
       drule sym,
       rotate-tac -1, drule sym)

apply (cut-tac  $z = z$  in z-in-aug-inf,
       cut-tac  $z = (z \bmod a)$  and  $x = a$  in asp-z-Z,
       cut-tac  $z = z \bmod a$  in z-in-aug-inf,
       frule-tac  $a = ant z$  and  $b = (z \bmod a) *_a ant a$  and
        $c = ant (z \bmod a)$  in ant-sol, assumption+,
       subst asprod-mult, simp, assumption, simp,
       frule-tac  $x = k$  and  $z = z \bmod a$  in val-exp[of v],
       (erule conjE)+, assumption, simp, simp,
       thin-tac  $(z \bmod a) *_a v k = v (k_K^{(z \bmod a)})$ ,
       erule conjE)

apply (frule-tac  $x = k$  and  $n = z \bmod a$  in field-potent-nonzero1,
       assumption+,
       frule-tac  $a = k$  and  $n = z \bmod a$  in npowf-mem, assumption,
       frule-tac  $x1 = k_K^{(z \bmod a)}$  in value-of-inv[THEN sym, of v], assumption+,
       simp add:diff-ant-def,
       thin-tac  $-v (k_K^{(z \bmod a)}) = v ((k_K^{(z \bmod a)})^- K)$ ,
       cut-tac  $x = k_K^{(z \bmod a)}$  in invf-closed1, simp,
       simp, erule conjE,
       frule-tac  $x1 = w$  and  $y1 = (k_K^{(z \bmod a)})^- K$  in
       val-t2p[THEN sym, of v], assumption+, simp,
       cut-tac field-is-ring,
       thin-tac  $v w + v ((k_K^{(z \bmod a)})^- K) = ant (z \bmod a)$ ,
       thin-tac  $v (k_K^{(z \bmod a)}) + ant (z \bmod a) = v w$ ,
       frule-tac  $x = w$  and  $y = (k_K^{(z \bmod a)})^- K$  in
       Ring.ring-tOp-closed[of K], assumption+)

apply (frule valuation-map[of v],
       frule-tac  $a = w \cdot_r (k_K^{(z \bmod a)})^- K$  in mem-in-image[of v
       carrier K Z $_\infty$ ], assumption+, simp)

apply (thin-tac AMin  $\{x. x \in v \text{ carrier } K \wedge 0 < x\} = v k$ ,
       thin-tac  $v \in \text{carrier } K \rightarrow Z_\infty$ ,
       subgoal-tac  $0 < v (w \cdot_r (k_K^{(z \bmod a)})^- K)$ ,
       drule-tac  $a = v (w \cdot_r (k_K^{(z \bmod a)})^- K)$  in forall-spec,
       simp add:image-def)

apply (drule sym, simp)

apply (frule-tac  $b = a$  and  $a = z$  in Divides.pos-mod-conj, erule conjE,
       simp, simp,

```

```

frule-tac  $b = a$  and  $a = z$  in Divides.pos-mod-conj, erule conjE, simp)
done

lemma (in Corps) val-principalTr1:valuation K v]  $\implies$ 
  Lv K v  $\in$  v ‘(carrier K – {0})  $\wedge$ 
    ( $\forall w \in v$  ‘carrier K.  $\exists a. w = a * Lv K v$ )  $\wedge$  0  $<$  Lv K v
apply (rule conjI,
  frule val-Pg[of v], erule conjE,
  simp add:image-def, frule sym, thin-tac v (Pg K v) = Lv K v,
  erule conjE, blast)
apply (rule conjI,
  rule ballI, simp add:image-def, erule bxE)
apply (
  frule-tac  $x = x$  in value-in-aug-inf[of v], assumption,
  frule sym, thin-tac  $w = v x$ , simp add:aug-inf-def,
  cut-tac  $a = w$  in mem-ant, simp, erule disjE, erule exE,
  frule-tac  $x = x$  in value-noninf-nonzero[of v], assumption+,
  simp, frule amin-generateTr[of v])
apply (drule-tac  $x = x$  in bspec, simp,
  erule exE,
  frule AMin-z[of v], erule exE, simp add:Lv-def,
  simp add:asprod-mult, frule sym, thin-tac za * a = z,
  simp, subst a-z-z[THEN sym], blast)
apply (simp add:Lv-def,
  frule AMin-z[of v], erule exE, simp,
  frule Lv-pos[of v], simp add:Lv-def,
  frule-tac m1 = a in a-i-pos[THEN sym], blast,
  simp add:Lv-pos)
done

lemma (in Corps) val-principalTr2:valuation K v;
   $c \in v$  ‘(carrier K – {0})  $\wedge$  ( $\forall w \in v$  ‘carrier K.  $\exists a. w = a * c$ )  $\wedge$  0  $<$  c;
   $d \in v$  ‘(carrier K – {0})  $\wedge$  ( $\forall w \in v$  ‘carrier K.  $\exists a. w = a * d$ )  $\wedge$  0  $<$  d]
 $\implies c = d$ 
apply ((erule conjE)+,
  drule-tac  $x = d$  in bspec,
  simp add:image-def, erule bxE, blast,
  drule-tac  $x = c$  in bspec,
  simp add:image-def, erule bxE, blast)

apply ((erule exE)+,
  drule sym, simp,
  simp add:image-def, (erule bxE)+, simp,
  (erule conjE)+,
  frule-tac  $x = x$  in val-nonzero-z[of v], assumption+, erule exE,
  frule-tac  $x = xa$  in val-nonzero-z[of v], assumption+, erule exE,
  simp) apply (

```

```

subgoal-tac  $a \neq \infty \wedge a \neq -\infty$ , subgoal-tac  $aa \neq \infty \wedge aa \neq -\infty$ ,
cut-tac  $a = a$  in mem-ant, cut-tac  $aa = aa$  in mem-ant, simp,
(erule exE)+, simp add:a-z-z,
thin-tac  $c = ant z$ , frule sym, thin-tac  $zb * z = za$ , simp)
apply (subgoal-tac  $0 < zb$ ,
       cut-tac  $a = zc$  and  $b = zb$  in mult.commute, simp,
       simp add:pos-zmult-eq-1-iff,
       rule contrapos-pp, simp+,
       cut-tac  $x = 0$  and  $y = zb$  in less-linear, simp,
       thin-tac  $\neg 0 < zb$ ,
       erule disjE, simp,
       frule-tac  $i = 0$  and  $j = z$  and  $k = zb$  in zmult-zless-mono-neg,
       assumption+, simp add:mult.commute)
apply (rule contrapos-pp, simp+, thin-tac  $a \neq \infty \wedge a \neq -\infty$ ,
       erule disjE, simp, rotate-tac 5, drule sym,
       simp, simp, rotate-tac 5, drule sym, simp)
apply (rule contrapos-pp, simp+,
       erule disjE, simp, rotate-tac 4,
       drule sym, simp, simp,
       rotate-tac 4, drule sym,
       simp)
done

lemma (in Corps) val-principal:valuation K v ==>
   $\exists!x_0. x_0 \in v \wedge (\text{carrier } K - \{\mathbf{0}\}) \wedge$ 
   $(\forall w \in v \wedge (\text{carrier } K). \exists(a::ant). w = a * x_0) \wedge 0 < x_0$ 
by (rule ex-exII,
  frule val-principalTr1[of v], blast,
  rule-tac  $c = x_0$  and  $d = y$  in val-principalTr2[of v],
  assumption+)

lemma (in Corps) n-val-defTr:[valuation K v; w in carrier K] ==>
   $\exists!a. a * Lv K v = v w$ 
apply (rule ex-exII,
  frule AMin-k[of v],
  frule Lv-pos[of v], simp add:Lv-def,
  erule bxE,
  frule-tac  $x = k$  in val-nonzero-z[of v], simp, simp,
  erule exE, simp, (erule conjE)+)
apply (case-tac  $w = \mathbf{0}_K$ , simp add:value-of-zero,
  frule-tac  $m = z$  in a-i-pos, blast)
apply (frule amin-generateTr[of v],
  drule-tac  $x = w$  in bspec, simp, simp)
apply (
  erule exE, simp add:asprod-mult,
  subst a-z-z[THEN sym], blast)
apply (frule AMin-k[of v]) apply (erule bxE,
  frule Lv-pos[of v], simp add:Lv-def) apply (
  erule conjE,

```

```

frule-tac  $x = k$  in val-nonzero-z[of  $v$ ], assumption+,
erule exE, simp) apply (
case-tac  $w = \mathbf{0}_K$ , simp del:a-i-pos add:value-of-zero,
subgoal-tac  $y = \infty$ , simp, rule contrapos-pp, simp+,
cut-tac  $a = a$  in mem-ant, simp,
erule disjE, simp, erule exE, simp add:a-z-z)
apply (rule contrapos-pp, simp+,
cut-tac  $a = y$  in mem-ant, simp, erule disjE, simp,
erule exE, simp add:a-z-z,
frule-tac  $x = w$  in val-nonzero-z[of  $v$ ], assumption+,
erule exE, simp, cut-tac  $a = a$  in mem-ant,
erule disjE, simp, frule sym, thin-tac  $\infty = \text{ant } za$ , simp,
erule disjE, erule exE, simp add:a-z-z)
apply (cut-tac  $a = y$  in mem-ant,
erule disjE, simp, rotate-tac 3, drule sym,
simp, erule disjE, erule exE, simp add:a-z-z, frule sym,
thin-tac  $zb * z = za$ , simp, simp,
rotate-tac 3, drule sym,
simp, simp, frule sym, thin-tac  $\infty = \text{ant } za$ , simp)
done

lemma (in Corps) n-valTr:[valuation K v;  $x \in \text{carrier } K$ ]  $\implies$ 
(THE l. ( $l * (Lv K v)$ ) =  $v x$ )*( $Lv K v$ ) =  $v x$ 
by (rule theI', rule n-val-defTr, assumption+)

lemma (in Corps) n-val:[valuation K v;  $x \in \text{carrier } K$ ]  $\implies$ 
( $n\text{-val } K v x$ )*( $Lv K v$ ) =  $v x$ 
by (frule n-valTr[of  $v x$ ], assumption+, simp add:n-val-def)

lemma (in Corps) val-pos-n-val-pos:[valuation K v;  $x \in \text{carrier } K$ ]  $\implies$ 
( $0 \leq v x$ ) = ( $0 \leq n\text{-val } K v x$ )
apply (frule n-val[of  $v x$ ], assumption+,
drule sym,
frule Lv-pos[of  $v$ ],
frule Lv-z[of  $v$ ], erule exE, simp)
apply (frule-tac  $w = z$  and  $x = 0$  and  $y = n\text{-val } K v x$  in amult-pos-mono-r,
simp add:amult-0-l)
done

lemma (in Corps) n-val-in-aug-inf:[valuation K v;  $x \in \text{carrier } K$ ]  $\implies$ 
 $n\text{-val } K v x \in Z_\infty$ 
apply (cut-tac field-is-ring, frule Ring.ring-zero[of  $K$ ],
frule Lv-pos[of  $v$ ],
frule Lv-z[of  $v$ ], erule exE,
simp add:aug-inf-def)
apply (rule contrapos-pp, simp+)
apply (case-tac  $x = \mathbf{0}_K$ , simp,
frule n-val[of  $v \mathbf{0}$ ],
simp add:value-of-zero, simp add:value-of-zero)

```

```

apply (frule n-val[of v x], simp,
         frule val-nonzero-z[of v x], assumption+,
         erule exE, simp, rotate-tac -2, drule sym,
         simp)
done

lemma (in Corps) n-val-0:valuation K v; x ∈ carrier K; v x = 0] ⇒
n-val K v x = 0
by (frule Lv-z[of v], erule exE,
       frule Lv-pos[of v],
       frule n-val[of v x], simp, simp,
       rule-tac z = z and a = n-val K v x in a-a-z-0, assumption+)
lemma (in Corps) value-n0-n-val-n0:valuation K v; x ∈ carrier K; v x ≠ 0] ⇒
n-val K v x ≠ 0
apply (frule n-val[of v x],
         rule contrapos-pp, simp+, frule Lv-z[of v],
         erule exE, simp, simp only:ant-0[THEN sym])
apply (rule contrapos-pp, simp+,
         simp add:a-z-z)
done

lemma (in Corps) val-0-n-val-0:valuation K v; x ∈ carrier K] ⇒
(v x = 0) = (n-val K v x = 0)
apply (rule iffI,
         simp add:n-val-0)
apply (rule contrapos-pp, simp+,
         frule value-n0-n-val-n0[of v x], assumption+)
apply simp
done

lemma (in Corps) val-noninf-n-val-noninf:valuation K v; x ∈ carrier K] ⇒
(v x ≠ ∞) = (n-val K v x ≠ ∞)
by (frule Lv-z[of v], erule exE,
       frule Lv-pos[of v], simp,
       frule n-val[THEN sym, of v x], simp, simp,
       thin-tac v x = n-val K v x * ant z,
       rule iffI, rule contrapos-pp, simp+,
       cut-tac mem-ant[of n-val K v x], erule disjE, simp,
       erule disjE, erule exE, simp add:a-z-z, simp, simp)
lemma (in Corps) val-inf-n-val-inf:valuation K v; x ∈ carrier K] ⇒
(v x = ∞) = (n-val K v x = ∞)
by (cut-tac val-noninf-n-val-noninf[of v x], simp, assumption+)
lemma (in Corps) val-eq-n-val-eq:valuation K v; x ∈ carrier K; y ∈ carrier K]
⇒ (v x = v y) = (n-val K v x = n-val K v y)
apply (subst n-val[THEN sym, of v x], assumption+,

```

```

subst n-val[THEN sym, of v y], assumption+,
frule Lv-pos[of v], frule Lv-z[of v], erule exE, simp,
frule-tac s = z in zless-neq[THEN not-sym, of 0])
apply (rule iffI)
apply (rule-tac z = z in amult-eq-eq-r[of - n-val K v x n-val K v y],
assumption+)
apply simp
done

lemma (in Corps) val-poss-n-val-poss:[valuation K v; x in carrier K] ==>
(0 < v x) = (0 < n-val K v x)
apply (simp add:less-le,
frule val-pos-n-val-pos[of v x], assumption+,
rule iffI, erule conjE, simp,
simp add:value-n0-n-val-n0[of v x])
apply (drule sym,
erule conjE, simp,
frule-tac val-0-n-val-0[THEN sym, of v x], assumption+,
simp)
done

lemma (in Corps) n-val-Pg:valuation K v ==> n-val K v (Pg K v) = 1
apply (frule val-Pg[of v], simp, (erule conjE)+,
frule n-val[of v Pg K v], simp, frule Lv-z[of v], erule exE, simp,
frule Lv-pos[of v], simp, frule-tac i = 0 and j = z in zless-neq)
apply (rotate-tac -1, frule not-sym, thin-tac 0 ≠ z,
subgoal-tac n-val K v (Pg K v) * ant z = 1 * ant z,
rule-tac z = z in adiv-eq[of - n-val K v (Pg K v) 1], assumption+,
simp add:amult-one-l)
done

lemma (in Corps) n-val-valuationTr1:valuation K v ==>
∀ x∈carrier K. n-val K v x ∈ Z∞
by (rule ballI,
frule n-val[of v], assumption,
frule-tac x = x in value-in-aug-inf[of v], assumption,
frule Lv-pos[of v], simp add:aug-inf-def,
frule Lv-z[of v], erule exE, simp,
rule contrapos-pp, simp+)

lemma (in Corps) n-val-t2p:[valuation K v; x in carrier K; y in carrier K] ==>
n-val K v (x ·r y) = n-val K v x + (n-val K v y)
apply (cut-tac field-is-ring,
frule Ring.ring-tOp-closed[of K x y], assumption+,
frule n-val[of v x ·r y], assumption+,
frule Lv-pos[of v],
simp add:val-t2p,
frule n-val[THEN sym, of v x], assumption+,
frule n-val[THEN sym, of v y], assumption+, simp,

```

```

frule Lv-z[of v], erule exE, simp)
apply (subgoal-tac ant z ≠ 0)
apply (frule-tac z1 = z in amult-distrib1[THEN sym, of - n-val K v x
n-val K v y], simp,
thin-tac n-val K v x * ant z + n-val K v y * ant z =
(n-val K v x + n-val K v y) * ant z,
rule-tac z = z and a = n-val K v (x ·r y) and
b = n-val K v x + n-val K v y in adiv-eq, simp, assumption+, simp)
done

lemma (in Corps) n-val-valuationTr2:[ valuation K v; x ∈ carrier K;
y ∈ carrier K] ⇒
amin (n-val K v x) (n-val K v y) ≤ (n-val K v (x ± y))
apply (frule n-val[THEN sym, of v x], assumption+,
frule n-val[THEN sym, of v y], assumption+,
frule n-val[THEN sym, of v x ± y],
cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
rule aGroup.ag-pOp-closed, assumption+)
apply (frule amin-le-plus[of v x y], assumption+, simp,
simp add:amult-commute[of - Lv K v],
frule Lv-z[of v], erule exE, simp,
frule Lv-pos[of v], simp,
simp add:amin-amult-pos, simp add:amult-pos-mono-l)
done

lemma (in Corps) n-val-valuation:valuation K v ⇒
valuation K (n-val K v)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag)
apply (frule Lv-z[of v], erule exE, frule Lv-pos[of v], simp,
subst valuation-def)
apply (rule conjI, simp add:n-val-def restrict-def extensional-def)
apply (rule conjI, simp add:n-val-valuationTr1)
apply (rule conjI, frule n-val[of v 0],
simp add:Ring.ring-zero,
frule Lv-z[of v], erule exE, frule Lv-pos[of v],
cut-tac mem-ant[of n-val K v (0)], erule disjE,
simp add:value-of-zero,
erule disjE, erule exE, simp add:a-z-z value-of-zero, assumption+)
apply (rule conjI, rule ballI,
frule-tac x = x in val-nonzero-noninf[of v], simp+,
simp add:val-noninf-n-val-noninf)
apply (rule conjI, (rule ballI)+, simp add:n-val-t2p,
rule conjI, rule ballI, rule impI,
frule Lv-z[of v], erule exE,
frule Lv-pos[of v], simp,
frule-tac x = x in n-val[of v], simp,
frule-tac w1 = z and x1 = 0 and y1 = n-val K v x in
amult-pos-mono-r[THEN sym], simp add:amult-0-l,
frule-tac x = x in val-axiom4[of v], assumption+,

```

```

frule-tac  $x_1 = 1_r \pm x$  in  $n\text{-val}[THEN \text{sym}, \text{of } v]$ ,
frule  $\text{Ring}.ring\text{-is}\text{-ag}[\text{of } K]$ ,
    rule  $a\text{Group}.ag\text{-pOp-closed}$ , assumption+, simp add: $\text{Ring}.ring\text{-one}$ ,
          assumption,
frule-tac  $w = z \text{ and } x = 0 \text{ and } y = n\text{-val } K v (1_r \pm x)$ 
    in amult-pos-mono-r,
    simp add:amult-0-l)

apply (frule val-axiom5[of v], erule exE,
      (erule conjE)+,
      frule-tac  $x = x$  in value-n0-n-val-n0[of v], assumption+,
      frule-tac  $x = x$  in val-noninf-n-val-noninf, simp,
      blast)
done

lemma (in Corps)  $n\text{-val-le-val}:\llbracket \text{valuation } K v; x \in \text{carrier } K; 0 \leq (v x) \rrbracket \implies (n\text{-val } K v x) \leq (v x)$ 
by (subst n-val[THEN sym, of v x], assumption+,
     frule Lv-pos[of v],
     simp add:val-pos-n-val-pos[of v x],
     frule Lv-z[of v], erule exE,
     cut-tac  $b = z \text{ and } x = n\text{-val } K v x$  in amult-pos, simp+,
     simp add:asprod-amult, simp add:amult-commute)

lemma (in Corps)  $n\text{-val-surj}:\text{valuation } K v \implies \exists x \in \text{carrier } K. n\text{-val } K v x = 1$ 
apply (frule Lv-z[of v], erule exE,
      frule Lv-pos[of v],
      frule AMin-k[of v], erule bxE, frule-tac  $x = k$  in n-val[of v], simp,
      simp add:Lv-def)
apply (subgoal-tac  $n\text{-val } K v k * \text{ant } z = 1 * \text{ant } z$ ,
       subgoal-tac  $z \neq 0$ ,
       frule-tac  $z = z \text{ and } a = n\text{-val } K v k \text{ and } b = 1$  in amult-eq-eq-r,
       assumption, blast, simp, simp add:amult-one-l)
done

lemma (in Corps)  $n\text{-value-in-aug-inf}:\llbracket \text{valuation } K v; x \in \text{carrier } K \rrbracket \implies n\text{-val } K v x \in Z_\infty$ 
by (frule n-val[of v x], assumption,
     simp add:aug-inf-def, rule contrapos-pp, simp+,
     frule Lv-pos[of v], frule Lv-z[of v], erule exE, simp,
     frule value-in-aug-inf[of v x], assumption+, simp add:aug-inf-def)

lemma (in Corps)  $\text{val-surj-n-valTr}:\llbracket \text{valuation } K v; \exists x \in \text{carrier } K. v x = 1 \rrbracket \implies Lv K v = 1$ 
apply (erule bxE,
      frule-tac  $x = x$  in n-val[of v],
      simp, frule Lv-pos[of v])
apply (frule-tac  $w = Lv K v \text{ and } x = n\text{-val } K v x$  in amult-1-both)

```

```

apply simp+
done

lemma (in Corps) val-surj-n-val:[valuation K v; ∃ x ∈ carrier K. v x = 1] ⇒
      (n-val K v) = v
apply (rule funcset-eq[of - carrier K],
      simp add:n-val-def restrict-def extensional-def,
      simp add:valuation-def)
apply (rule ballI,
      frule val-surj-n-valTr[of v], assumption+,
      frule-tac x = x in n-val[of v], assumption+,
      simp add:amult-one-r)
done

lemma (in Corps) n-val-n-val:valuation K v ⇒
      n-val K (n-val K v) = n-val K v
by (frule n-val-valuation[of v],
     frule n-val-surj[of v],
     simp add:val-surj-n-val)

lemma nnonzero-annzero:0 < N ⇒ an N ≠ 0
apply (simp only:an-0[THEN sym])
apply (subst aneq-natneq, simp)
done

```

2.3 Valuation ring

definition

```

Vr :: [('r, 'm) Ring-scheme, 'r ⇒ ant] ⇒ ('r, 'm) Ring-scheme where
  Vr K v = Sr K (⟨x. x ∈ carrier K ∧ 0 ≤ (v x)⟩)

```

definition

```

vp :: [('r, 'm) Ring-scheme, 'r ⇒ ant] ⇒ 'r set where
  vp K v = {x. x ∈ carrier (Vr K v) ∧ 0 < (v x)}

```

definition

```

r-apow :: [('r, 'm) Ring-scheme, 'r set, ant] ⇒ 'r set where
  r-apow R I a = (if a = ∞ then {0_R} else
                    (if a = 0 then carrier R else I^{R (na a)}))

```

abbreviation

```

RAPOW ((β- -) [62,62,63]62) where
  I^R a == r-apow R I a

```

```

lemma (in Ring) ring-pow-apow:ideal R I ⇒
      I^{R n} = I^R (an n)
apply (simp add:r-apow-def)

```

```

apply (case-tac n = 0, simp)
apply (simp add:n nonzero-annonzero)
apply (simp add:an-neq-inf na-an)
done

lemma (in Ring) r-apow-Suc:ideal R I ==> I^R (an (Suc 0)) = I
apply (cut-tac an-1, simp add:r-apow-def)
apply (simp add:a0-neq-1[THEN not-sym])
apply (simp only:ant-1[THEN sym])
apply (simp del:ant-1 add:z-neq-inf[of 1, THEN not-sym])
apply (simp add:na-1)
apply (simp add:idealprod-whole-r)
done

lemma (in Ring) apow-ring-pow:ideal R I ==>
I^{<R} n = I^R (an n)
apply (simp add:r-apow-def)
apply (case-tac n = 0, simp add:an-0)
apply (simp add: aless-nat-less[THEN sym],
       cut-tac an-neq-inf[of n],
       simp add: less-le[of 0 an n] na-an)
done

lemma (in Corps) Vr-ring:valuation K v ==> Ring (Vr K v)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       simp add:Vr-def, rule Ring.Sr-ring, assumption+)
apply (simp add:sr-def)
apply (intro conjI subsetI)
apply (simp-all add: value-of-one Ring.ring-one[of K])
apply ((rule allI, rule impI)+,
       (erule conjE)+, rule conjI, rule aGroup.ag-pOp-closed, assumption+,
       rule aGroup.ag-mOp-closed, assumption+)
apply (frule-tac x = x and y = -_a y in amin-le-plus[of v], assumption+,
       rule aGroup.ag-mOp-closed, assumption+,
       simp add:val-minus-eq[of v]) apply (
       frule-tac z = 0 and x = v x and y = v y in amin-ge1, assumption+,
       frule-tac i = 0 and j = amin (v x) (v y) and k = v (x ± -_a y) in
       ale-trans, assumption+, simp)
by (simp add: Ring.ring-tOp-closed aadd-two-pos val-t2p)

lemma (in Corps) val-pos-mem-Vr:[valuation K v; x ∈ carrier K] ==>
(0 ≤ (v x)) = (x ∈ carrier (Vr K v))
by (rule iffI, (simp add:Vr-def Sr-def)+)

lemma (in Corps) val-poss-mem-Vr:[valuation K v; x ∈ carrier K; 0 < (v x)] ==>
x ∈ carrier (Vr K v)
by (frule aless-imp-le[of 0 v x], simp add:val-pos-mem-Vr)

lemma (in Corps) Vr-one:valuation K v ==> 1_r K ∈ carrier (Vr K v)

```

```

by (cut-tac field-is-ring, frule Ring.ring-one[of K],
           frule val-pos-mem-Vr[of v 1_r], assumption+,
           simp add:value-of-one)

lemma (in Corps) Vr-mem-f-mem:[valuation K v; x ∈ carrier (Vr K v)]
    ⟹ x ∈ carrier K
by (simp add:Vr-def Sr-def)

lemma (in Corps) Vr-0-f-0:valuation K v ⟹ 0_Vr K v = 0
by (simp add:Vr-def Sr-def)

lemma (in Corps) Vr-1-f-1:valuation K v ⟹ 1_r(Vr K v) = 1_r
by (simp add:Vr-def Sr-def)

lemma (in Corps) Vr-pOp-f-pOp:[valuation K v; x ∈ carrier (Vr K v);
    y ∈ carrier (Vr K v)] ⟹ x ±_{Vr K v} y = x ± y
by (simp add:Vr-def Sr-def)

lemma (in Corps) Vr-mOp-f-mOp:[valuation K v; x ∈ carrier (Vr K v)]
    ⟹ -_a(Vr K v) x = -_a x
by (simp add:Vr-def Sr-def)

lemma (in Corps) Vr-tOp-f-tOp:[valuation K v; x ∈ carrier (Vr K v);
    y ∈ carrier (Vr K v)] ⟹ x ·_r (Vr K v) y = x ·_r y
by (simp add:Vr-def Sr-def)

lemma (in Corps) Vr-pOp-le:[valuation K v; x ∈ carrier K;
    y ∈ carrier (Vr K v)] ⟹ v x ≤ (v x + (v y))
apply (frule val-pos-mem-Vr[THEN sym, of v y],
       simp add:Vr-mem-f-mem, simp, frule aadd-pos-le[of v y v x],
       simp add:aadd-commute)
done

lemma (in Corps) Vr-integral:valuation K v ⟹ Idomain (Vr K v)
apply (simp add:Idomain-def,
       simp add:Vr-ring, simp add:Idomain-axioms-def,
       rule allI, rule impI, rule allI, (rule impI)+,
       simp add:Vr-tOp-f-tOp, simp add:Vr-0-f-0)
apply (rule contrapos-pp, simp+, erule conjE,
       cut-tac field-is-idom,
       frule-tac x = a in Vr-mem-f-mem[of v], assumption,
       frule-tac x = b in Vr-mem-f-mem[of v], assumption,
       frule-tac x = a and y = b in Idomain.idom-tOp-nonzeros[of K],
       assumption+, simp)
done

lemma (in Corps) Vr-exp-mem:[valuation K v; x ∈ carrier (Vr K v)]
    ⟹ x^K n ∈ carrier (Vr K v)
by (frule Vr-ring[of v],

```

$\text{induct-tac } n, \text{ simp add: Vr-one,}$
 $\text{simp add: Vr-tOp-f-tOp[THEN sym, of v],}$
 $\text{simp add: Ring.ring-tOp-closed)}$

lemma (in Corps) $\text{Vr-exp-f-exp} : \llbracket \text{valuation } K v; x \in \text{carrier} (\text{Vr } K v) \rrbracket \implies$
 $x^{\wedge}(\text{Vr } K v)^n = x^{\wedge}K^n$

apply ($\text{induct-tac } n,$
 $\text{simp, simp add: Vr-1-f-1, simp,}$
 $\text{thin-tac } x^{\wedge}(\text{Vr } K v)^n = x^{\wedge}K^n)$

apply ($\text{rule Vr-tOp-f-tOp, assumption+},$
 $\text{simp add: Vr-exp-mem, assumption})$

done

lemma (in Corps) $\text{Vr-potent-nonzero} : \llbracket \text{valuation } K v;$
 $x \in \text{carrier} (\text{Vr } K v) - \{\mathbf{0}_{\text{Vr } K v}\} \rrbracket \implies x^{\wedge}K^n \neq \mathbf{0}_{\text{Vr } K v}$

apply ($\text{frule Vr-mem-f-mem[of v x], simp,}$
 $\text{simp add: Vr-0-f-0, erule conjE})$

apply ($\text{frule Vr-mem-f-mem[of v x], assumption+},$
 $\text{simp add: field-potent-nonzero})$

done

lemma (in Corps) $\text{elem-0-val-if} : \llbracket \text{valuation } K v; x \in \text{carrier } K; v x = 0 \rrbracket$
 $\implies x \in \text{carrier} (\text{Vr } K v) \wedge x^{-K} \in \text{carrier} (\text{Vr } K v)$

apply ($\text{frule val-pos-mem-Vr[of v x], assumption, simp})$

apply ($\text{frule value-zero-nonzero[of v x], simp add: Vr-mem-f-mem, simp})$

apply ($\text{frule value-of-inv[of v x], assumption+},$
 $\text{simp, subst val-pos-mem-Vr[THEN sym, of v } x^{-K}] \text{, assumption+},$
 $\text{cut-tac invf-closed[of x], simp+})$

done

lemma (in Corps) $\text{elem0val} : \llbracket \text{valuation } K v; x \in \text{carrier } K; x \neq \mathbf{0} \rrbracket \implies$
 $(v x = 0) = (x \in \text{carrier} (\text{Vr } K v) \wedge x^{-K} \in \text{carrier} (\text{Vr } K v))$

apply ($\text{rule iffI, rule elem-0-val-if[of v], assumption+,}$
 $\text{erule conjE})$

apply ($\text{simp add: val-pos-mem-Vr[THEN sym, of v x],}$
 $\text{frule Vr-mem-f-mem[of v } x^{-K}], \text{ assumption+},$
 $\text{simp add: val-pos-mem-Vr[THEN sym, of v } x^{-K}],$
 $\text{simp add: value-of-inv, frule ale-minus[of } 0 - v x],$
 $\text{simp add: a-minus-minus})$

done

lemma (in Corps) $\text{ideal-inc-elem0val-whole} : \llbracket \text{valuation } K v; x \in \text{carrier } K;$
 $v x = 0; \text{ideal } (\text{Vr } K v) I; x \in I \rrbracket \implies I = \text{carrier} (\text{Vr } K v)$

apply ($\text{frule elem-0-val-if[of v x], assumption+, erule conjE,}$
 $\text{frule value-zero-nonzero[of v x], assumption+},$
 $\text{frule Vr-ring[of v],}$
 $\text{frule-tac } I = I \text{ and } x = x \text{ and } r = x^{-K} \text{ in}$
 $\text{Ring.ideal-ring-multiple[of Vr } K v], \text{ assumption+},$

```

cut-tac invf-closed1[of x], simp+, (erule conjE)+
apply (simp add:Vr-tOp-f-tOp, cut-tac invf-inv[of x], simp+
      simp add: Vr-1-f-1[THEN sym, of v],
      simp add:Ring.ideal-inc-one, simp+
done

lemma (in Corps) vp-mem-Vr-mem: $\llbracket \text{valuation } K v; x \in (\text{vp } K v) \rrbracket \implies$ 
 $x \in \text{carrier } (\text{Vr } K v)$ 
by (rule val-poss-mem-Vr[of v x], assumption+, (simp add:vp-def
 $\text{Vr-def Sr-def})+)$ 

lemma (in Corps) vp-mem-val-poss: $\llbracket \text{valuation } K v; x \in \text{carrier } K \rrbracket \implies$ 
 $(x \in \text{vp } K v) = (0 < (v x))$ 
by (simp add:vp-def, simp add:Vr-def Sr-def less-ant-def)

lemma (in Corps) Pg-in-Vr:valuation K v  $\implies$   $\text{Pg } K v \in \text{carrier } (\text{Vr } K v)$ 
by (frule val-Pg[of v], erule conjE,
 $\text{frule Lv-pos[of v], drule sym,}$ 
 $\text{simp, erule conjE,}$ 
 $\text{simp add:val-poss-mem-Vr})$ 

lemma (in Corps) vp-ideal:valuation K v  $\implies$   $\text{ideal } (\text{Vr } K v) (\text{vp } K v)$ 
apply (cut-tac field-is-ring,
 $\text{frule Vr-ring[of v],}$ 
 $\text{rule Ring.ideal-condition1, assumption+,}$ 
 $\text{rule subsetI, simp add:vp-mem-Vr-mem,}$ 
 $\text{simp add:vp-def})$ 
apply (frule val-Pg[of v],
 $\text{frule Lv-pos[of v], simp, (erule conjE)+,}$ 
 $\text{drule sym, simp,}$ 
 $\text{frule val-poss-mem-Vr[of v Pg K v], assumption+, blast})$ 

apply ((rule ballI)+,
 $\text{frule-tac } x = x \text{ in vp-mem-Vr-mem[of v], assumption) apply (}$ 
 $\text{frule-tac } x = y \text{ in vp-mem-Vr-mem[of v], assumption,}$ 
 $\text{simp add:vp-def,}$ 
 $\text{frule Ring.ring-is-ag[of Vr } K v\text{],}$ 
 $\text{frule-tac } x = x \text{ and } y = y \text{ in aGroup.ag-pOp-closed, assumption+, simp})$ 
apply (simp add:Vr-pOp-f-pOp,
 $\text{cut-tac } x = v x \text{ and } y = v y \text{ in amin-le-l,}$ 
 $\text{frule-tac } x = x \text{ and } y = y \text{ in amin-le-plus,}$ 
 $\text{(simp add:Vr-mem-f-mem)+,}$ 
 $\text{(frule-tac } z = 0 \text{ and } x = v x \text{ and } y = v y \text{ in amin-gt, assumption+),}$ 
 $\text{rule-tac } x = 0 \text{ and } y = amin (v x) (v y) \text{ and } z = v (x \pm y) \text{ in}$ 
 $\text{less-le-trans, assumption+)}$ 
apply ((rule ballI)+,
 $\text{frule-tac } x1 = r \text{ in val-poss-mem-Vr[THEN sym, of v],}$ 
 $\text{simp add:Vr-mem-f-mem, simp,}$ 
 $\text{frule-tac } x = x \text{ in vp-mem-Vr-mem[of v], simp add:Vr-pOp-f-pOp,}$ 

```

```

simp add:vp-def, simp add:Ring.ring-tOp-closed,
simp add:Vr-tOp-f-tOp)
apply (frule-tac x = r in Vr-mem-f-mem[of v], assumption+,
      frule-tac x = x in Vr-mem-f-mem[of v], assumption+,
      simp add:val-t2p, simp add:aadd-pos-poss)
done

lemma (in Corps) vp-not-whole:valuation K v ==>
  (vp K v) ≠ carrier (Vr K v)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
      frule Vr-ring[of v])
apply (rule contrapos-pp, simp+,
      drule sym,
      frule Ring.ring-one[of Vr K v], simp,
      simp add:Vr-1-f-1,
      frule Ring.ring-one[of K])
apply (simp only:vp-mem-val-poss[of v 1_r],
      simp add:value-of-one)
done

lemma (in Ring) elem-out-ideal-nonzero:[ideal R I; x ∈ carrier R;
  x ∉ I] ==> x ≠ 0_R
by (rule contrapos-pp, simp+, frule ideal-zero[of I],
    simp)

lemma (in Corps) vp-prime:valuation K v ==> prime-ideal (Vr K v) (vp K v)
apply (simp add:prime-ideal-def, simp add:vp-ideal)
apply (rule conjI)

apply (rule contrapos-pp, simp+,
      frule Vr-ring[of v],
      frule vp-ideal[of v],
      frule Ring.ideal-inc-one[of Vr K v vp K v], assumption+,
      simp add:vp-not-whole[of v])

apply ((rule ballI)+, rule impI, rule contrapos-pp, simp+, (erule conjE)+,
      frule Vr-ring[of v]) apply (
      frule-tac x = x in Vr-mem-f-mem[of v], assumption) apply (
      frule-tac x = y in Vr-mem-f-mem[of v], assumption) apply (
      frule vp-ideal[of v],
      frule-tac x = x in Ring.elem-out-ideal-nonzero[of Vr K v vp K v],
      assumption+) apply (
      frule-tac x = y in Ring.elem-out-ideal-nonzero[of Vr K v vp K v],
      assumption+, simp add:Vr-0-f-0,
      simp add:Vr-tOp-f-tOp) apply (
      frule-tac x = x ·_r y in vp-mem-val-poss[of v],
      cut-tac field-is-ring, simp add:Ring.ring-tOp-closed, simp)
apply (cut-tac field-is-ring,

```

```

frule-tac  $x = x$  and  $y = y$  in Ring.ring-tOp-closed, assumption+,
simp add:Ring.ring-tOp-closed[of  $Vr K v$ ],
simp add:vp-def, simp add:aneg-less,
frule-tac  $x_1 = x$  in val-pos-mem- $Vr$ [THEN sym, of  $v$ ], assumption+,
frule-tac  $x_1 = y$  in val-pos-mem- $Vr$ [THEN sym, of  $v$ ], assumption+,
frule-tac  $P = x \in carrier (Vr K v)$  and  $Q = 0 \leq v x$  in eq-prop,
assumption,
frule-tac  $P = y \in carrier (Vr K v)$  and  $Q = 0 \leq v y$  in eq-prop,
assumption,
frule-tac  $x = v x$  and  $y = 0$  in ale-antisym, assumption+,
frule-tac  $x = v y$  and  $y = 0$  in ale-antisym, assumption+,
simp add:val-t2p aadd-0-l)
done

lemma (in Corps) vp-pow-ideal:valuation K v  $\Rightarrow$ 
ideal ( $Vr K v$ )  $((vp K v)^{\diamondsuit} (Vr K v)^n)$ 
by (frule Vr-ring[of  $v$ ], frule vp-ideal[of  $v$ ],
simp add:Ring.ideal-pow-ideal)

lemma (in Corps) vp-apow-ideal:[valuation K v; 0 ≤ n] ⇒
ideal ( $Vr K v$ )  $((vp K v)^{(Vr K v)^n})$ 
apply (frule Vr-ring[of  $v$ ])
apply (case-tac  $n = 0$ ,
simp add:r-apow-def, simp add:Ring.whole-ideal[of  $Vr K v$ ])
apply (case-tac  $n = \infty$ ,
simp add:r-apow-def, simp add:Ring.zero-ideal)
apply (simp add:r-apow-def, simp add:vp-pow-ideal)
done

lemma (in Corps) mem-vp-apow-mem-Vr:[valuation K v;
 $0 \leq N; x \in vp K v (Vr K v)^N]$   $\Rightarrow x \in carrier (Vr K v)$ 
by (frule Vr-ring[of  $v$ ], frule vp-apow-ideal[of  $v N$ ], assumption,
simp add:Ring.ideal-subset)

lemma (in Corps) elem-out-vp-unit:[valuation K v; x ∈ carrier (Vr K v);
 $x \notin vp K v] \Rightarrow v x = 0$ 
by (metis Vr-mem-f-mem ale-antisym aneg-le val-pos-mem- $Vr$  vp-mem-val-poss)

lemma (in Corps) vp-maximal:valuation K v  $\Rightarrow$ 
maximal-ideal ( $Vr K v$ )  $(vp K v)$ 
apply (frule Vr-ring[of  $v$ ],
simp add:maximal-ideal-def, simp add:vp-ideal)

apply (frule vp-not-whole[of  $v$ ],
rule conjI, rule contrapos-pp, simp+, frule vp-ideal[of  $v$ ],
frule Ring.ideal-inc-one[of  $Vr K v vp K v$ ], assumption+)
apply simp

apply (rule equalityI,

```

```

rule subsetI, simp, erule conjE,
case-tac x = vp K v, simp, simp, rename-tac X)

apply (frule-tac A = X in sets-not-eq[of - vp K v], assumption+,
erule bxE,
frule-tac I = X and h = a in Ring.ideal-subset[of Vr K v],
assumption+,
frule-tac x = a in elem-out-vp-unit[of v], assumption+)

apply (frule-tac x = a and I = X in ideal-inc-elem0val-whole [of v],
simp add: Vr-mem-f-mem, assumption+)

apply (rule subsetI, simp, erule disjE,
simp add:prime-ideal-def, simp add:vp-ideal,
simp add:Ring.whole-ideal, rule subsetI, simp add:vp-mem-Vr-mem)
done

lemma (in Corps) ideal-sub-vp:[ valuation K v; ideal (Vr K v) I;
I ≠ carrier (Vr K v)] ⇒ I ⊆ (vp K v)
apply (frule Vr-ring[of v], rule contrapos-pp, simp+)
apply (simp add:subset-eq,
erule bxE)
apply (frule-tac h = x in Ring.ideal-subset[of Vr K v I], assumption+,
frule-tac x = x in elem-out-vp-unit[of v], assumption+,
frule-tac x = x in ideal-inc-elem0val-whole[of v - I],
simp add: Vr-mem-f-mem, assumption+, simp)
done

lemma (in Corps) Vr-local:[valuation K v; maximal-ideal (Vr K v) I] ⇒
(vp K v) = I
apply (frule Vr-ring[of v],
frule ideal-sub-vp[of v I], simp add:Ring.maximal-ideal-ideal)
apply (simp add:maximal-ideal-def,
frule conjunct2, fold maximal-ideal-def, frule conjunct1,
rule Ring.proper-ideal, assumption+,simp add:maximal-ideal-def, assumption)
apply (rule equalityI) prefer 2 apply assumption
apply (rule contrapos-pp, simp+,
frule sets-not-eq[of vp K v I], assumption+, erule bxE)
apply (frule-tac x = a in vp-mem-Vr-mem[of v],
frule Ring.maximal-ideal-ideal[of Vr K v I], assumption,
frule-tac x = a in Ring.elem-out-ideal-nonzero[of Vr K v I],
assumption+,
frule vp-ideal[of v], rule Ring.ideal-subset[of Vr K v vp K v],
assumption+)

apply (frule-tac a = a in Ring.principal-ideal[of Vr K v], assumption+,
frule Ring.maximal-ideal-ideal[of Vr K v I], assumption+,
frule-tac ?I2.0 = Vr K v ◇_p ain Ring.sum-ideals[of Vr K v I],
simp add:Ring.maximal-ideal-ideal, assumption,

```

```

frule-tac ?I2.0 = Vr K v  $\diamond_p$  ain Ring.sum-ideals-la1[of Vr K v I],
  assumption+,
frule-tac ?I2.0 = Vr K v  $\diamond_p$  ain Ring.sum-ideals-la2[of Vr K v I],
  assumption+,
frule-tac a = a in Ring.a-in-principal[of Vr K v], assumption+,
frule-tac A = Vr K v  $\diamond_p$  a and B = I  $\mp$ (Vr K v) (Vr K v  $\diamond_p$  a)
  and c = a in subsetD, assumption+
thm Ring.sum-ideals-cont[of Vr K v vp K v I ]
apply (frule-tac B = Vr K v  $\diamond_p$  a in Ring.sum-ideals-cont[of Vr K v
  vp K v I], simp add:vp-ideal, assumption)
apply (frule-tac a = a in Ring.ideal-cont-Rxa[of Vr K v vp K v],
  simp add:vp-ideal, assumption+)
apply (simp add:maximal-ideal-def, erule conjE)+,
subgoal-tac I  $\mp$ (Vr K v) (Vr K v  $\diamond_p$  a)  $\in$  {J. ideal (Vr K v) J  $\wedge$  I  $\subseteq$  J},
  simp, thin-tac {J. ideal (Vr K v) J  $\wedge$  I  $\subseteq$  J} = {I, carrier (Vr K v)}
apply (erule disjE, simp)
apply (cut-tac A = carrier (Vr K v) and B = I  $\mp$ Vr K v Vr K v  $\diamond_p$  a and
  C = vp K v in subset-trans, simp, assumption,
  frule Ring.ideal-subset1[of Vr K v vp K v], simp add:vp-ideal,
  frule equalityI[of vp K v carrier (Vr K v)], assumption+,
  frule vp-not-whole[of v], simp)
apply blast
done

lemma (in Corps) v-residue-field:valuation K v ==>
  Corps ((Vr K v) /r (vp K v))
by (frule Vr-ring[of v],
  rule Ring.residue-field-cd [of Vr K v vp K v], assumption+,
  simp add:vp-maximal)

lemma (in Corps) Vr-n-val-Vr:valuation K v ==>
  carrier (Vr K v) = carrier (Vr K (n-val K v))
by (simp add:Vr-def Sr-def,
  rule equalityI,
  (rule subsetI, simp, erule conjE, simp add:val-pos-n-val-pos),
  (rule subsetI, simp, erule conjE, simp add:val-pos-n-val-pos[THEN sym]))

```

2.4 Ideals in a valuation ring

```

lemma (in Corps) Vr-has-poss-elem:valuation K v ==>
   $\exists x \in \text{carrier} (\text{Vr } K \text{ v}) - \{\mathbf{0}_{\text{Vr } K \text{ v}}\}. 0 < v x$ 
apply (frule val-Pg[of v], erule conjE,
  frule Lv-pos[of v], drule sym,
  subst Vr-0-f-0, assumption+)
apply (frule aeq-ale[of Lv K v v (Pg K v)],
  frule aless-le-trans[of 0 Lv K v v (Pg K v)], assumption+,
  frule val-poss-mem-Vr[of v Pg K v],
  simp, assumption, blast)

```

done

lemma (in Corps) vp-nonzero:valuation K v \implies vp K v $\neq \{\mathbf{0}_{Vr K v}\}$
apply (frule Vr-has-poss-elem[of v], erule bxE,
 simp, erule conjE,
 frule-tac x1 = x in vp-mem-val-poss[THEN sym, of v],
 simp add:Vr-mem-f-mem, simp, rule contrapos-pp, simp+)

done

**lemma (in Corps) field-frac-mul:[x in carrier K; y in carrier K; y $\neq \mathbf{0}$]
 $\implies x = (x \cdot_r (y^K)) \cdot_r y$**
apply (cut-tac invf-closed[of y],
 cut-tac field-is-ring,
 simp add:Ring.ring-tOp-assoc,
 subst linuf[of y], simp, simp add:Ring.ring-r-one[of K], simp)

done

**lemma (in Corps) elems-le-val:[valuation K v; x in carrier K; y in carrier K;
 $x \neq \mathbf{0}; v x \leq (v y)$] $\implies \exists r \in \text{carrier} (\text{Vr } K v). y = r \cdot_r x$**
apply (frule ale-diff-pos[of v x v y], simp add:diff-ant-def,
 simp add:value-of-inv[THEN sym, of v x],
 cut-tac invf-closed[of x],
 simp only:val-t2p[THEN sym, of v y x^K])

**apply (cut-tac field-is-ring,
frule-tac x = y and y = x^K in Ring.ring-tOp-closed[of K],
assumption+,
simp add:val-pos-mem-Vr[of v y \cdot_r (x^K)],
frule field-frac-mul[of y x], assumption+, blast)**

apply simp

done

**lemma (in Corps) val-Rxa-gt-a:[valuation K v; x in carrier (\text{Vr } K v) - \{\mathbf{0}\};
y in carrier (\text{Vr } K v); y in Rxa (\text{Vr } K v) x] $\implies v x \leq (v y)$**
apply (simp add:Rxa-def,

erule bxE,
 simp add:Vr-tOp-f-tOp, (erule conjE)+,
 frule-tac x = r in Vr-mem-f-mem[of v], assumption+,
 frule-tac x = x in Vr-mem-f-mem[of v], assumption+)

**apply (subst val-t2p, assumption+,
simp add:val-pos-mem-Vr[THEN sym, of v],
frule-tac y = v r in aadd-le-mono[of 0 - v x],
simp add:aadd-0-l)**

done

**lemma (in Corps) val-Rxa-gt-a-1:[valuation K v; x in carrier (\text{Vr } K v);
y in carrier (\text{Vr } K v); x $\neq \mathbf{0}; v x \leq (v y)$] $\implies y \in Rxa (\text{Vr } K v) x$**
apply (frule-tac x = x in Vr-mem-f-mem[of v], assumption+,

frule-tac x = y in Vr-mem-f-mem[of v], assumption+,
 frule v-ale-diff[of v x y], assumption+)

```

cut-tac invf-closed[of  $x$ ],
cut-tac field-is-ring, frule Ring.ring-tOp-closed[of  $K$   $y$   $x^K$ ],
assumption+)
apply (simp add:val-pos-mem-Vr[of  $v$   $y \cdot_r (x^K)$ ],
frule field-frac-mul[of  $y$   $x$ ], assumption+,
simp add:Rxa-def, simp add:Vr-tOp-f-tOp, blast, simp)
done

lemma (in Corps) eval-inv:[[valuation  $K$   $v$ ;  $x \in carrier K$ ;  $y \in carrier K$ ;
 $y \neq \mathbf{0}$ ;  $v x = v y$ ]  $\implies$   $0 = v (x \cdot_r (y^K))$ ]
by (cut-tac invf-closed[of  $y$ ],
simp add:val-t2p value-of-inv, simp add:aadd-minus-r,
simp)
lemma (in Corps) eq-val-eq-idealTr:[[valuation  $K$   $v$ ;
 $x \in carrier (Vr K v) - \{\mathbf{0}\}$ ;  $y \in carrier (Vr K v)$ ;  $v x \leq (v y)$ ]  $\implies$ 
 $Rxa (Vr K v) y \subseteq Rxa (Vr K v) x$ ]
apply (frule val-Rxa-gt-a-1[of  $v$   $x$   $y$ ], simp+,
erule conjE)
apply (frule-tac  $x = x$  in Vr-mem-f-mem[of  $v$ ], assumption+,
frule Vr-ring[of  $v$ ],
frule Ring.principal-ideal[of  $Vr K v x$ ], assumption,
frule Ring.ideal-cont-Rxa[of  $Vr K v (Vr K v) \diamond_p x y$ ],
assumption+)
done

lemma (in Corps) eq-val-eq-ideal:[[valuation  $K$   $v$ ;
 $x \in carrier (Vr K v)$ ;  $y \in carrier (Vr K v)$ ;  $v x = v y$ ]  $\implies$ 
 $Rxa (Vr K v) x = Rxa (Vr K v) y$ ]
apply (case-tac  $x = \mathbf{0}_K$ ,
simp add:value-of-zero,
frule value-inf-zero[of  $v$   $y$ ],
simp add:Vr-mem-f-mem, rule sym, assumption, simp)
apply (rule equalityI,
rule eq-val-eq-idealTr[of  $v$   $y$   $x$ ], assumption+,
drule sym, simp,
rule contrapos-pp, simp+, simp add:value-of-zero,
frule Vr-mem-f-mem[of  $v$   $x$ ], assumption+,
frule value-inf-zero[of  $v$   $x$ ], assumption+,
rule sym, assumption, simp, simp, simp)
apply (rule eq-val-eq-idealTr[of  $v$   $x$   $y$ ], assumption+, simp,
assumption, rule aeq-ale, assumption+)
done

lemma (in Corps) eq-ideal-eq-val:[[valuation  $K$   $v$ ;  $x \in carrier (Vr K v)$ ;
 $y \in carrier (Vr K v)$ ;  $Rxa (Vr K v) x = Rxa (Vr K v) y$ ]  $\implies$   $v x = v y$ ]
apply (case-tac  $x = \mathbf{0}_K$ , simp,
drule sym,
frule Vr-ring[of  $v$ ],

```

```

frule Ring.a-in-principal[of Vr K v y], assumption+, simp,
thin-tac Vr K v  $\diamond_p$  y = Vr K v  $\diamond_p$  (0), simp add:Rxa-def,
erule bxE, simp add:Vr-0-f-0[of v, THEN sym])
apply (simp add:Vr-tOp-f-tOp, simp add:Vr-0-f-0,
frule-tac x = r in Vr-mem-f-mem[of v], assumption+,
cut-tac field-is-ring, simp add:Ring.ring-times-x-0)
apply (frule Vr-ring[of v],
frule val-Rxa-gt-a[of v x y], simp,
simp)
apply (drule sym,
frule Ring.a-in-principal[of Vr K v y], simp, simp)
apply (frule val-Rxa-gt-a[of v y x],
simp, rule contrapos-pp, simp+,
frule Ring.a-in-principal[of Vr K v x], assumption+,
simp add:Rxa-def,
erule bxE, simp add:Vr-tOp-f-tOp, cut-tac field-is-ring,
frule-tac x = r in Vr-mem-f-mem[of v], assumption+,
simp add:Ring.ring-times-x-0, simp,
frule Ring.a-in-principal[of Vr K v x], assumption+, simp,
rule ale-antisym, assumption+)
done

lemma (in Corps) zero-val-gen-whole:
 $\llbracket \text{valuation } K v; x \in \text{carrier} (\text{Vr } K v) \rrbracket \implies$ 
 $(v x = 0) = (\text{Rxa } (\text{Vr } K v) x = \text{carrier } (\text{Vr } K v))$ 
apply (frule Vr-mem-f-mem[of v x], assumption,
frule Vr-ring[of v])
apply (rule iffI,
frule Ring.principal-ideal[of Vr K v x], assumption+,
frule Ring.a-in-principal[of Vr K v x], assumption+,
rule ideal-inc-elem0val-whole[of v x Vr K v  $\diamond_p$  x], assumption+,
frule Ring.ring-one[of Vr K v],
frule eq-set-inc[of 1_r(Vr K v)
carrier (Vr K v) Vr K v  $\diamond_p$  x], drule sym, assumption,
thin-tac 1_r(Vr K v)  $\in$  carrier (Vr K v),
thin-tac Vr K v  $\diamond_p$  x = carrier (Vr K v))
apply (simp add:Rxa-def, erule bxE,
simp add:Vr-1-f-1, simp add:Vr-tOp-f-tOp,
frule value-of-one[of v], simp,
frule-tac x = r in Vr-mem-f-mem[of v], assumption+,
cut-tac field-is-ring, simp add:val-t2p,
simp add:val-pos-mem-Vr[THEN sym, of v],
rule contrapos-pp, simp+,
cut-tac less-le[THEN sym, of 0 v x], drule not-sym, simp,
frule-tac x = v r and y = v x in aadd-pos-poss, assumption+,
simp)
done

lemma (in Corps) elem-nonzeroval-gen-proper:  $\llbracket \text{valuation } K v;$ 

```

```

 $x \in \text{carrier} (\text{Vr } K v); v x \neq 0] \implies Rxa (\text{Vr } K v) x \neq \text{carrier} (\text{Vr } K v)$ 
apply (rule contrapos-pp, simp+)
apply (simp add: zero-val-gen-whole[THEN sym])
done

```

We prove that $\text{Vr } K v$ is a principal ideal ring

definition

```

 $LI :: [('r, 'm) \text{ Ring-scheme}, 'r \Rightarrow \text{ant}, 'r \text{ set}] \Rightarrow \text{ant where}$ 

```

```

 $LI K v I = AMin (v ` I)$ 

```

definition

```

 $Ig :: [('r, 'm) \text{ Ring-scheme}, 'r \Rightarrow \text{ant}, 'r \text{ set}] \Rightarrow 'r \text{ where}$ 

```

```

 $Ig K v I = (\text{SOME } x. x \in I \wedge v x = LI K v I)$ 

```

```

lemma (in Corps) val-in-image:  $[\text{valuation } K v; \text{ideal } (\text{Vr } K v) I; x \in I] \implies$ 
 $v x \in v ` I$ 
by (simp add:image-def, blast)

```

```

lemma (in Corps) I-vals-nonempty:  $[\text{valuation } K v; \text{ideal } (\text{Vr } K v) I] \implies$ 
 $v ` I \neq \{\}$ 
by (frule Vr-ring[of v], frule Ring.ideal-zero[of Vr K v I], assumption+, rule contrapos-pp, simp+)

```

```

lemma (in Corps) I-vals-LBset:  $[\text{valuation } K v; \text{ideal } (\text{Vr } K v) I] \implies$ 
 $v ` I \subseteq LBset 0$ 
apply (frule Vr-ring[of v], rule subsetI, simp add:LBset-def, simp add:image-def)
apply (erule bxE, frule-tac h = xa in Ring.ideal-subset[of Vr K v I], assumption+)
apply (frule-tac x1 = xa in val-pos-mem-Vr[THEN sym, of v], simp add:Vr-mem-f-mem, simp)
done

```

```

lemma (in Corps) LI-pos:  $[\text{valuation } K v; \text{ideal } (\text{Vr } K v) I] \implies 0 \leq LI K v I$ 
apply (simp add:LI-def, frule I-vals-LBset[of v], simp add:ant-0[THEN sym], frule I-vals-nonempty[of v], simp only:ant-0)

```

```

apply (simp only:ant-0[THEN sym], frule AMin[of v ` I 0], assumption, erule conjE, frule subsetD[of v ` I LBset (ant 0) AMin (v ` I)], assumption+, simp add:LBset-def)
done

```

```

lemma (in Corps) LI-poss:  $[\text{valuation } K v; \text{ideal } (\text{Vr } K v) I;$ 
 $I \neq \text{carrier } (\text{Vr } K v)] \implies 0 < LI K v I$ 

```

```

apply (subst less-le)
apply (simp add:LI-pos)
apply (rule contrapos-pp, simp+

apply (simp add:LI-def,
      frule I-vals-LBset[of v], assumption+,
      simp add:ant-0[THEN sym],
      frule I-vals-nonempty[of v], assumption+, simp only:ant-0)

apply (simp only:ant-0[THEN sym], frule AMin[of v ` I 0], assumption,
      erule conjE, frule subsetD[of v ` I LBset (ant 0) AMin (v ` I)],
      assumption+, simp add:LBset-def)

apply (thin-tac  $\forall x \in I. \text{ant } 0 \leq v x$ ,
      thin-tac  $v ` I \subseteq \{x. \text{ant } 0 \leq x\}$ , simp add:image-def,
      erule bxE, simp add:ant-0)
apply (frule Vr-ring[of v],
      frule-tac  $h = x$  in Ring.ideal-subset[of Vr K v I], assumption+,
      frule-tac  $x = x$  in zero-val-gen-whole[of v], assumption+,
      simp,
      frule-tac  $a = x$  in Ring.ideal-cont-Rxa[of Vr K v I], assumption+,
      simp, frule Ring.ideal-subset1[of Vr K v I], assumption+)
apply (frule equalityI[of I carrier (Vr K v)], assumption+, simp)
done

lemma (in Corps) LI-z: [valuation K v; ideal (Vr K v) I;  $I \neq \{\mathbf{0}_{Vr K v}\}] \implies \exists z. LI K v I = \text{ant } z$ 
apply (frule Vr-ring[of v],
      frule Ring.ideal-zero[of Vr K v I], assumption+,
      cut-tac mem-ant[of LI K v I],
      frule LI-pos[of v I], assumption,
      erule disjE, simp,
      cut-tac minf-le-any[of 0],
      frule ale-antisym[of 0  $\rightarrow \infty$ ], assumption+, simp)
apply (erule disjE, simp,
      frule singleton-sub[of  $\mathbf{0}_{Vr K v} I$ ],
      frule sets-not-eq[of I  $\{\mathbf{0}_{Vr K v}\}], assumption+,$ 
      erule bxE, simp)

apply (simp add:LI-def,
      frule I-vals-LBset[of v], assumption+,
      simp only:ant-0[THEN sym],
      frule I-vals-nonempty[of v], assumption+,
      frule AMin[of v ` I 0], assumption, erule conjE)
apply (frule-tac  $x = a$  in val-in-image[of v I], assumption+,
      drule-tac  $x = v$  a in bspec, simp,
      simp add:Vr-0-f-0,
      frule-tac  $x = a$  in val nonzero-z[of v],
      simp add:Ring.ideal-subset Vr-mem-f-mem, assumption+,

```

```

erule exE, simp,
cut-tac x = ant z in inf-ge-any, frule-tac x = ant z in
ale-antisym[of - ∞], assumption+, simp)
done

lemma (in Corps) LI-k:[valuation K v; ideal (Vr K v) I] ==>
  ∃ k ∈ I. LI K v I = v k
by (simp add:LI-def,
  frule I-vals-LBset[of v], assumption+,
  simp only:ant-0[THEN sym],
  frule I-vals-nonempty[of v], assumption+,
  frule AMin[of v ` I 0], assumption, erule conjE,
  thin-tac ∀ x ∈ v ` I. AMin (v ` I) ≤ x, simp add:image-def)

lemma (in Corps) LI-infinity:[valuation K v; ideal (Vr K v) I] ==>
  (LI K v I = ∞) = (I = {0_Vr K v})
apply (frule Vr-ring[of v])
apply (rule iffI)
apply (rule contrapos-pp, simp+,
  frule Ring.ideal-zero[of Vr K v I], assumption+,
  frule singleton-sub[of 0_Vr K v I],
  frule sets-not-eq[of I {0_Vr K v}], assumption+,
  erule bxE,
  frule-tac h = a in Ring.ideal-subset[of Vr K v I], assumption+,
  simp add:Vr-0-f-0,
  frule-tac x = a in Vr-mem-f-mem[of v], assumption+,
  frule-tac x = a in val nonzero-z[of v], assumption+,
  erule exE,
  simp add:LI-def,
  frule I-vals-LBset[of v], assumption+,
  simp only:ant-0[THEN sym],
  frule I-vals-nonempty[of v], assumption+,
  frule AMin[of v ` I 0], assumption, erule conjE)
apply (frule-tac h = a in Ring.ideal-subset[of Vr K v I], assumption+,
  frule-tac x = a in val-in-image[of v I], assumption+,
  drule-tac x = v a in bspec, simp)
apply (frule-tac x = a in val nonzero-z[of v], assumption+,
  erule exE, simp,
  cut-tac x = ant z in inf-ge-any, frule-tac x = ant z in
  ale-antisym[of - ∞], assumption+, simp)

apply (frule sym, thin-tac I = {0_Vr K v},
  simp add:LI-def,
  frule I-vals-LBset[of v], assumption+,
  simp only:ant-0[THEN sym],
  frule I-vals-nonempty[of v], assumption+,
  frule AMin[of v ` I 0], assumption, erule conjE,
  drule sym, simp,
  simp add:Vr-0-f-0 value-of-zero)

```

done

```

lemma (in Corps) val-Ig:[[valuation K v; ideal (Vr K v) I] ==>
  (Ig K v I) ∈ I ∧ v (Ig K v I) = LI K v I
by (simp add:Ig-def, rule someI2-ex,
  frule LI-k[of v I], assumption+, erule bxE,
  drule sym, blast, assumption)

lemma (in Corps) Ig-nonzero:[[valuation K v; ideal (Vr K v) I; I ≠ {0 Vr K v}] ==
  (Ig K v I) ≠ 0
by (rule contrapos-p, simp+,
  frule LI-infinity[of v I], assumption+,
  frule val-Ig[of v I], assumption+, erule conjE,
  simp add:value-of-zero)

lemma (in Corps) Vr-ideal-npowf-closed:[[valuation K v; ideal (Vr K v) I;
  x ∈ I; 0 < n] ==> x_K^n ∈ I
by (simp add:npowf-def, frule Vr-ring[of v],
  frule Ring.ideal-npow-closed[of Vr K v I x nat n], assumption+,
  simp, frule Ring.ideal-subset[of Vr K v I x], assumption+,
  simp add:Vr-exp-f-exp)

lemma (in Corps) Ig-generate-I:[[valuation K v; ideal (Vr K v) I] ==>
  (Vr K v) ◇_p (Ig K v I) = I
apply (frule Vr-ring[of v])
apply (case-tac I = carrier (Vr K v),
  frule sym, thin-tac I = carrier (Vr K v),
  frule Ring.ring-one[of Vr K v],
  simp, simp add: Vr-1-f-1,
  frule val-Ig[of v I], assumption+, erule conjE,
  frule LI-pos[of v I], assumption+,
  simp add: LI-def cong del: image-cong-simp,
  frule I-vals-LBset[of v], assumption+,
  simp only: ant-0[THEN sym],
  frule I-vals-nonempty[of v], assumption+,
  frule AMin[of v ` I 0], assumption, erule conjE,
  frule val-in-image[of v I 1_r], assumption+,
  drule-tac x = v (1_r) in bspec, assumption+,
  simp add: value-of-one ant-0 cong del: image-cong-simp,
  simp add: zero-val-gen-whole[of v Ig K v I])

apply (frule val-Ig[of v I], assumption+, (erule conjE)+,
  frule Ring.ideal-cont-Rxa[of Vr K v I Ig K v I], assumption+,
  rule equalityI, assumption+)

apply (case-tac LI K v I = ∞,
  frule LI-infinity[of v I], simp,

```

```

simp add:Rxa-def, simp add:Ring.ring-times-x-0,
frule Ring.ring-zero, blast)

apply (rule subsetI,
       case-tac v x = 0,
       frule-tac x = x in Vr-mem-f-mem[of v],
       simp add:Ring.ideal-subset,
       frule-tac x = x in zero-val-gen-whole[of v],
       simp add:Ring.ideal-subset, simp,
       frule-tac a = x in Ring.ideal-cont-Rxa[of Vr K v I], assumption+,
       simp, frule Ring.ideal-subset1[of Vr K v I], assumption,
       frule equalityI[of I carrier (Vr K v)], assumption+, simp)
apply (simp add:LI-def,
       frule I-vals-LBset[of v], assumption+,
       simp only:ant-0[THEN sym],
       frule I-vals-nonempty[of v], assumption+,
       frule AMin[of v ` I 0], assumption, erule conjE,
       frule-tac x = v x in bspec,
       frule-tac x = x in val-in-image[of v I], assumption+,
       simp)
apply (drule-tac x = x in bspec, assumption,
       frule-tac y = x in eq-val-eq-idealTr[of v Ig K v I],
       simp add:Ring.ideal-subset,
       rule contrapos-pp, simp+, simp add:value-of-zero,
       simp add:Ring.ideal-subset, simp)

apply (frule-tac a = x in Ring.a-in-principal[of Vr K v],
       simp add:Ring.ideal-subset, rule subsetD, assumption+)
done

lemma (in Corps) Pg-gen-vp:valuation K v ==>
          (Vr K v) ⋄_p (Pg K v) = vp K v
apply (frule vp-ideal[of v],
       frule Ig-generate-I[of v vp K v], assumption+,
       frule vp-not-whole[of v],
       frule eq-val-eq-ideal[of v Ig K v (vp K v) Pg K v],
       frule val-Ig [of v vp K v], assumption+, erule conjE,
       simp add:vp-mem-Vr-mem)

apply (frule val-Pg[of v], erule conjE,
       frule Lv-pos[of v],
       rotate-tac -2, drule sym, simp,
       simp add:val-poss-mem-Vr)
apply (thin-tac Vr K v ⋄_p Ig K v (vp K v) = vp K v,
       frule val-Pg[of v], erule conjE,
       simp, frule val-Ig[of v vp K v], assumption+, erule conjE,
       simp, thin-tac v (Pg K v) = Lv K v,
       thin-tac Ig K v (vp K v) ∈ vp K v ∧ v (Ig K v (vp K v)) =

```

```

LI K v (vp K v), simp add:LI-def Lv-def,
subgoal-tac v ` vp K v = {x. x ∈ v ` carrier K ∧ 0 < x},
simp)

apply (thin-tac ideal (Vr K v) (vp K v), thin-tac Pg K v ∈ carrier K,
thin-tac Pg K v ≠ 0,
rule equalityI, rule subsetI,
simp add:image-def vp-def, erule exE, erule conjE,
(erule conjE)+,
frule-tac x = xa in Vr-mem-f-mem[of v], assumption+, simp, blast)

apply (rule subsetI, simp add:image-def vp-def, erule conjE, erule bxE, simp,
frule-tac x = xa in val-poss-mem-Vr[of v], assumption+,
cut-tac y = v xa in less-le[0], simp, blast, simp)
done

lemma (in Corps) vp-gen-t:valuation K v ==>
  ∃ t∈carrier (Vr K v). vp K v = (Vr K v) ◇p t
by (frule Pg-gen-vp[of v], frule Pg-in-Vr[of v], blast)

lemma (in Corps) vp-gen-nonzero:[valuation K v; vp K v = (Vr K v) ◇p t] ==>
  t ≠ 0_Vr K v
apply (rule contrapos-pp, simp+,
cut-tac Ring.Rxa-zero[of Vr K v], drule sym, simp,
simp add:vp-nonzero)
apply (simp add:Vr-ring)
done

lemma (in Corps) n-value-idealTr:[valuation K v; 0 ≤ n] ==>
  (vp K v) ◇(Vr K v) n = Vr K v ◇p ((Pg K v) ▷(Vr K v) n)
apply (frule Vr-ring[of v],
frule Pg-gen-vp[THEN sym, of v],
simp add:vp-ideal,
frule val-Pg[of v], simp, (erule conjE)+)
apply (subst Ring.principal-ideal-n-pow[of Vr K v Pg K v
Vr K v ◇p Pg K v], assumption+,
frule Lv-pos[of v], rotate-tac -2, frule sym,
thin-tac v (Pg K v) = Lv K v, simp, simp add:val-poss-mem-Vr,
simp+)
done

lemma (in Corps) ideal-pow-vp:[valuation K v; ideal (Vr K v) I;
I ≠ carrier (Vr K v); I ≠ {0_Vr K v}] ==>
  I = (vp K v) ◇(Vr K v) (na (n-val K v (Ig K v I)))
apply (frule Vr-ring[of v],
frule Ig-generate-I[of v I], assumption+)

apply (frule n-val[of v Ig K v I],
frule val-Ig[of v I], assumption+, erule conjE,

```

```

simp add:Ring.ideal-subset[of Vr K v I Ig K v I] Vr-mem-f-mem)

apply (frule val-Pg[of v], erule conjE,
      rotate-tac -1, drule sym, simp,
      frule Ig-nonzero[of v I], assumption+,
      frule LI-pos[of v I], assumption+,
      frule Lv-pos[of v],
      frule val-Ig[of v I], assumption+, (erule conjE)+,
      rotate-tac -1, drule sym, simp,
      frule val-pos-n-val-pos[of v Ig K v I],
      simp add:Ring.ideal-subset Vr-mem-f-mem,
      simp)
apply (frule zero-val-gen-whole[THEN sym, of v Ig K v I],
      simp add:Ring.ideal-subset,
      simp, rotate-tac -1, drule not-sym,
      cut-tac less-le[THEN sym, of 0 v (Ig K v I)], simp,
      thin-tac 0 ≤ v (Ig K v I),
      frule Ring.ideal-subset[of Vr K v I Ig K v I], assumption+,
      frule Vr-mem-f-mem[of v Ig K v I], assumption+,
      frule val-poss-n-val-poss[of v Ig K v I], assumption+, simp)
apply (frule Ig-nonzero[of v I],
      frule val-nonzero-noninf[of v Ig K v I], assumption+,
      simp add:val-noninf-n-val-noninf[of v Ig K v I],
      frule val-poss-mem-Vr[of v Pg K v], assumption+,
      subst n-value-idealTr[of v na (n-val K v (Ig K v I))],
      assumption+, simp add:na-def)

apply (frule eq-val-eq-ideal[of v Ig K v I
                           (Pg K v) ∼(Vr K v) (na (n-val K v (Ig K v I)))], assumption+,
      rule Ring.npClose, assumption+,
      simp add:Vr-exp-f-exp[of v Pg K v],
      subst val-exp-ring[THEN sym, of v Pg K v
                           na (n-val K v (Ig K v I))], assumption+)
apply (frule Lv-z[of v], erule exE, simp,
      rotate-tac 6, drule sym, simp,
      subst asprod-amult,
      simp add:val-poss-n-val-poss[of v Ig K v I],
      frule val-nonzero-noninf[of v Ig K v I], assumption+,
      frule val-noninf-n-val-noninf[of v Ig K v I], assumption+, simp,
      rule aposs-na-poss[of n-val K v (Ig K v I)], assumption+)

apply (fold an-def)
apply (subst an-na[THEN sym, of n-val K v (Ig K v I)],
      frule val-nonzero-noninf[of v Ig K v I], assumption+,
      frule val-noninf-n-val-noninf[of v Ig K v I], assumption+, simp,
      simp add:aless-imp-le, simp)
apply simp
done

lemma (in Corps) ideal-apow-vp:[valuation K v; ideal (Vr K v) I] ==>

```

```

 $I = (vp K v) (Vr K v) (n-val K v (Ig K v I))$ 
apply (frule Vr-ring[of v])
apply (case-tac v (Ig K v I) =  $\infty$ ,
        frule val-Ig[of v I], assumption,
        frule val-inf-n-val-inf[of v Ig K v I],
        simp add:Ring.ideal-subset Vr-mem-f-mem, simp, simp add:r-apow-def,
        simp add:LI-infinity[of v I])

apply (case-tac v (Ig K v I) = 0,
        frule val-0-n-val-0[of v Ig K v I],
        frule val-Ig[of v I], assumption+, erule conjE,
        simp add:Ring.ideal-subset Vr-mem-f-mem, simp,
        frule val-Ig[of v I], assumption,
        frule zero-val-gen-whole[of v Ig K v I],
        simp add:Ring.ideal-subset, (erule conjE)+, simp,
        frule Ring.ideal-cont-Rxa[of Vr K v I Ig K v I], assumption+)

apply (simp,
        frule Ring.ideal-subset1[of Vr K v I], assumption+,
        frule equalityI[of I carrier (Vr K v)], assumption+,
        simp add:r-apow-def)

apply (frule val-noninf-n-val-noninf[of v Ig K v I],
        frule val-Ig[of v I], assumption,
        simp add:Ring.ideal-subset Vr-mem-f-mem, simp,
        frule value-n0-n-val-n0[of v Ig K v I],
        frule val-Ig[of v I], assumption,
        simp add:Ring.ideal-subset Vr-mem-f-mem, assumption)

apply (simp add:r-apow-def,
        rule ideal-pow-vp, assumption+,
        frule elem-nonzeroval-gen-proper[of v Ig K v I],
        frule val-Ig[of v I], assumption+, erule conjE,
        simp add:Ring.ideal-subset, assumption, simp add:Ig-generate-I)

apply (frule val-Ig[of v I], assumption+, erule conjE, simp,
        simp add:LI-infinity[of v I])
done

```

```

lemma (in Corps) ideal-apow-n-val:valuation K v; x ∈ carrier (Vr K v)]]  $\Rightarrow$ 
 $(Vr K v) \diamond_p x = (vp K v)^{(Vr K v) (n-val K v x)}$ 
apply (frule Vr-ring[of v],
        frule Ring.principal-ideal[of Vr K v x], assumption+,
        frule ideal-apow-vp[of v Vr K v \diamond_p x], assumption+)
apply (frule val-Ig[of v Vr K v \diamond_p x], assumption+, erule conjE,
        frule Ring.ideal-subset[of Vr K v Vr K v \diamond_p x
        Ig K v (Vr K v \diamond_p x)], assumption+,
        frule Ig-generate-I[of v Vr K v \diamond_p x], assumption+)

```

```

apply (frule eq-ideal-eq-val[of v Ig K v (Vr K v  $\diamond_p$  x) x],
       assumption+,
       thin-tac Vr K v  $\diamond_p$  Ig K v (Vr K v  $\diamond_p$  x) = Vr K v  $\diamond_p$  x,
       thin-tac v (Ig K v (Vr K v  $\diamond_p$  x)) = LI K v (Vr K v  $\diamond_p$  x),
       frule n-val[THEN sym, of v x],
       simp add:Vr-mem-f-mem, simp,
       thin-tac v x = n-val K v x * Lv K v,
       frule n-val[THEN sym, of v Ig K v (Vr K v  $\diamond_p$  x)],
       simp add:Vr-mem-f-mem, simp,
       thin-tac v (Ig K v (Vr K v  $\diamond_p$  x)) = n-val K v x * Lv K v)
apply (frule Lv-pos[of v],
       frule Lv-z[of v], erule exE, simp,
       frule-tac s = z in zless-neq[THEN not-sym, of 0],
       frule-tac z = z in adiv-eq[of - n-val K v (Ig K v (Vr K v  $\diamond_p$  x))
                                   n-val K v x], assumption+, simp)
done

```

lemma (in Corps) t-gen-vp: $\llbracket \text{valuation } K v; t \in \text{carrier } K; v t = 1 \rrbracket \implies (Vr K v) \diamond_p t = vp K v$

proof –

assume a1: $\text{valuation } K v$ **and**
 a2: $t \in \text{carrier } K$ **and**
 a3: $v t = 1$

from a1 **and** a2 **and** a3 **have** h1: $t \in \text{carrier } (Vr K v)$
 apply (cut-tac a0-less-1)
 apply (rule val-poss-mem-Vr[of v t], assumption+, simp) **done**

from a1 **and** a2 **and** a3 **have** h2: $n\text{-val } K v = v$
 apply (subst val-surj-n-val[of v]) apply assumption
 apply blast apply simp **done**

from a1 **and** h1 **have** h3: $Vr K v \diamond_p t = vp K v (Vr K v) (n\text{-val } K v t)$
 apply (simp add:ideal-apow-n-val[of v t]) **done**

from a1 **and** a3 **and** h2 **and** h3 **show** ?thesis
 apply (simp add:r-apow-def)
 apply (simp only:ant-1[THEN sym], simp only:ant-0[THEN sym])
 apply (simp only:aeq-zeq, simp)
 apply (cut-tac z-neq-inf[THEN not-sym, of 1], simp)
 apply (simp only:an-1[THEN sym]) apply (simp add:na-an)
 apply (rule Ring.idealprod-whole-r[of Vr K v vp K v])
 apply (simp add:Vr-ring)
 apply (simp add:vp-ideal) **done**

qed

lemma (in Corps) t-vp-apow: $\llbracket \text{valuation } K v; t \in \text{carrier } K; v t = 1 \rrbracket \implies (Vr K v) \diamond_p (t^{\neg(Vr K v)} n) = (vp K v) (an n)$

proof –

```

assume a1:valuation K v and
    a2:t ∈ carrier K and
    a3:v t = 1
from a1 have h1:Ring (Vr K v) by (simp add:Vr-ring[of v])
from a1 and a2 and a3 have h2:t ∈ carrier (Vr K v)
    apply (cut-tac a0-less-1)
    apply (rule val-poss-mem-Vr) apply assumption+ apply simp done
from a1 and a2 and a3 and h1 and h2 show ?thesis
    apply (subst Ring.principal-ideal-n-pow[THEN sym, of Vr K v t vp K v n])
    apply assumption+
    apply (simp add:t-gen-vp)
    apply (simp add:r-apow-def)
    apply (rule conjI, rule impI,
        simp only:an-0[THEN sym], frule an-inj[of n 0], simp)
    apply (rule impI)
    apply (rule conjI, rule impI)
    apply (simp add:an-def)
    apply (rule impI, cut-tac an-nat-pos[of n], simp add:na-an)
done
qed

lemma (in Corps) nonzeroelem-gen-nonzero:valuation K v; x ≠ 0;
    x ∈ carrier (Vr K v)  $\implies$  Vr K v  $\diamond_p$  x ≠ {0}  $V_r K v$ 
by (frule Vr-ring[of v],
    frule-tac a = x in Ring.a-in-principal[of Vr K v], assumption+,
    rule contrapos-pp, simp+, simp add:Vr-0-f-0)

```

2.4.1 Amin lemma (in Corps)s

```

lemma (in Corps) Amin-le-addTr:valuation K v  $\implies$ 
    ( $\forall j \leq n. f j \in \text{carrier } K$ )  $\longrightarrow$  Amin n (v ∘ f)  $\leq$  (v (nsum K f n))
    apply (induct-tac n)
    apply (rule impI, simp)

    apply (rule impI,
        simp,
        frule-tac x =  $\Sigma_e K f n$  and y = f (Suc n) in amin-le-plus[of v],
        cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
        cut-tac n = n in aGroup.nsum-mem[of K - f], assumption,
        rule allI, simp add:funcset-mem, assumption, simp)
    apply (frule-tac z = Amin n ( $\lambda u. v (f u)$ ) and z' = v ( $\Sigma_e K f n$ ) and
        w = v (f (Suc n)) in amin-aminTr,
        rule-tac i = amin (Amin n ( $\lambda u. v (f u)$ )) (v (f (Suc n))) and
        j = amin (v ( $\Sigma_e K f n$ )) (v (f (Suc n))) and
        k = v ( $\Sigma_e K f n \pm (f (Suc n))$ ) in ale-trans, assumption+)
    done

lemma (in Corps) Amin-le-add:valuation K v;  $\forall j \leq n. f j \in \text{carrier } K$   $\implies$ 
    Amin n (v ∘ f)  $\leq$  (v (nsum K f n))

```

```

by (frule Amin-le-addTr[of v n f], simp)

lemma (in Corps) value-ge-add:[valuation K v;  $\forall j \leq n. f j \in \text{carrier } K$ ;
 $\forall j \leq n. z \leq ((v \circ f) j)] \implies z \leq (v (\Sigma_e K f n))$ 
apply (frule Amin-le-add[of v n f], assumption+,
      cut-tac Amin-ge[of n v \circ f z],
      rule ale-trans, assumption+)
apply (rule allI, rule impI,
      simp add:comp-def Zset-def,
      rule value-in-aug-inf[of v], assumption+, simp+)
done

lemma (in Corps) Vr-ideal-powTr1:[valuation K v; ideal (Vr K v) I;
 $I \neq \text{carrier } (Vr K v); b \in I] \implies b \in (vp K v)$ 
by (frule ideal-sub-vp[of v I], assumption+, simp add:subsetD)

```

2.5 pow of vp and n-value – convergence –

```

lemma (in Corps) n-value-x-1:[valuation K v;  $0 \leq n;$ 
 $x \in (vp K v)^{(Vr K v)^n} \implies n \leq (n\text{-val } K v x)$ ]

apply ((case-tac  $n = \infty$ , simp add:r-apow-def,
        simp add:Vr-0-f-0, cut-tac field-is-ring,
        frule Ring.ring-zero[of K], frule val-inf-n-val-inf[of v 0],
        assumption+, simp add:value-of-zero),
       (case-tac  $n = 0$ , simp add:r-apow-def,
        subst val-pos-n-val-pos[THEN sym, of v x], assumption+,
        simp add:Vr-mem-f-mem,
        subst val-pos-mem-Vr[of v x], assumption+,
        simp add:Vr-mem-f-mem, assumption,
        simp add:r-apow-def, frule Vr-ring[of v],
        frule vp-pow-ideal[of v na n],
        frule Ring.ideal-subset[of Vr K v (vp K v) \diamond(Vr K v) (na n) x],
        assumption+, frule Vr-mem-f-mem[of v x], assumption+))

```

```

apply (case-tac  $x = 0_K$ , simp,
       frule value-of-zero[of v],
       simp add:val-inf-n-val-inf,
       simp add:n-value-idealTr[of v na n],
       frule val-Pg[of v], erule conjE, simp, erule conjE,
       frule Lv-pos[of v],
       rotate-tac -4, frule sym, thin-tac  $v (Pg K v) = Lv K v$ , simp,
       frule val-poss-mem-Vr[of v Pg K v], assumption+,
       frule val-Rxa-gt-a[of v Pg K v \diamond(Vr K v) (na n) x],

```

```

frule Vr-integral[of v],
simp only: Vr-0-f-0[of v, THEN sym],
frule Idomain.idom-potent-nonzero[of Vr K v Pg K v na n],
assumption+, simp, simp add:Ring,npClose, assumption+)

apply (thin-tac x ∈ Vr K v ◇p (Pg K v ↨(Vr K v) (na n)),
      thin-tac ideal (Vr K v) (Vr K v ◇p (Pg K v ↨(Vr K v) (na n)))))

apply (simp add: Vr-exp-f-exp[of v Pg K v],
      simp add:val-exp-ring[THEN sym, of v Pg K v],
      simp add:n-val[THEN sym, of v x],
      frule val-nonzero-z[of v Pg K v], assumption+,
      erule exE, simp,
      frule aposs-na-poss[of n], simp add: less-le,
      simp add:asprod-amult,

frule-tac w = z in amult-pos-mono-r[of - ant (int (na n))
n-val K v x], simp,
cut-tac an-na[of n], simp add:an-def, assumption+)
done

lemma (in Corps) n-value-x-1-nat:[valuation K v; x ∈ (vp K v)◇(Vr K v) n] ==>
(an n) ≤ (n-val K v x)
apply (cut-tac an-nat-pos[of n])
apply( frule n-value-x-1[of v an n x], assumption+)
apply (simp add:r-apow-def)
apply (case-tac n = 0, simp, simp)
apply (cut-tac aless-nat-less[THEN sym, of 0 n])
apply simp
unfolding less-le
apply simp
apply (cut-tac an-neq-inf [of n])
apply simp
apply (simp add: na-an)
apply assumption
done

lemma (in Corps) n-value-x-2:[valuation K v; x ∈ carrier (Vr K v);
n ≤ (n-val K v x); 0 ≤ n] ==> x ∈ (vp K v) (Vr K v) n
apply (frule Vr-ring[of v],
frule val-Pg[of v], erule conjE,
simp, erule conjE, drule sym,
frule Lv-pos[of v], simp,
frule val-poss-mem-Vr[of v Pg K v], assumption+)

apply (case-tac n = ∞,
simp add:r-apow-def, cut-tac inf-ge-any[of n-val K v x],
frule ale-antisym[of n-val K v x ∞], assumption+,

```

```

frule val-inf-n-val-inf[THEN sym, of v x],
simp add:Vr-mem-f-mem, simp,
frule value-inf-zero[of v x],
simp add:Vr-mem-f-mem, simp+, simp add:Vr-0-f-0)

apply (case-tac n = 0,
      simp add:r-apow-def,
      simp add:r-apow-def,
      subst n-value-idealTr[of v na n], assumption+,
      simp add:apos-na-pos)
apply (rule val-Rxa-gt-a-1[of v Pg K v ^ (Vr K v) (na n) x],
       assumption+,
       rule Ring.npClose, assumption+,
       simp add:Vr-0-f-0[THEN sym, of v],
       frule Vr-integral[of v],
       frule val-poss-mem-Vr[of v Pg K v], assumption+,
       simp add:Idomain.idom-potent-nonzero)

apply (simp add:Vr-exp-f-exp,
      simp add:val-exp-ring[THEN sym, of v],
      rotate-tac -5, drule sym,
      frule Lv-z[of v], erule exE, simp,
      frule aposs-na-poss[of n], simp add: less-le,
      simp add:asprod-amult, subst n-val[THEN sym, of v x],
      assumption+,
      simp add:Vr-mem-f-mem, simp,
      subst amult-pos-mono-r[of - ant (int (na n)) n-val K v x],
      assumption,
      cut-tac an-na[of n], simp add:an-def, assumption+)
done

```

lemma (in Corps) n-value-x-2-nat:[valuation K v; x ∈ carrier (Vr K v);
 $(an n) \leq ((n\text{-val } K v) x)] \implies x \in (vp K v)^{\diamond} (Vr K v)^n$

by (frule n-value-x-2[of v x an n], assumption+,
 simp, simp add:r-apow-def,
 case-tac an n = ∞, simp add:an-def, simp,
 case-tac n = 0, simp,
 subgoal-tac an n ≠ 0, simp, simp add:na-an,
 rule contrapos-pp, simp+, simp add:an-def)

lemma (in Corps) n-val-n-pow:[valuation K v; x ∈ carrier (Vr K v); 0 ≤ n] \implies
 $(n \leq (n\text{-val } K v x)) = (x \in (vp K v)^{(Vr K v)^n})$

by (rule iffI, simp add:n-value-x-2, simp add:n-value-x-1)

lemma (in Corps) eval-in-vpr-apow:[valuation K v; x ∈ carrier K; 0 ≤ n;
 $y \in \text{carrier } K; n\text{-val } K v x = n\text{-val } K v y; x \in (vp K v)^{(Vr K v)^n}] \implies$

$y \in (vp K v) (Vr K v) n$
apply (*frule n-value-x-1*[*of v n x*], *assumption+*, *simp*,
rule n-value-x-2[*of v y n*], *assumption+*,
frule mem-vp-apow-mem-Vr[*of v n x*], *assumption+*)
apply (*frule val-pos-mem-Vr*[*THEN sym, of v x*], *assumption+*, *simp*,
simp add:val-pos-n-val-pos[*of v x*],
simp add:val-pos-n-val-pos[*THEN sym, of v y*],
simp add:val-pos-mem-Vr, *assumption+*)
done

lemma (in Corps) convergenceTr: $\llbracket \text{valuation } K v; x \in \text{carrier } K; b \in \text{carrier } K;$
 $b \in (vp K v) (Vr K v) n; (\text{Abs} (n\text{-val } K v x)) \leq n \rrbracket \implies$
 $x \cdot_r b \in (vp K v) (Vr K v) (n + (n\text{-val } K v x))$

apply (*cut-tac Abs-pos*[*of n-val K v x*],
frule ale-trans[*of 0 Abs (n-val K v x) n*], *assumption+*,
thin-tac $0 \leq \text{Abs} (n\text{-val } K v x)$)
apply (*frule Vr-ring*[*of v*],
frule-tac aadd-le-mono[*of Abs (n-val K v x) n n-val K v x*],
cut-tac Abs-x-plus-x-pos[*of n-val K v x*],
frule ale-trans[*of 0 Abs (n-val K v x) + n-val K v x*,
n + n-val K v x], *assumption+*,
thin-tac $0 \leq \text{Abs} (n\text{-val } K v x) + \text{n-val } K v x$,
thin-tac $\text{Abs} (n\text{-val } K v x) + \text{n-val } K v x \leq n + \text{n-val } K v x$,
rule n-value-x-2[*of v x ·_r b n + n-val K v x*], *assumption+*)
apply (*frule n-value-x-1*[*of v n b*], *assumption+*)
apply (*frule aadd-le-mono*[*of n n-val K v b n-val K v x*],
frule ale-trans[*of 0 n + n-val K v x n-val K v b + n-val K v x*],
assumption)
apply (*thin-tac* $0 \leq n + \text{n-val } K v x$,
thin-tac $n \leq \text{n-val } K v b$,
thin-tac $n + \text{n-val } K v x \leq \text{n-val } K v b + \text{n-val } K v x$,
simp add:aadd-commute[*of n-val K v b n-val K v x*])
apply (*frule n-val-valuation*[*of v*],
simp add:val-t2p[*THEN sym, of n-val K v x b*],
cut-tac field-is-ring,
frule Ring.ring-tOp-closed[*of K x b*], *assumption+*,
simp add:val-pos-n-val-pos[*THEN sym, of v x ·_r b*],
simp add:val-pos-mem-Vr,
frule n-val-valuation[*of v*],
subst val-t2p[*of n-val K v*], *assumption+*,
frule n-value-x-1[*of v n b*], *assumption+*,
simp add:aadd-commute[*of n-val K v x n-val K v b*],
rule aadd-le-mono[*of n n-val K v b n-val K v x*], *assumption+*)
done

lemma (in Corps) convergenceTr1: $\llbracket \text{valuation } K v; x \in \text{carrier } K;$
 $b \in (vp K v) (Vr K v) (n + \text{Abs} (n\text{-val } K v x)); 0 \leq n \rrbracket \implies$
 $x \cdot_r b \in (vp K v) (Vr K v) n$

```

apply (cut-tac field-is-ring,
         frule Vr-ring[of v],
         frule vp-apow-ideal[of v n + Abs (n-val K v x)],
         cut-tac Abs-pos[of n-val K v x],
         rule aadd-two-pos[of n Abs (n-val K v x)], assumption+)

apply (frule Ring.ideal-subset[of Vr K v vp K v (Vr K v) (n + Abs (n-val K v x))
          b], assumption+,
          frule Vr-mem-f-mem[of v b], assumption,
          frule convergenceTr[of v x b n + Abs (n-val K v x)], assumption+,
          rule aadd-pos-le[of n Abs (n-val K v x)], assumption)

apply (frule apos-in-aug-inf[of n],
         cut-tac Abs-pos[of n-val K v x],
         frule apos-in-aug-inf[of Abs (n-val K v x)],
         frule n-value-in-aug-inf[of v x], assumption+,
         frule aadd-assoc-i[of n Abs (n-val K v x) n-val K v x],
         assumption+,
         cut-tac Abs-x-plus-x-pos[of n-val K v x])

apply (frule-tac Ring.ring-tOp-closed[of K x b], assumption+,
          rule n-value-x-2[of v x ·r b n], assumption+)

apply (subst val-pos-mem-Vr[THEN sym, of v x ·r b], assumption+,
          subst val-pos-n-val-pos[of v x ·r b], assumption+)

apply (frule n-value-x-1[of v n + Abs(n-val K v x) + n-val K v x x ·r b],
          subst aadd-assoc-i, assumption+,
          rule aadd-two-pos[of n], assumption+,
          rule ale-trans[of 0 n + Abs (n-val K v x) + n-val K v x
             n-val K v (x ·r b)],
          simp, simp add:aadd-two-pos, assumption,
          frule n-value-x-1[of v n + Abs (n-val K v x) b],
          cut-tac Abs-pos[of n-val K v x],
          rule aadd-two-pos[of n Abs (n-val K v x)], assumption+)

apply (frule n-val-valuation[of v],
          subst val-t2p[of n-val K v], assumption+)
apply (frule aadd-le-mono[of n + Abs (n-val K v x) n-val K v b
          n-val K v x],
          simp add:aadd-commute[of n-val K v b n-val K v x],
          rule ale-trans[of n n + (Abs (n-val K v x) + n-val K v x)
             n-val K v x + n-val K v b],
          frule aadd-pos-le[of Abs (n-val K v x) + n-val K v x n],
          simp add:aadd-commute[of n], assumption+)
done

lemma (in Corps) vp-potent-zero:  $\llbracket \text{valuation } K v; 0 \leq n \rrbracket \implies$ 
 $(n = \infty) = (\text{vp } K v (Vr K v)^n = \{\mathbf{0}_{Vr K v}\})$ 

```

```

apply (rule iffI)
apply (simp add:r-apow-def, rule contrapos-pp, simp+,
         frule apos-neq-minf[of n],
         cut-tac mem-ant[of n], simp, erule exE, simp,
         simp add:ant-0[THEN sym], thin-tac n = ant z)

apply (case-tac z = 0, simp add:ant-0, simp add:r-apow-def,
         frule Vr-ring[of v],
         frule Ring.ring-one[of Vr K v], simp,
         simp add:Vr-0-f-0, simp add:Vr-1-f-1,
         frule value-of-one[of v], simp, simp add:value-of-zero,
         cut-tac n = z in zneq-aneg[of - 0], simp only:ant-0)
apply (simp add:r-apow-def,
         frule-tac n = na (ant z) in n-value-idealTr[of v],
         simp add:na-def,
         simp, thin-tac vp K v ◇(Vr K v) (na (ant z)) = {0}_{Vr K v},
         frule Vr-ring[of v],
         frule Pg-in-Vr[of v],
         frule-tac n = na (ant z) in Ring,npClose[of Vr K v Pg K v],
         assumption)
apply (frule-tac a = (Pg K v) ∼(Vr K v) (na (ant z)) in
         Ring.a-in-principal[of Vr K v], assumption,
         simp, frule Vr-integral[of v],
         frule val-Pg[of v], simp, (erule conjE)+,
         frule-tac n = na (ant z) in Idomain.idom-potent-nonzero[of Vr K v
         Pg K v], assumption+,
         simp add:Vr-0-f-0, simp)

```

done

lemma (in Corps) Vr-potent-eqTr1: $\llbracket \text{valuation } K v; 0 \leq n; 0 \leq m;$
 $(vp K v)^{(Vr K v)} n = (vp K v)^{(Vr K v)} m; m = 0 \rrbracket \implies n = m$

```

apply (frule Vr-ring[of v],
         simp add:r-apow-def,
         case-tac n = 0, simp,
         case-tac n = ∞, simp,
         frule val-Pg[of v], erule conjE, simp,
         erule conjE,
         rotate-tac -3, drule sym,
         frule Lv-pos[of v], simp,
         frule val-poss-mem-Vr[of v Pg K v], assumption+,
         drule sym, simp, simp add:Vr-0-f-0)

```

```

apply (simp,
         drule sym,
         frule Ring.ring-one[of Vr K v], simp,
         frule n-value-x-1-nat[of v 1_r(Vr K v) na n], assumption,

```

```

simp add:an-na, simp add:Vr-1-f-1,
frule n-val-valuation[of v],
simp add:value-of-one[of n-val K v])
done

lemma (in Corps) Vr-potent-eqTr2:[valuation K v;
(vp K v)  $\diamond$ (Vr K v) n = (vp K v)  $\diamond$ (Vr K v) m]  $\implies$  n = m

apply (frule Vr-ring[of v],
frule val-Pg[of v], simp, (erule conjE)+,
rotate-tac -1, frule sym, thin-tac v (Pg K v) = Lv K v,
frule Lv-pos[of v], simp)

apply (subgoal-tac 0  $\leq$  int n, subgoal-tac 0  $\leq$  int m,
frule n-value-idealTr[of v m]) apply simp apply simp
apply(
thin-tac vp K v  $\diamond$ (Vr K v) m = Vr K v  $\diamond_p$  (Pg K v  $\sim$ (Vr K v) m),
frule n-value-idealTr[of v n], simp, simp,
thin-tac vp K v  $\diamond$ (Vr K v) n = Vr K v  $\diamond_p$  (Pg K v  $\sim$ (Vr K v) m),
frule val-poss-mem-Vr[of v Pg K v], assumption+)

apply (frule Lv-z[of v], erule exE,
rotate-tac -4, drule sym, simp,
frule eq-ideal-eq-val[of v Pg K v  $\sim$ (Vr K v) n Pg K v  $\sim$ (Vr K v) m])
apply (rule Ring,npClose, assumption+, rule Ring,npClose, assumption+)
apply (simp only:Vr-exp-f-exp,
simp add:val-exp-ring[THEN sym, of v Pg K v],
thin-tac Vr K v  $\diamond_p$  (Pg K v  $\sim^K$  n) = Vr K v  $\diamond_p$  (Pg K v  $\sim^K$  m))

apply (case-tac n = 0, simp, case-tac m = 0, simp,
simp only:of-nat-0-less-iff[THEN sym, of m],
simp only:asprod-amult a-z-z,
simp only:ant-0[THEN sym], simp only:aeq-zeq, simp)
apply (auto simp add: asprod-mult)
done

lemma (in Corps) Vr-potent-eq:[valuation K v; 0  $\leq$  n; 0  $\leq$  m;
(vp K v) (Vr K v) n = (vp K v) (Vr K v) m]  $\implies$  n = m

apply (frule n-val-valuation[of v],
case-tac m = 0,
simp add:Vr-potent-eqTr1)

apply (case-tac n = 0,
frule sym, thin-tac vp K v (Vr K v) n = vp K v (Vr K v) m,
frule Vr-potent-eqTr1[of v m n], assumption+,
rule sym, assumption,
frule vp-potent-zero[of v n], assumption+)

```

```

apply (case-tac  $n = \infty$ , simp,
  thin-tac  $vp K v (Vr K v)^\infty = \{\mathbf{0}_{Vr K v}\}$ ,
  frule  $vp\text{-potent-zero}[\text{THEN } \text{sym}, \text{ of } v m]$ , assumption+, simp,
  simp,
  frule  $vp\text{-potent-zero}[\text{THEN } \text{sym}, \text{ of } v m]$ , assumption+, simp,
  thin-tac  $vp K v (Vr K v)^m \neq \{\mathbf{0}_{Vr K v}\}$ )

```

```

apply (frule  $a\text{pos}s\text{-na-poss}[\text{of } n]$ , subst  $\text{less-}le$ , simp,
  frule  $a\text{pos}s\text{-na-poss}[\text{of } m]$ , subst  $\text{less-}le$ , simp,
  simp  $\text{add:}r\text{-apow-def}$ ,
  frule  $Vr\text{-potent-eqTr2}[\text{of } v \text{ na } n \text{ na } m]$ , assumption+,
  thin-tac  $vp K v \diamondsuit (Vr K v) (na n) = vp K v \diamondsuit (Vr K v) (na m)$ ,
  simp  $\text{add:}aeq\text{-nat-eq}[\text{THEN } \text{sym}]$ )

```

done

the following two lemma (in Corps) s are used in completion of K

```

lemma (in Corps)  $Vr\text{-prime-maximalTr1}:\llbracket \text{valuation } K v; x \in \text{carrier } (Vr K v);$ 
 $Suc 0 < n \rrbracket \implies x \cdot_r (Vr K v) (x^K (n - Suc 0)) \in (Vr K v) \diamondsuit_p (x^K n)$ 

```

```

apply (frule  $Vr\text{-ring}[\text{of } v]$ ,
  subgoal-tac  $x^K n = x^K (Suc (n - Suc 0))$ ,
  simp  $\text{del:}Suc\text{-pred}$ ,
  rotate-tac  $-1$ , drule  $\text{sym}$ )

```

```

apply (subst  $Vr\text{-tOp-f-tOp}$ , assumption+,
  subst  $Vr\text{-exp-f-exp}[\text{of } v, \text{ THEN } \text{sym}]$ , assumption+,
  simp  $\text{only:}Ring.\text{npClose}$ , simp  $\text{del:}Suc\text{-pred}$ )

```

```

apply (cut-tac  $\text{field-is-ring}$ ,
  frule  $Ring.\text{npClose}[\text{of } K x n - Suc 0]$ ,
  frule  $Vr\text{-mem-f-mem}[\text{of } v x]$ , assumption+,
  frule  $Vr\text{-mem-f-mem}[\text{of } v x]$ , assumption+)
  apply (simp  $\text{add:}Ring.\text{ring-tOp-commute}[\text{of } K x x^K (n - Suc 0)]$ )

```

apply (*rule* $Ring.\text{a-in-principal}$, *assumption*)

```

apply (frule  $Ring.\text{npClose}[\text{of } Vr K v x n]$ , assumption,
  simp  $\text{add:}Vr\text{-exp-f-exp}$ )

```

apply (*simp* $\text{only:}Suc\text{-pred}$)

done

```

lemma (in Corps)  $Vr\text{-prime-maximalTr2}:\llbracket \text{valuation } K v; x \in vp K v; x \neq \mathbf{0};$ 
 $Suc 0 < n \rrbracket \implies x \notin Vr K v \diamondsuit_p (x^K n) \wedge x^K (n - Suc 0) \notin (Vr K v) \diamondsuit_p$ 
 $(x^K n)$ 

```

apply (*frule* $Vr\text{-ring}[\text{of } v]$)

```

apply (frule  $vp\text{-mem-Vr-mem}[\text{of } v x]$ , assumption,
  frule  $Ring.\text{npClose}[\text{of } Vr K v x n]$ ,
  simp  $\text{only:}Vr\text{-exp-f-exp}$ )

```

```

apply (cut-tac  $\text{field-is-ring}$ ,
  cut-tac  $\text{field-is-idom}$ ,
  frule  $Vr\text{-mem-f-mem}[\text{of } v x]$ , assumption+,
  frule  $Idomain.\text{idom-potent-nonzero}[\text{of } K x n]$ , assumption+)

```

apply (*rule* conjI)

```

apply (rule contrapos-pp, simp+)
apply (frule val-Rxa-gt-a[of v x^K n x],
      simp, simp add: Vr-exp-f-exp, assumption+)
apply (simp add:val-exp-ring[THEN sym, of v x n])
apply (frule val-nonzero-z[of v x], assumption+, erule exE,
      simp add:asprod-amult a-z-z)
apply (simp add:vp-mem-val-poss[of v x])
apply (rule contrapos-pp, simp+)
apply (frule val-Rxa-gt-a[of v x^K n x^K (n - Suc 0)])
  apply (simp, frule Ring.npClose[of Vr K v x n - Suc 0], assumption+)
    apply (simp add:Vr-exp-f-exp)
    apply (frule Ring.npClose[of Vr K v x n - Suc 0], assumption+,
           simp add:Vr-exp-f-exp, assumption)
    apply (simp add:val-exp-ring[THEN sym, of v x])
    apply (simp add:vp-mem-val-poss[of v x])
    apply (frule val-nonzero-z[of v x], assumption+, erule exE,
           simp add:asprod-amult a-z-z)
done

lemma (in Corps) Vring-prime-maximal: [| valuation K v; prime-ideal (Vr K v) I;
I ≠ {0} |] ==> maximal-ideal (Vr K v) I
apply (frule Vr-ring[of v],
       frule Ring.prime-ideal-proper[of Vr K v I], assumption+,
       frule Ring.prime-ideal-ideal[of Vr K v I], assumption+,
       frule ideal-pow-vp[of v I],
       frule n-value-idealTr[of v na (n-val K v (Ig K v I))],
       simp, simp, assumption+)

apply (case-tac na (n-val K v (Ig K v I)) = 0,
      simp, frule Ring.Rxa-one[of Vr K v], simp,
      frule Suc-leI[of 0 na (n-val K v (Ig K v I))],
      thin-tac 0 < na (n-val K v (Ig K v I)))
apply (case-tac na (n-val K v (Ig K v I)) = Suc 0, simp,
      frule Pg-in-Vr[of v])
apply (frule vp-maximal[of v],
      frule Ring.maximal-ideal-ideal[of Vr K v vp K v], assumption+,
      subst Ring.idealprod-whole-r[of Vr K v vp K v], assumption+)

apply (rotate-tac -1, drule not-sym,
      frule le-neq-implies-less[of Suc 0 na (n-val K v (Ig K v I))],
      assumption+,
      thin-tac Suc 0 ≤ na (n-val K v (Ig K v I)),
      thin-tac Suc 0 ≠ na (n-val K v (Ig K v I)),
      thin-tac Vr K v ◇_p 1_r Vr K v = carrier (Vr K v))
apply (frule val-Pg[of v], simp, (erule conjE)+,
      frule Lv-pos[of v], rotate-tac -2, drule sym)
apply (frule val-poss-mem-Vr[of v Pg K v],
      frule vp-mem-val-poss[THEN sym, of v Pg K v], assumption+, simp)

```

```

apply (frule Vr-prime-maximalTr2[of v Pg K v
na (n-val K v (Ig K v I))],
simp add:vp-mem-val-poss[of v Pg K v], assumption+, erule conjE)
apply (frule Ring.npMulDistr[of Vr K v Pg K v na 1 na (n-val K v (Ig K v I)) -
Suc 0], assumption+, simp add:na-1)

apply (rotate-tac 8, drule sym)
apply (frule Ring.a-in-principal[of Vr K v
Pg K v  $\nwarrow$  (Vr K v) (na (n-val K v (Ig K v I)))], simp add:Ring.npClose)

apply (simp add:Vr-exp-f-exp[of v])
apply (simp add:Ring.ring-l-one[of Vr K v])
apply (frule n-value-idealTr[THEN sym,
of v na (n-val K v (Ig K v I))], simp)
apply (simp add:Vr-exp-f-exp)
apply (rotate-tac 6, drule sym, simp)
apply (thin-tac I ≠ carrier (Vr K v),
thin-tac I = vp K v  $\diamondsuit$  (Vr K v) (na (n-val K v (Ig K v I))),
thin-tac v (Pg K v) = Lv K v,
thin-tac (Vr K v) \diamondsuit_p ((Pg K v) \cdot_r (Vr K v)
((Pg K v) \sim^K (na ((n-val K v) (Ig K v I)) - (Suc 0))) =
I,
thin-tac Pg K v ∈ carrier K,
thin-tac Pg K v ≠ 0,
thin-tac Pg K v \sim^K (na ((n-val K v) (Ig K v I))) =
Pg K v \cdot_r Vr K v Pg K v \sim^K ((na ((n-val K v) (Ig K v I)) - Suc 0))

apply (simp add:prime-ideal-def, erule conjE,
drule-tac x = Pg K v in bspec, assumption,
drule-tac x = Pg K v \sim^K (na (n-val K v (Ig K v I)) - Suc 0) in bspec)
apply (simp add:Vr-exp-f-exp[THEN sym, of v])
apply (rule Ring.npClose[of Vr K v Pg K v], assumption+)
apply simp
done

```

From the above lemma (in Corps) , we see that a valuation ring is of dimension one.

```

lemma (in Corps) field-frac1:  $\llbracket 1_r \neq 0; x \in \text{carrier } K \rrbracket \implies x = x \cdot_r ((1_r)^{-K})$ 
by (simp add:invf-one,
cut-tac field-is-ring,
simp add:Ring.ring-r-one[THEN sym])

```

```

lemma (in Corps) field-frac2:  $\llbracket x \in \text{carrier } K; x \neq 0 \rrbracket \implies x = (1_r) \cdot_r ((x^{-K})^{-K})$ 
by (cut-tac field-is-ring, simp add:field-inv-inv,
simp add:Ring.ring-l-one[THEN sym])

```

```

lemma (in Corps) val-nonpos-inv-pos:  $\llbracket \text{valuation } K v; x \in \text{carrier } K;$ 

```

$\neg 0 \leq (v x) \Rightarrow 0 < (v (x^K))$
by (case-tac $x = \mathbf{0}_K$, simp add:value-of-zero,
frule Vr-ring[of v],
simp add:aneg-le[of $0 v x$],
frule value-of-inv[THEN sym, of $v x$], assumption+,
frule aless-minus[of $v x 0$], simp)

lemma (in Corps) frac-Vr-is-K:[valuation $K v$; $x \in \text{carrier } K$] \Rightarrow
 $\exists s \in \text{carrier } (Vr K v). \exists t \in \text{carrier } (Vr K v) - \{\mathbf{0}\}. x = s \cdot_r (t^K)$
apply (frule Vr-ring[of v],
frule has-val-one-neq-zero[of v])
apply (case-tac $x = \mathbf{0}_K$,
frule Ring.ring-one[of $Vr K v$],
frule field-frac1[of x],
simp only: Vr-1-f-1, frule Ring.ring-zero[of $Vr K v$],
simp add: Vr-0-f-0 Vr-1-f-1, blast)
apply (case-tac $0 \leq (v x)$,
frule val-pos-mem-Vr[THEN sym, of $v x$], assumption+, simp,
frule field-frac1[of x], assumption+,
frule has-val-one-neq-zero[of v],
frule Ring.ring-one[of $Vr K v$], simp only: Vr-1-f-1, blast)
apply (frule val-nonpos-inv-pos[of $v x$], assumption+,
cut-tac invf-inv[of x], erule conjE,
frule val-poss-mem-Vr[of $v x^K$], assumption+)
apply (frule Ring.ring-one[of $Vr K v$], simp only: Vr-1-f-1,
frule field-frac2[of x], assumption+)
apply (cut-tac invf-closed1[of x], blast, simp+)
done

lemma (in Corps) valuations-eqTr1:[valuation $K v$; valuation $K v'$;
 $Vr K v = Vr K v'$; $\forall x \in \text{carrier } (Vr K v). v x = v' x \Rightarrow v = v'$]
apply (rule funcset-eq [of - carrier K],
simp add:valuation-def, simp add:valuation-def,
rule ballI,
frule-tac $x = x$ in frac-Vr-is-K[of v], assumption+,
(erule bexE)+, simp, erule conjE)
apply (frule-tac $x = t$ in Vr-mem-f-mem[of v'], assumption,
cut-tac $x = t$ in invf-closed1, simp, simp, erule conjE)
apply (frule-tac $x = s$ in Vr-mem-f-mem[of v'], assumption+,
simp add:val-t2p, simp add:value-of-inv)
done

lemma (in Corps) ridmap-rhom:[valuation $K v$; valuation $K v'$;
carrier $(Vr K v) \subseteq \text{carrier } (Vr K v')$]
ridmap $(Vr K v) \in rHom (Vr K v) (Vr K v')$
apply (frule Vr-ring[of v], frule Vr-ring[of v'],
subst rHom-def, simp, rule conjI)
apply (simp add:aHom-def, rule conjI,
rule Pi-I, simp add:ridmap-def subsetD,

```

simp add:ridmap-def restrict-def extensional-def,
(rule ballI)+,
frule Ring.ring-is-ag[of Vr K v], simp add:aGroup.ag-pOp-closed,
simp add:Vr-pOp-f-pOp subsetD)
apply (rule conjI, (rule ballI)+, simp add:ridmap-def,
      simp add:Ring.ring-tOp-closed, simp add:Vr-tOp-f-tOp subsetD,
      frule Ring.ring-one[of Vr K v], frule Ring.ring-one[of Vr K v'],
      simp add:Vr-1-f-1, simp add:ridmap-def )
done

lemma (in Corps) contract-ideal:[valuation K v; valuation K v';
carrier (Vr K v) ⊆ carrier (Vr K v')] ==>
ideal (Vr K v) (carrier (Vr K v) ∩ vp K v')
apply (frule-tac ridmap-rhom[of v v'], assumption+,
frule Vr-ring[of v], frule Vr-ring[of v'])
apply (cut-tac TwoRings.i-contract-ideal[of Vr K v Vr K v'
ridmap (Vr K v) vp K v'],
subgoal-tac (i-contract (ridmap (Vr K v)) (Vr K v) (Vr K v')
(vp K v')) = (carrier (Vr K v) ∩ vp K v'))
apply simp
apply(thin-tac ideal (Vr K v) (i-contract (ridmap (Vr K v))
(Vr K v) (Vr K v') (vp K v')), simp add:i-contract-def invim-def ridmap-def, blast)
apply (simp add:TwoRings-def TwoRings-axioms-def, simp)
apply (simp add:vp-ideal)
done

lemma (in Corps) contract-prime:[valuation K v; valuation K v';
carrier (Vr K v) ⊆ carrier (Vr K v')] ==>
prime-ideal (Vr K v) (carrier (Vr K v) ∩ vp K v')
apply (frule-tac ridmap-rhom[of v v'], assumption+,
frule Vr-ring[of v],
frule Vr-ring[of v'],
cut-tac TwoRings.i-contract-prime[of Vr K v Vr K v' ridmap (Vr K v)
vp K v'])
apply (subgoal-tac (i-contract (ridmap (Vr K v)) (Vr K v) (Vr K v')
(vp K v')) = (carrier (Vr K v) ∩ vp K v'), simp,
thin-tac prime-ideal (Vr K v) (i-contract
(ridmap (Vr K v)) (Vr K v) (Vr K v') (vp K v')), simp add:i-contract-def invim-def ridmap-def, blast)
apply (simp add:TwoRings-def TwoRings-axioms-def, simp)
apply (simp add:vp-prime)
done

lemma (in Corps) valuation-equivTr:[valuation K v; valuation K v';
x ∈ carrier K; 0 < (v' x); carrier (Vr K v) ⊆ carrier (Vr K v')]==>
0 ≤ (v x)

```

```

apply (rule contrapos-pp, simp+,
  frule val-nonpos-inv-pos[of v x], assumption+,
  case-tac  $x = \mathbf{0}_K$ , simp add:value-of-zero[of v]) apply (
  cut-tac invf-closed1[of x], simp, erule conjE,
  frule aless-imp-le[of 0 v ( $x^{-K}$ )])
apply (simp add:val-pos-mem-Vr[of v  $x^{-K}$ ],
  frule subsetD[of carrier (Vr K v) carrier (Vr K v')  $x^{-K}$ ],
  assumption+,
  frule val-pos-mem-Vr[THEN sym, of  $v' x^{-K}$ ], assumption+)
apply (simp, simp add:value-of-inv[of  $v' x$ ],
  cut-tac ale-minus[of 0 -  $v' x$ ], thin-tac  $0 \leq - v' x$ ,
  simp only:a-minus-minus,
  cut-tac aneg-less[THEN sym, of  $v' x - 0$ ], simp,
  assumption, simp)
done

lemma (in Corps) contract-maximal: [| valuation K v; valuation K  $v'$ ;
  carrier (Vr K v) ⊆ carrier (Vr K  $v'$ ) |] ==>
maximal-ideal (Vr K v) (carrier (Vr K v) ∩ vp K  $v'$ )
apply (frule Vr-ring[of v],
  frule Vr-ring[of  $v'$ ],
  rule Vring-prime-maximal, assumption+,
  simp add:contract-prime)
apply (frule vp nonzero[of  $v'$ ],
  frule vp-ideal[of  $v'$ ],
  frule Ring.ideal-zero[of Vr K  $v' vp K v'$ ], assumption+,
  frule sets-not-eq[of vp K  $v'$  { $\mathbf{0}_{(Vr K v')}$ }],
  simp add: singleton-sub[of  $\mathbf{0}_{(Vr K v')}$  carrier (Vr K  $v'$ )],
  erule bxE, simp add:Vr-0-f-0)

apply (case-tac  $a \in$  carrier (Vr K v), blast,
  frule-tac  $x = a$  in vp-mem-Vr-mem[of  $v'$ ], assumption+,
  frule-tac  $x = a$  in Vr-mem-f-mem[of  $v'$ ], assumption+,
  subgoal-tac  $a \in$  carrier (Vr K v), blast,
  frule-tac  $x_1 = a$  in val-pos-mem-Vr[THEN sym, of v], assumption+,
  simp, frule val-nonpos-inv-pos[of v], assumption+)

apply (frule-tac  $y = v (a^{-K})$  in aless-imp-le[of 0],
  cut-tac  $x = a$  in invf-closed1, simp,
  frule-tac  $x = a^{-K}$  in val-pos-mem-Vr[of v], simp, assumption+)
apply (frule-tac  $c = a^{-K}$  in subsetD[of carrier (Vr K v)
  carrier (Vr K  $v'$ )], assumption+) apply (
  frule-tac  $x = a^{-K}$  in val-pos-mem-Vr[of  $v'$ ],
  simp, simp only:value-of-inv[of  $v'$ ], simp,
  simp add:value-of-inv[of  $v'$ ])
apply (frule-tac  $y = - v' a$  in ale-minus[of 0], simp add:a-minus-minus,
  frule-tac  $x = a$  in vp-mem-val-poss[of  $v'$ ], assumption+,
  simp)

```

done

2.6 Equivalent valuations

definition

$v\text{-equiv} :: [-, 'r \Rightarrow ant, 'r \Rightarrow ant] \Rightarrow \text{bool}$ **where**
 $v\text{-equiv } K v1 v2 \longleftrightarrow n\text{-val } K v1 = n\text{-val } K v2$

lemma (in Corps) $\text{valuation-equivTr1} :: [\text{valuation } K v; \text{valuation } K v'; \forall x \in \text{carrier } K. 0 \leq (v x) \rightarrow 0 \leq (v' x)] \Rightarrow \text{carrier } (\text{Vr } K v) \subseteq \text{carrier } (\text{Vr } K v')$

apply (frule Vr-ring[of v],
frule Vr-ring[of v'])
apply (rule subsetI,
case-tac $x = \mathbf{0}_K$, simp, simp add: Vr-def Sr-def,
frule-tac $x1 = x$ in val-pos-mem-Vr[THEN sym, of v],
frule-tac $x = x$ in Vr-mem-f-mem[of v],
simp, frule-tac $x = x$ in Vr-mem-f-mem[of v], assumption+)
apply (drule-tac $x = x$ in bspec, simp add: Vr-mem-f-mem)
apply simp
apply (subst val-pos-mem-Vr[THEN sym, of v'], assumption+,
simp add: Vr-mem-f-mem, assumption+)
done

lemma (in Corps) $\text{valuation-equivTr2} :: [\text{valuation } K v; \text{valuation } K v'; \text{carrier } (\text{Vr } K v) \subseteq \text{carrier } (\text{Vr } K v'); vp K v = \text{carrier } (\text{Vr } K v) \cap vp K v'] \Rightarrow \text{carrier } (\text{Vr } K v') \subseteq \text{carrier } (\text{Vr } K v)$

apply (frule Vr-ring[of v], frule Vr-ring[of v'])
apply (rule subsetI)
apply (case-tac $x = \mathbf{0}_{(\text{Vr } K v')}$, simp,
subst Vr-0-f-0[of v'], assumption+,
subst Vr-0-f-0[of v, THEN sym], assumption,
simp add: Ring.ring-zero)
apply (rule contrapos-pp, simp+)
apply (frule-tac $x = x$ in Vr-mem-f-mem[of v'], assumption+)
apply (simp add: val-pos-mem-Vr[THEN sym, of v])
apply (cut-tac $y = v x$ in aneg-le[of 0], simp)
apply (simp add: Vr-0-f-0[of v'])
apply (frule-tac $x = v x$ in aless-minus[of - 0], simp,
thin-tac $v x < 0$, thin-tac $\neg 0 \leq v x$)
apply (simp add: value-of-inv[THEN sym, of v])
apply (cut-tac $x = x$ in invf-closed1, simp, simp, erule conjE)
apply (frule-tac $x1 = x^{-K}$ in vp-mem-val-poss[THEN sym, of v],
assumption, simp, erule conjE)
apply (frule vp-ideal [of v'])
apply (frule-tac $x = x^{-K}$ and $r = x$ in Ring.ideal-ring-multiple[of Vr K v'
vp K v'], assumption+)

```

apply (frule-tac  $x = x^{-K}$  in vp-mem-Vr-mem[of v'], assumption+)
apply (frule-tac  $x = x$  and  $y = x^{-K}$  in Ring.ring-tOp-commute[of Vr K v'],
assumption+, simp,
thin-tac  $x \cdot_r Vr K v' x^{-K} = x^{-K} \cdot_r Vr K v' x$ )
apply (simp add:Vr-tOp-f-tOp)
apply (cut-tac  $x = x$  in linuf, simp, simp)
apply (cut-tac field-is-ring, frule Ring.ring-one[of K])
apply (frule ideal-inc-elem0val-whole[of v' 1_r vp K v'],
assumption+, simp add:value-of-one, assumption+)
apply (frule vp-not-whole[of v'], simp)
done

lemma (in Corps) eq-carr-eq-Vring:  $\llbracket \text{valuation } K v; \text{valuation } K v';$ 
carrier (Vr K v) = carrier (Vr K v')  $\implies Vr K v = Vr K v'$ 
apply (simp add:Vr-def Sr-def)
done

lemma (in Corps) valuations-equiv:  $\llbracket \text{valuation } K v; \text{valuation } K v';$ 
 $\forall x \in \text{carrier } K. 0 \leq (v x) \longrightarrow 0 \leq (v' x) \rrbracket \implies v\text{-equiv } K v v'$ 

apply (frule Vr-ring[of v], frule Vr-ring[of v'])

apply (frule valuation-equivTr1[of v v'], assumption+)

apply (frule contract-maximal [of v v'], assumption+)

apply (frule Vr-local[of v (carrier (Vr K v) \cap vp K v')],
assumption+)

apply (frule valuation-equivTr2[of v v'], assumption+,
frule equalityI[of carrier (Vr K v) carrier (Vr K v')],
assumption+,
thin-tac carrier (Vr K v) \subseteq carrier (Vr K v'),
thin-tac carrier (Vr K v') \subseteq carrier (Vr K v)))

apply (frule vp-ideal[of v'],
frule Ring.ideal-subset1[of Vr K v' vp K v'], assumption,
simp add:Int-absorb1,
thin-tac  $\forall x \in \text{carrier } K. 0 \leq v x \longrightarrow 0 \leq v' x$ ,
thin-tac vp K v' \subseteq carrier (Vr K v'),
thin-tac ideal (Vr K v') (vp K v'),
thin-tac maximal-ideal (Vr K v) (vp K v'))

apply (simp add:v-equiv-def,
rule valuations-eqTr1[of n-val K v n-val K v'],

```

```

(simp add:n-val-valuation)+,
rule eq-carr-eq-Vring[of n-val K v n-val K v'],
(simp add:n-val-valuation)+,
subst Vr-n-val-Vr[THEN sym, of v], assumption+,
subst Vr-n-val-Vr[THEN sym, of v'], assumption+)
apply (rule ballI,
frule n-val-valuation[of v],
frule n-val-valuation[of v'],
frule-tac x1 = x in val-pos-mem-Vr[THEN sym, of n-val K v],
simp add:Vr-mem-f-mem, simp,
frule Vr-n-val-Vr[THEN sym, of v], simp,
thin-tac carrier (Vr K (n-val K v)) = carrier (Vr K v'),
frule-tac x1 = x in val-pos-mem-Vr[THEN sym, of v'],
simp add:Vr-mem-f-mem,
simp,
frule-tac x = x in val-pos-n-val-pos[of v'],
simp add:Vr-mem-f-mem, simp,
frule-tac x = x in ideal-apow-n-val[of v],
simp add:Vr-n-val-Vr[THEN sym, of v], simp)
apply (frule eq-carr-eq-Vring[of v v'], assumption+,
frule-tac x = x in ideal-apow-n-val[of v'], assumption,
simp add:Vr-n-val-Vr[THEN sym, of v],
thin-tac Vr K v' ◇_p x = vp K v' (Vr K v') (n-val K v x),
frule-tac n = n-val K v' x and m = n-val K v x in
Vr-potent-eq[of v'], assumption+,
frule sym, assumption+)
done

lemma (in Corps) val-equiv-axiom1:valuation K v ==> v-equiv K v v
apply (simp add:v-equiv-def)
done

lemma (in Corps) val-equiv-axiom2:[ valuation K v; valuation K v';
v-equiv K v v ] ==> v-equiv K v' v
apply (simp add:v-equiv-def)
done

lemma (in Corps) val-equiv-axiom3:[ valuation K v; valuation K v';
valuation K v'; v-equiv K v v'; v-equiv K v' v' ] ==> v-equiv K v v"
apply (simp add:v-equiv-def)
done

lemma (in Corps) n-val-equiv-val:[ valuation K v ] ==>
v-equiv K v (n-val K v)
apply (frule valuations-equiv[of v n-val K v], simp add:n-val-valuation)
apply (rule ballI, rule impI, simp add:val-pos-n-val-pos,
assumption)
done

```

2.7 Prime divisors

definition

```
prime-divisor :: [-, 'b ⇒ ant] ⇒
  ('b ⇒ ant) set ((2P _ _) [96,97]96) where
    P_K v = {v'. valuation K v' ∧ v-equiv K v v'}
```

definition

```
prime-divisors :: - ⇒ ('b ⇒ ant) set set (Pds1 96) where
  Pds_K = {P. ∃ v. valuation K v ∧ P = P_K v }
```

definition

```
normal-valuation-belonging-to-prime-divisor :: [-, ('b ⇒ ant) set] ⇒ ('b ⇒ ant) ((ν- _) [96,97]96) where
  ν_K P = n-val K (SOME v. v ∈ P)
```

```
lemma (in Corps) val-in-P-valuation:[valuation K v; v' ∈ P_K v] ⇒
  valuation K v'
apply (simp add:prime-divisor-def)
done
```

```
lemma (in Corps) vals-in-P-equiv:[ valuation K v; v' ∈ P_K v] ⇒
  v-equiv K v v'
apply (simp add:prime-divisor-def)
done
```

```
lemma (in Corps) v-in-prime-v:valuation K v ⇒ v ∈ P_K v
apply (simp add:prime-divisor-def,
       frule val-equiv-axiom1[of v], assumption+)
done
```

```
lemma (in Corps) some-in-prime-divisor:valuation K v ⇒
  (SOME w. w ∈ P_K v) ∈ P_K v
apply (subgoal-tac P_K v ≠ {}, rule nonempty-some[of P_K v], assumption+,
       frule v-in-prime-v[of v])
apply blast
done
```

```
lemma (in Corps) valuation-some-in-prime-divisor:valuation K v
  ⇒ valuation K (SOME w. w ∈ P_K v)
apply (frule some-in-prime-divisor[of v],
       simp add:prime-divisor-def)
done
```

```
lemma (in Corps) valuation-some-in-prime-divisor1:P ∈ Pds ⇒
  valuation K (SOME w. w ∈ P)
apply (simp add:prime-divisors-def, erule exE)
apply (simp add:valuation-some-in-prime-divisor)
```

```

done

lemma (in Corps) representative-of-pd-valuation:
   $P \in Pds \implies \text{valuation } K (\nu_K P)$ 
apply (simp add:prime-divisors-def,
  erule exE, erule conjE,
  simp add:normal-valuation-belonging-to-prime-divisor-def,
  frule-tac  $v = v$  in valuation-some-in-prime-divisor)

apply (rule n-val-valuation, assumption+)
done

lemma (in Corps) some-in-P-equiv:valuation  $K v \implies$ 
   $v\text{-equiv } K v (\text{SOME } w. w \in P_K v)$ 
apply (frule some-in-prime-divisor[of  $v$ ])
apply (rule vals-in-P-equiv, assumption+)
done

lemma (in Corps) n-val-n-val1: $P \in Pds \implies n\text{-val } K (\nu_K P) = (\nu_K P)$ 
apply (simp add: normal-valuation-belonging-to-prime-divisor-def,
  frule valuation-some-in-prime-divisor1[of  $P$ ])
apply (rule n-val-n-val[of SOME  $v. v \in P$ ], assumption+)
done

lemma (in Corps) P-eq-val-equiv:[valuation  $K v; \text{valuation } K v']] \implies$ 
   $(v\text{-equiv } K v v') = (P_K v = P_K v')$ 
apply (rule iffI,
  rule equalityI,
  rule subsetI, simp add:prime-divisor-def, erule conjE,
  frule val-equiv-axiom2[of  $v v'$ ], assumption+,
  rule val-equiv-axiom3[of  $v' v$ ], assumption+,
  rule subsetI, simp add:prime-divisor-def, erule conjE)
apply (rule val-equiv-axiom3[of  $v v'$ ], assumption+,
  frule v-in-prime-v[of  $v$ ], simp,
  thin-tac  $P_K v = P_K v'$ ,
  simp add:prime-divisor-def,
  rule val-equiv-axiom2[of  $v' v$ ], assumption+)
done

lemma (in Corps) unique-n-valuation:[ $P \in Pds_K; P' \in Pds$ ] \implies
   $(P = P') = (\nu_K P = \nu_K P')$ 
apply (rule iffI, simp)
apply (simp add:prime-divisors-def,
  (erule exE)+, (erule conjE)+)
apply (simp add:normal-valuation-belonging-to-prime-divisor-def,
  frule-tac  $v = v$  in some-in-P-equiv,
  frule-tac  $v = va$  in some-in-P-equiv,
  subgoal-tac  $v\text{-equiv } K (\text{SOME } w. w \in P_K v) (\text{SOME } w. w \in P_K va)$ )
apply (frule-tac  $v = v$  in some-in-prime-divisor,

```

```

frule-tac  $v = va$  in some-in-prime-divisor,
frule-tac  $v = v$  and  $v' = \text{SOME } w. w \in P_K v$  and  $v'' = \text{SOME } w. w \in P_K va$  in val-equiv-axiom3)
apply (simp add:prime-divisor-def,
      simp add:prime-divisor-def, assumption+,
      frule-tac  $v = va$  and  $v' = \text{SOME } w. w \in P_K va$  in
                  val-equiv-axiom2,
      simp add:prime-divisor-def, assumption+)
apply (frule-tac  $v = v$  and  $v' = \text{SOME } w. w \in P_K va$  and  $v'' = va$  in
                  val-equiv-axiom3,
      simp add:prime-divisor-def,
      simp add:prime-divisor-def, assumption+,
      frule-tac  $v = v$  and  $v' = va$  in P-eq-val-equiv, assumption+)
apply simp
apply (simp add:v-equiv-def)
done

lemma (in Corps) n-val-representative:  $P \in Pds \implies (\nu_K P) \in P$ 
apply (simp add:prime-divisors-def,
       erule exE, erule conjE,
       simp add:normal-valuation-belonging-to-prime-divisor-def,
       frule-tac  $v = v$  in valuation-some-in-prime-divisor,
       frule-tac  $v = \text{SOME } w. w \in P_K v$  in
                  n-val-equiv-val,
       frule-tac  $v = v$  in some-in-P-equiv,
       frule-tac  $v = v$  and  $v' = \text{SOME } w. w \in P_K v$  and  $v'' = n\text{-val } K (\text{SOME } w. w \in P_K v)$  in val-equiv-axiom3,
       assumption+,
       frule-tac  $v = v$  in n-val-valuation,
       simp add:prime-divisor-def, simp add:n-val-valuation)
done

lemma (in Corps) val-equiv-eq-pdiv:[ $P \in Pds_K; P' \in Pds_K; \text{valuation } K v;$ 
                                     $\text{valuation } K v'; v\text{-equiv } K v v'; v \in P; v' \in P'$ ]  $\implies P = P'$ 
apply (simp add:prime-divisors-def,
       (erule exE)+, (erule conjE)+)
apply (rename-tac w w',
       frule-tac  $v = w$  in vals-in-P-equiv[of - v], simp,
       frule-tac  $v = w'$  in vals-in-P-equiv[of - v'], simp,
       frule-tac  $v = w$  and  $v' = v$  and  $v'' = v'$  in val-equiv-axiom3,
       assumption+,
       frule-tac  $v = w'$  in val-equiv-axiom2[of - v'], assumption+,
       frule-tac  $v = w$  and  $v' = v'$  and  $v'' = w'$  in val-equiv-axiom3,
       assumption+) apply simp+
apply (simp add:P-eq-val-equiv)
done

lemma (in Corps) distinct-p-divisors:[ $P \in Pds_K; P' \in Pds_K$ ]  $\implies$ 
 $(\neg P = P') = (\neg v\text{-equiv } K (\nu_K P) (\nu_K P'))$ 

```

```

apply (rule iffI,
      rule contrapos-pp, simp+,
      frule val-equiv-eq-pdiv[of P P' νK P νK P'], assumption+,
      simp add: representative-of-pd-valuation,
      simp add: representative-of-pd-valuation, assumption)
apply (rule n-val-representative[of P], assumption,
      rule n-val-representative[of P'], assumption,
      simp,
      rule contrapos-pp, simp+, frule sym, thin-tac P = P',
      simp,
      frule representative-of-pd-valuation[of P],
      frule val-equiv-axiom1[of νK P], simp)
done

```

2.8 Approximation

definition

```

valuations :: [-, nat, nat ⇒ ('r ⇒ ant)] ⇒ bool where
valuations K n vv ↔ (∀ j ≤ n. valuation K (vv j))

```

definition

```

vals-nonequiv :: [-, nat, nat ⇒ ('r ⇒ ant)] ⇒ bool where
vals-nonequiv K n vv ↔ valuations K n vv ∧
(∀ j ≤ n. ∀ l ≤ n. j ≠ l → ¬ (v-equiv K (vv j) (vv l)))

```

definition

```

Ostrowski-elem :: [-, nat, nat ⇒ ('b ⇒ ant), 'b] ⇒ bool where
Ostrowski-elem K n vv x ↔
(0 < (vv 0 (1r K ±K (-a K x)))) ∧ (∀ j ∈ nset (Suc 0) n. 0 < (vv j x))

```

lemma (in Corps) Ostrowski-elem-0: vals-nonequiv K n vv; x ∈ carrier K;

Ostrowski-elem K n vv x] ⇒ 0 < (vv 0 (1_r ± (-_a x)))

apply (simp add: Ostrowski-elem-def)

done

lemma (in Corps) Ostrowski-elem-Suc: vals-nonequiv K n vv; x ∈ carrier K;

Ostrowski-elem K n vv x; j ∈ nset (Suc 0) n] ⇒ 0 < (vv j x)

apply (simp add: Ostrowski-elem-def)

done

lemma (in Corps) vals-nonequiv-valuation: vals-nonequiv K n vv; m ≤ n] ⇒ valuation K (vv m)

apply (simp add: vals-nonequiv-def, erule conjE)

apply (thin-tac ∀ j ≤ n. ∀ l ≤ n. j ≠ l → ¬ (v-equiv K (vv j) (vv l)))

apply (simp add: valuations-def)

done

```

lemma (in Corps) vals-nonequiv:[ $\llbracket \text{vals-nonequiv } K (\text{Suc } (\text{Suc } n)) vv;$ 
 $i \leq (\text{Suc } (\text{Suc } n)); j \leq (\text{Suc } (\text{Suc } n)); i \neq j \rrbracket \implies$ 
 $\neg (\text{v-equiv } K (vv i) (vv j))$ 
apply (simp add:vals-nonequiv-def)
done

lemma (in Corps) skip-vals-nonequiv:vals-nonequiv [ $K (\text{Suc } (\text{Suc } n)) vv \implies$ 
 $\text{vals-nonequiv } K (\text{Suc } n) (\text{compose } \{l. l \leq (\text{Suc } n)\} vv (\text{skip } j))$ 
apply (subst vals-nonequiv-def)
apply (rule conjI)
apply (subst valuations-def, rule allI, rule impI,
      simp add:compose-def)
apply (cut-tac l = ja and n = Suc n in skip-mem[of - - j], simp,
      frule-tac m = skip j ja in vals-nonequiv-valuation[of
      Suc (Suc n) vv], simp, assumption)
apply ((rule allI, rule impI)+, rule impI,
      cut-tac l = ja and n = Suc n in skip-mem[of - - j], simp,
      cut-tac l = l and n = Suc n in skip-mem[of - - j], simp+)
apply (cut-tac i = ja and j = l in skip-inj[of - Suc n - j], simp+,
      simp add:compose-def,
      rule-tac i = skip j ja and j = skip j l in
      vals-nonequiv[of n], assumption+)
done

lemma (in Corps) not-v-equiv-reflex:[valuation K v; valuation K v';
 $\neg \text{v-equiv } K v v' \rrbracket \implies \neg \text{v-equiv } K v' v$ 
apply (simp add:v-equiv-def)
done

lemma (in Corps) nonequiv-ex-Ostrowski-elem:[valuation K v; valuation K v';
 $\neg \text{v-equiv } K v v' \rrbracket \implies \exists x \in \text{carrier } K. 0 \leq (v x) \wedge (v' x) < 0$ 
apply (subgoal-tac  $\neg (\forall x \in \text{carrier } K. 0 \leq (v x) \longrightarrow 0 \leq (v' x))$ )
prefer 2
apply (rule contrapos-pp, simp+,
      frule valuations-equiv[of v v'], assumption+,
      simp add:val-equiv-axiom2[of v v'])
apply (simp, erule bxE, erule conjE, simp add:aneg-le)
apply blast
done

lemma (in Corps) field-op-minus:[ $a \in \text{carrier } K; b \in \text{carrier } K; b \neq \mathbf{0} \rrbracket \implies$ 
 $-_a (a \cdot_r (b^{-K})) = (-_a a) \cdot_r (b^{-K})$ 
apply (cut-tac invf-closed1[of b], simp,
      erule conjE, cut-tac field-is-ring,
      simp add:Ring.ring-inv1[of K a b^{-K}], simp)
done

lemma (in Corps) field-one-plus-frac1:[ $a \in \text{carrier } K; b \in \text{carrier } K; b \neq \mathbf{0} \rrbracket \implies$ 
 $1_r \pm (a \cdot_r (b^{-K})) = (b \pm a) \cdot_r (b^{-K})$ 

```

```

apply (cut-tac field-is-ring,
        cut-tac invf-closed1[of b], simp+, erule conjE,
        cut-tac field-is-idom)
apply (rule Idomain.idom-mult-cancel-r[of K 1r ± (a ·r (b-K))]
        (b ± a) ·r (b-K) b, assumption+,
        frule Idomain.idom-is-ring[of K], frule Ring.ring-is-ag[of K],
        rule aGroup.ag-pOp-closed [of K], assumption+,
        simp add:Ring.ring-one,rule Ring.ring-tOp-closed, assumption+,
        rule Ring.ring-tOp-closed, assumption+,
        frule Ring.ring-is-ag[of K],
        rule aGroup.ag-pOp-closed, assumption+,
        subst Ring.ring-distrib2[of K b], assumption+,
        simp add:Ring.ring-one, simp add:Ring.ring-tOp-closed,
        simp add:Ring.ring-l-one) thm Ring.ring-distrib2[of K b-K]
apply (subst Ring.ring-distrib2[of K b-K], assumption+,
        simp add:Ring.ring-tOp-commute[of K b b-K],
        subst linvf[of b], simp,
        subst Ring.ring-distrib2[of K b], assumption+,
        simp add:Ring.ring-one, simp add:Ring.ring-tOp-closed,
        simp add:Ring.ring-l-one, simp)
done

lemma (in Corps) field-one-plus-frc2: $\llbracket a \in \text{carrier } K; b \in \text{carrier } K; a \pm b \neq \mathbf{0} \rrbracket \implies 1_r \pm (-_a (a \cdot_r (a \pm b)^{-K})) = b \cdot_r ((a \pm b)^{-K})$ 
apply (frule field-op-minus[of a a ± b],
        cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
        simp add:aGroup.ag-pOp-closed, assumption, simp,
        thin-tac -_a (a ·r (a ± b)-K) = (-_a a) ·r (a ± b)-K)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
        frule aGroup.ag-mOp-closed[of K a], assumption,
        frule field-one-plus-frc1[of -_a a a ± b],
        simp add:aGroup.ag-pOp-closed, simp, simp,
        thin-tac 1r ± (-_a a) ·r (a ± b)-K = (a ± b ± -_a a) ·r (a ± b)-K,
        simp add:aGroup.ag-pOp-assoc[of K a b -_a a],
        simp add:aGroup.ag-pOp-commute[of K b -_a a],
        simp add:aGroup.ag-pOp-assoc[THEN sym],
        simp add:aGroup.ag-r-inv1,
        simp add:aGroup.ag-l-zero)
done

lemma (in Corps) field-one-plus-frc3: $\llbracket x \in \text{carrier } K; x \neq 1_r; 1_r \pm x \cdot_r (1_r \pm -_a x) \neq \mathbf{0} \rrbracket \implies 1_r \pm -_a x \cdot_r (1_r \pm x \cdot_r (1_r \pm -_a x))^{-K} = (1_r \pm -_a x^{\sim K} (\text{Suc } (\text{Suc } 0))) \cdot_r (1_r \pm x \cdot_r (1_r \pm -_a x))^{-K}$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag, frule Ring.ring-one,
        cut-tac invf-closed1[of 1r ± x ·r (1r ± -a x)], simp, erule conjE)
apply (subst Ring.ring-inv1-1, assumption+,
        subst field-one-plus-frc1[of -a x 1r ± x ·r (1r ± -a x)])

```

```

apply (rule aGroup.ag-mOp-closed, assumption+,
       rule aGroup.ag-pOp-closed, assumption+,
       rule Ring.ring-tOp-closed, assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+, rule aGroup.ag-mOp-closed,
       assumption+,
       subst Ring.ring-distrib1, assumption+,
       rule aGroup.ag-mOp-closed, assumption+)
apply (simp add:Ring.ring-r-one)
apply (simp add:Ring.ring-inv1-2[THEN sym, of K x x])
apply (subgoal-tac 1r ± (x ± -a x ·r x) ± -a x = 1r ± -a x^K (Suc (Suc 0)),
       simp,
       frule Ring.ring-tOp-closed[of K x x], assumption+)

apply (frule Ring.ring-tOp-closed[of K x x], assumption+,
       frule aGroup.ag-mOp-closed[of K x ·r x], assumption+,
       frule aGroup.ag-mOp-closed[of K x], assumption+)
apply (subst aGroup.ag-pOp-assoc, assumption+,
       rule aGroup.ag-pOp-closed, assumption+)
apply (rule aGroup.ag-pOp-add-l[of K x ± -a x ·r x ± -a x
      -a x^K (Suc (Suc 0)) 1r], assumption+,
      (rule aGroup.ag-pOp-closed, assumption+)+,
      rule aGroup.ag-mOp-closed, assumption+, rule Ring.npClose,
      assumption+,
      subst aGroup.ag-pOp-commute, assumption+,
      simp add:aGroup.ag-pOp-assoc aGroup.ag-r-inv1 aGroup.ag-r-zero)
apply (simp add:Ring.ring-l-one)
apply simp
apply (rule aGroup.ag-pOp-closed, assumption+,
       rule Ring.ring-tOp-closed, assumption+,
       rule aGroup.ag-pOp-closed, assumption+,
       rule aGroup.ag-mOp-closed[of K x], assumption+)
done

lemma (in Corps) OstrowskiTr1: [| valuation K v; s ∈ carrier K; t ∈ carrier K;
0 ≤ (v s); v t < 0 |] ==> s ± t ≠ 0
apply (rule contrapos-pp, simp+,
       cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       simp only:aGroup.ag-plus-zero[THEN sym, of K s t])
apply (simp add:val-minus-eq[of v t])
done

lemma (in Corps) OstrowskiTr2: [| valuation K v; s ∈ carrier K; t ∈ carrier K;
0 ≤ (v s); v t < 0 |] ==> 0 < (v (1r ± (-a ((t ·r ((s ± t)^-K))))))
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       frule OstrowskiTr1[of v s t], assumption+,
       frule field-one-plus-frac2[of t s], assumption+,
       simp add:aGroup.ag-pOp-commute)
apply (subst aGroup.ag-pOp-commute[of K s t], assumption+, simp,
       simp add:aGroup.ag-pOp-commute[of K t s],

```

```

thin-tac  $1_r \pm -_a (t \cdot_r (s \pm t)^- K) = s \cdot_r (s \pm t)^- K$ ,
frule aGroup.ag-pOp-closed[of  $K s t$ ], assumption+,
cut-tac invf-closed1[of  $s \pm t$ ], simp, erule conjE)
apply (simp add:val-t2p,
       simp add:value-of-inv,
       frule aless-le-trans[of  $v t 0 v s$ ], assumption+,
       frule value-less-eq[THEN sym, of  $v t s$ ], assumption+,
       simp add:aGroup.ag-pOp-commute,
       frule aless-diff-poss[of  $v t v s$ ], simp add:diff-ant-def, simp)
done

lemma (in Corps) OstrowskiTr3:[valuation  $K v$ ;  $s \in carrier K$ ;  $t \in carrier K$ ;
 $0 \leq (v t)$ ;  $v s < 0$ ]  $\implies 0 < (v (t \cdot_r ((s \pm t)^- K)))$ 
apply (frule aless-le-trans[of  $v s 0 v t$ ], assumption+,
       cut-tac field-is-ring, frule Ring.ring-is-ag[of  $K$ ],
       frule aGroup.ag-pOp-closed[of  $K s t$ ], assumption+,
       frule OstrowskiTr1[of  $v t s$ ], assumption+,
       frule value-less-eq[THEN sym, of  $v s t$ ], assumption+)
apply (simp add:aGroup.ag-pOp-commute[of  $K t s$ ],
       cut-tac invf-closed1[of  $s \pm t$ ], simp) apply (
       erule conjE, simp add:val-t2p[of  $v$ ], simp add:value-of-inv)
apply (cut-tac aless-diff-poss[of  $v s v t$ ],
       simp add:diff-ant-def, simp+)
done

lemma (in Corps) restrict-Ostrowski-elem:[ $x \in carrier K$ ;
 $Ostrowski\text{-}elem K (\text{Suc } (\text{Suc } n)) vv x$ ]  $\implies Ostrowski\text{-}elem K (\text{Suc } n) vv x$ 
apply (simp add:Ostrowski-elem-def,
       erule conjE, rule ballI, simp add:nset-def,
       insert lessI [of Suc n])
done

lemma (in Corps) restrict-vals-nonequiv:vals-nonequiv  $K (\text{Suc } (\text{Suc } n)) vv \implies$ 
 $vals\text{-}nonequiv K (\text{Suc } n) vv$ 
apply (simp add:vals-nonequiv-def,
       erule conjE, simp add:valuations-def)
done

lemma (in Corps) restrict-vals-nonequiv1:vals-nonequiv  $K (\text{Suc } (\text{Suc } n)) vv \implies$ 
 $vals\text{-}nonequiv K (\text{Suc } n) (\text{compose } \{h. h \leq (\text{Suc } n)\} vv (\text{skip } 1))$ 
apply (simp add:vals-nonequiv-def, (erule conjE)+,
       rule conjI,
       thin-tac  $\forall j \leq \text{Suc } (\text{Suc } n). \forall l \leq \text{Suc } (\text{Suc } n). j \neq l \longrightarrow$ 
 $\neg v\text{-equiv } K (vv j) (vv l)$ ,
       simp add:valuations-def, rule allI, rule impI,
       simp add:compose-def skip-def nset-def)
apply ((rule allI, rule impI)+, rule impI)
apply (simp add:compose-def skip-def nset-def)
done

```

```

lemma (in Corps) restrict-vals-nonequiv2: vals-nonequiv K (Suc (Suc n)) vv]
  ==> vals-nonequiv K (Suc n) (compose {j. j ≤ (Suc n)} vv (skip 2))
apply (simp add:vals-nonequiv-def, (erule conjE)+,
       rule conjI,
       thin-tac ∀ j≤Suc (Suc n). ∀ l≤Suc (Suc n). j ≠ l —>
           ¬ v-equiv K (vv j) (vv l),
       simp add:valuations-def,
       rule allI, rule impI)
apply (simp add:compose-def skip-def nset-def,
       (rule allI, rule impI)+, rule impI,
       simp add:compose-def skip-def nset-def)
done

lemma (in Corps) OstrowskiTr31: valuation K v; s ∈ carrier K;
  0 < (v (1r ± (−a s))) ==> s ≠ 0
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (rule contrapos-pp, simp+)
apply (simp add:aGroup.ag-inv-zero,
       frule Ring.ring-one[of K], simp add:aGroup.ag-r-zero)
apply (simp add:value-of-one)
done

lemma (in Corps) OstrowskiTr32: valuation K v; s ∈ carrier K;
  0 < (v (1r ± (−a s))) ==> 0 ≤ (v s)
apply (rule contrapos-pp, simp+,
       cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       simp add:aneg-le,
       frule has-val-one-neq-zero[of v])
apply (frule OstrowskiTr31[of v s], assumption+,
       frule not-sym,
       frule Ring.ring-one[of K])
apply (frule value-less-eq[THEN sym, of v −a s 1r],
       simp add:aGroup.ag-mOp-closed, assumption+,
       simp add:val-minus-eq)
apply (simp add:value-of-one,
       frule aGroup.ag-mOp-closed[of K s], assumption+,
       simp add:aGroup.ag-pOp-commute[of K −a s 1r],
       simp add:val-minus-eq)
done

lemma (in Corps) OstrowskiTr4: valuation K v; s ∈ carrier K; t ∈ carrier K;
  0 < (v (1r ± (−a s))); 0 < (v (1r ± (−a t))) ==>
  0 < (v (1r ± (−a (s ·r t))))
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       frule Ring.ring-one[of K])
apply (subgoal-tac 1r ± (−a (s ·r t)) =
       1r ± (−a s) ± (s ·r (1r ± (−a t))), simp,
       thin-tac 1r ± −a (s ·r t) = 1r ± −a s ± s ·r (1r ± −a t))

```

```

apply (frule aGroup.ag-mOp-closed[of K s], assumption+,
         frule aGroup.ag-pOp-closed[of K 1r -a s], assumption+,
         frule aGroup.ag-mOp-closed[of K t], assumption+,
         frule aGroup.ag-pOp-closed[of K 1r -a t], assumption+,
         frule Ring.ring-tOp-closed[of K s 1r ± (-a t)], assumption+,
         frule amin-le-plus[of v 1r ± (-a s) s ·r (1r ± (-a t))], assumption+)
apply (frule amin-gt[of 0 v (1r ± -a s) v (s ·r (1r ± -a t))])
apply (simp add:val-t2p,
         frule OstrowskiTr32[of v s], assumption+,
         rule aadd-pos-poss[of v s v (1r ± -a t)], assumption+,
         simp add:Ring.ring-distrib1)
apply (frule aGroup.ag-mOp-closed[of K t], assumption,
         simp add:Ring.ring-distrib1 Ring.ring-r-one,
         frule aGroup.ag-mOp-closed[of K s], assumption+,
         subst aGroup.pOp-assocTr43, assumption+,
         simp add:Ring.ring-tOp-closed,
         simp add:aGroup.ag-l-inv1 aGroup.ag-r-zero,
         simp add:Ring.ring-inv1-2)
done

lemma (in Corps) OstrowskiTr5:[] vals-nonequiv K (Suc (Suc n)) vv;
s ∈ carrier K; t ∈ carrier K;
0 ≤ (vv (Suc 0)) s ∧ 0 ≤ (vv (Suc (Suc 0))) t;
Ostrowski-elem K (Suc n) (compose {j. j ≤ (Suc n)} vv (skip 1)) s;
Ostrowski-elem K (Suc n) (compose {j. j ≤ (Suc n)} vv (skip 2)) t] ⇒
Ostrowski-elem K (Suc (Suc n)) vv (s ·r t)
apply (erule conjE,
        cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
        frule-tac x = s and y = t in Ring.ring-tOp-closed[of K], assumption+,
        frule skip-vals-nonequiv[of n vv 1],
        frule skip-vals-nonequiv[of n vv 2],
        subst Ostrowski-elem-def, rule conjI)

apply (rule OstrowskiTr4,
        simp add:vals-nonequiv-valuation[of Suc (Suc n) vv 0],
        assumption+,
        frule Ostrowski-elem-0[of Suc n
          compose {j. j ≤ (Suc n)} vv (skip 1) s], assumption+,
        simp add:skip-def compose-def,
        frule Ostrowski-elem-0[of Suc n
          compose {j. j ≤ (Suc n)} vv (skip 2) t], assumption+,
        simp add:skip-def compose-def)

apply (rule ballI,
        case-tac j = Suc 0,
        frule-tac j = Suc 0 in Ostrowski-elem-Suc[of Suc n
          compose {j. j ≤ (Suc n)} vv (skip 2) t], assumption+,
        simp add:nset-def) apply (
        thin-tac Ostrowski-elem K (Suc n) (compose {j. j ≤ Suc n} vv (skip 1)) s,

```

```

thin-tac Ostrowski-elem K (Suc n) (compose {j. j ≤ Suc n} vv (skip 2)) t,
thin-tac vals-nonequiv K (Suc n) (compose {l. l ≤ Suc n} vv (skip 1)),
frule vals-nonequiv-valuation[of Suc n
compose {j. j ≤ (Suc n)} vv (skip 2) Suc 0]
apply simp+
apply (simp add:skip-def compose-def,
simp add:val-t2p, simp add:aadd-pos-poss)

apply (case-tac j = Suc (Suc 0),
frule vals-nonequiv-valuation[of Suc n
compose {j. j ≤ Suc n} vv (skip 1) Suc 0],
simp,
frule-tac j = Suc 0 in Ostrowski-elem-Suc[of Suc n
compose {j. j ≤ (Suc n)} vv (skip 1) s],
assumption+, simp add:nset-def,
simp add:skip-def compose-def,
simp add:val-t2p, rule aadd-poss-poss, assumption+)
apply (frule-tac j = j in nsetTr1[of - Suc 0 Suc (Suc n)], assumption,
frule-tac j = j in nsetTr2[of - Suc 0 Suc n],
thin-tac j ∈ nset (Suc (Suc 0)) (Suc (Suc n)),
frule-tac j = j - Suc 0 in Ostrowski-elem-Suc[of Suc n
compose {j. j ≤ (Suc n)} vv (skip 1) s], assumption+)

apply (frule-tac j = j - Suc 0 in Ostrowski-elem-Suc[of Suc n
compose {j. j ≤ (Suc n)} vv (skip 2) t], assumption+,
thin-tac Ostrowski-elem K (Suc n) (compose {j. j ≤ (Suc n)} vv (skip 1)) s,
thin-tac Ostrowski-elem K (Suc n) (compose {j. j ≤ (Suc n)} vv (skip 2)) t,
thin-tac vals-nonequiv K (Suc n) (compose {j. j ≤ (Suc n)} vv (skip 1)),
thin-tac vals-nonequiv K (Suc n) (compose {j. j ≤ (Suc n)} vv (skip 2)))

apply (simp add:compose-def skip-def nset-def,
(erule conjE)+, simp, subgoal-tac ¬ (j - Suc 0 ≤ Suc 0), simp)
apply (frule-tac m = j in vals-nonequiv-valuation[of Suc (Suc n)],
assumption+,
simp add:val-t2p,
rule-tac x = vv j s and y = vv j t in aadd-pos-poss,
simp add:aless-imp-le, assumption)
apply simp
done

lemma (in Corps) one-plus-x-nonzero:[valuation K v; x ∈ carrier K; v x < 0]
    ⇒ 1_r ± x ∈ carrier K ∧ v (1_r ± x) < 0
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule Ring.ring-one[of K],
frule aGroup.ag-pOp-closed[of K 1_r x], assumption+,
simp)
apply (frule value-less-eq[of v x 1_r], assumption+,
simp add:value-of-one, simp add:aGroup.ag-pOp-commute)

```

done

lemma (in Corps) $\text{val-neg-nonzero} : \text{valuation } K v; x \in \text{carrier } K; v x < 0 \Rightarrow x \neq 0$

apply (rule contrapos-pp, simp+, simp add:value-of-zero)
apply (frule aless-imp-le[of $\infty 0$],
 cut-tac inf-ge-any[of 0],
 frule ale-antisym[of 0 ∞], assumption+, simp)

done

lemma (in Corps) OstrowskiTr6 : $\text{valuation } K v; x \in \text{carrier } K; \neg 0 \leq (v x) \Rightarrow (1_r \pm x \cdot_r (1_r \pm -_a x)) \in \text{carrier } K - \{\mathbf{0}\}$

apply (simp add:aneg-le,
 cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
 frule aGroup.ag-mOp-closed[of K x], assumption+,
 frule one-plus-x-nonzero[of v -_a x], assumption+,
 simp add:val-minus-eq, erule conjE)

apply (rule conjI,
 rule aGroup.ag-pOp-closed[of K], assumption+,
 simp add:Ring.ring-one, rule Ring.ring-tOp-closed[of K], assumption+)

apply (frule val-t2p[of v x 1_r ± (-_a x)], assumption+,
 frule val-neg-nonzero[of v x], assumption+,
 frule val-nonzero-z[of v x], assumption+, erule exE,
 frule-tac $z = z$ in aadd-less-mono-z[of v (1_r ± (-_a x)) 0],
 simp add:aadd-0-l,
 simp only:aadd-commute[of v (1_r ± -_a x)],
 frule-tac $x = \text{ant } z + v (1_r \pm -_a x)$ and $y = \text{ant } z$ in
 aless-trans[of - - 0], assumption,
 drule sym, simp)

apply (frule-tac $x = x$ and $y = 1_r \pm -_a x$ in Ring.ring-tOp-closed[of K],
 assumption+,
 frule one-plus-x-nonzero[of v x ·_r (1_r ± (-_a x))],
 assumption+, erule conjE,
 rule val-neg-nonzero[of v], assumption+)

done

lemma (in Corps) OstrowskiTr7 : $\text{valuation } K v; x \in \text{carrier } K; \neg 0 \leq (v x) \Rightarrow 1_r \pm -_a (x \cdot_r ((1_r \pm x \cdot_r (1_r \pm -_a x))^{-K})) =$

$(1_r \pm -_a x \pm x \cdot_r (1_r \pm -_a x)) \cdot_r ((1_r \pm x \cdot_r (1_r \pm -_a x))^{-K})$

apply (cut-tac field-is-ring,
 frule OstrowskiTr6[of v x], assumption+, simp, erule conjE,
 cut-tac field-is-idom,
 cut-tac invf-closed1[of 1_r ± x ·_r (1_r ± -_a x)], simp,
 frule Ring.ring-is-ag[of K],
 frule aGroup.ag-mOp-closed[of K x], assumption+,
 frule Ring.ring-one[of K],

```

frule aGroup.ag-pOp-closed[of K 1r -a x], assumption+,
rule Idomain.idom-mult-cancel-r[of K 1r ± -a (x ·r ((1r ± x ·r
(1r ± -a x))-K) (1r ± -a x ± x ·r (1r ± -a x)) ·r
((1r ± x ·r (1r ± -a x))-K) (1r ± x ·r (1r ± -a x))], assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+, rule aGroup.ag-mOp-closed,
assumption+,
rule Ring.ring-tOp-closed, assumption+, simp, rule Ring.ring-tOp-closed,
assumption+,
(rule aGroup.ag-pOp-closed, assumption+)+,
rule Ring.ring-tOp-closed, assumption+, simp, assumption+,
subst Ring.ring-tOp-assoc, assumption+,
rule aGroup.ag-pOp-closed, assumption+,
simp add:Ring.ring-tOp-closed, simp, simp)
apply (subst linvf[of 1r ± x ·r (1r ± -a x)], simp,
(subst Ring.ring-distrib2, assumption+)+, erule conjE)
apply (rule aGroup.ag-mOp-closed, assumption,
rule Ring.ring-tOp-closed, assumption+,
subst Ring.ring-r-one, assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+,
rule Ring.ring-tOp-closed, assumption+,
erule conjE,
simp add:Ring.ring-inv1-1,
simp add:Ring.ring-tOp-assoc[of K -a x (1r ± x ·r (1r ± -a x))-K],
simp add:linvf, simp add:Ring.ring-r-one Ring.ring-l-one,
frule Ring.ring-tOp-closed[of K x 1r ± -a x], assumption+,
simp add:aGroup.ag-pOp-assoc, simp add:aGroup.ag-pOp-commute)
apply simp
done

lemma (in Corps) Ostrowski-elem-nonzero:[vals-nonequiv K (Suc n) vv;
x ∈ carrier K; Ostrowski-elem K (Suc n) vv x] ⇒ x ≠ 0
apply (simp add:Ostrowski-elem-def,
frule conjunct1, fold Ostrowski-elem-def,
frule vals-nonequiv-valuation[of Suc n vv 0], simp)
apply (rule contrapos-pp, simp+,
cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
simp add:aGroup.ag-inv-zero, frule Ring.ring-one[of K],
simp add:aGroup.ag-r-zero, simp add:value-of-one)
done

lemma (in Corps) Ostrowski-elem-not-one:[vals-nonequiv K (Suc n) vv;
x ∈ carrier K; Ostrowski-elem K (Suc n) vv x] ⇒ 1r ± -a x ≠ 0
apply (frule vals-nonequiv-valuation [of Suc n vv Suc 0],
simp,
simp add:Ostrowski-elem-def, frule conjunct2,
fold Ostrowski-elem-def)
apply (subgoal-tac 0 < (vv (Suc 0) x),
rule contrapos-pp, simp+,

```

```

cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule Ring.ring-one[of K],
simp only:aGroup.ag-eq-diffzero[THEN sym, of K 1_r x],
drule sym, simp, simp add:value-of-one,
subgoal-tac Suc 0 ∈ nset (Suc 0) (Suc n), simp,
simp add:nset-def)
done

lemma (in Corps) val-unit-cond:[ valuation K v; x ∈ carrier K;
0 < (v (1_r ± -_a x)) ] ⇒ v x = 0
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule Ring.ring-one[of K])

apply (frule aGroup.ag-mOp-closed[of K 1_r], assumption+,
frule has-val-one-neq-zero[of v])

apply (frule aGroup.ag-pOp-assoc[of K -_a 1_r 1_r -_a x], assumption+,
simp add:aGroup.ag-mOp-closed, simp add:aGroup.ag-l-inv1,
frule aGroup.ag-mOp-closed[of K x], assumption+,
simp add:aGroup.ag-l-zero)
apply (subgoal-tac v (-_a x) = v ( -_a 1_r ± (1_r ± -_a x))) prefer 2
  apply simp
  apply (thin-tac -_a x = -_a 1_r ± (1_r ± -_a x),
frule value-less-eq[of v -_a 1_r 1_r ± -_a x],
assumption+,
rule aGroup.ag-pOp-closed, assumption+,
simp add:val-minus-eq value-of-one, simp add:val-minus-eq)
apply (simp add: value-of-one)
done

end

```

```

theory Valuation2
imports Valuation1
begin

lemma (in Corps) OstrowskiTr8:[valuation K v; x ∈ carrier K;
0 < v (1_r ± -_a x)] ⇒
0 < (v (1_r ± -_a (x ·_r ((1_r ± x ·_r (1_r ± -_a x))^-K))))
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (frule aGroup.ag-mOp-closed[of K x], assumption+,
frule Ring.ring-one[of K],
frule aGroup.ag-pOp-closed[of K 1_r -_a x], assumption+,
frule OstrowskiTr32[of v x], assumption+)
apply (case-tac x = 1_r_K, simp,
simp add:aGroup.ag-r-inv1, simp add:Ring.ring-times-x-0,
simp add:aGroup.ag-r-zero, cut-tac val-field-1-neq-0,
cut-tac invf-one, simp, simp add:Ring.ring-r-one,

```

```

simp add:aGroup.ag-r-inv1, assumption+)
apply (frule aGroup.ag-pOp-closed[of K 1_r x ·_r (1_r ± -_a x)], assumption+,
      rule Ring.ring-tOp-closed, assumption+)
apply (cut-tac invf-closed[of 1_r ± x ·_r (1_r ± -_a x)])
```

```

apply (cut-tac field-one-plus-frac3[of x], simp del:npow-suc,
      subst val-t2p[of v], assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+,
      rule aGroup.ag-mOp-closed, assumption+, rule Ring.npClose,
      assumption+,
      thin-tac 1_r ± -_a x ·_r (1_r ± x ·_r (1_r ± -_a x))⁻ K =
      (1_r ± -_a x)⁻ K (Suc (Suc 0))) ·_r (1_r ± x ·_r (1_r ± -_a x))⁻ K,
      subgoal-tac 1_r ± -_a x⁻ K (Suc (Suc 0)) = (1_r ± x) ·_r (1_r ± -_a x),
      simp del:npow-suc,
      thin-tac 1_r ± -_a x⁻ K (Suc (Suc 0)) = (1_r ± x) ·_r (1_r ± -_a x))
apply (subst val-t2p[of v], assumption+,
      rule aGroup.ag-pOp-closed, assumption+,
      subst value-of-inv[of v 1_r ± x ·_r (1_r ± -_a x)], tactic `CHANGED distinct-subgoals-tac`, assumption+)
```

```

apply (rule contrapos-pp, simp+,
      frule Ring.ring-tOp-closed[of K x (1_r ± -_a x)], assumption+,
      simp add:aGroup.ag-pOp-commute[of K 1_r],
      frule aGroup.ag-eq-diffzero[THEN sym, of K x ·_r (-_a x ± 1_r) -_a 1_r],
      assumption+, rule aGroup.ag-mOp-closed, assumption+)
apply (simp add:aGroup.ag-inv-inv[of K 1_r],
      frule eq-elems-eq-val[of x ·_r (-_a x ± 1_r) -_a 1_r v],
      thin-tac x ·_r (-_a x ± 1_r) = -_a 1_r,
      simp add:val-minus-eq value-of-one)
apply (simp add:val-t2p,
      frule aadd-pos-poss[of v x v (-_a x ± 1_r)], assumption+,
      simp,
      subst value-less-eq[THEN sym, of v 1_r x ·_r (1_r ± -_a x)],
      assumption+,
      rule Ring.ring-tOp-closed, assumption+,
      simp add:value-of-one, subst val-t2p[of v], assumption+,
      rule aadd-pos-poss[of v x v (1_r ± -_a x)], assumption+,
      simp add:value-of-one,
      cut-tac aadd-pos-poss[of v (1_r ± x) v (1_r ± -_a x)],
      simp add:aadd-0-r, rule val-axiom4, assumption+)
apply (subst Ring.ring-distrib2, assumption+, simp add:Ring.ring-l-one,
      subst Ring.ring-distrib1, assumption+, simp add:Ring.ring-r-one,
      subst aGroup.pOp-assocTr43, assumption+,
      rule Ring.ring-tOp-closed, assumption+,
      simp add:aGroup.ag-l-inv1 aGroup.ag-r-zero,
      subst Ring.ring-inv1-2, assumption+, simp, assumption+)
```

```

apply simp
```

```

apply (rule contrapos-pp, simp+,
        frule Ring.ring-tOp-closed[of K x (1r ± -a x)], assumption+,
        simp add:aGroup.ag-pOp-commute[of K 1r],
        frule aGroup.ag-eq-diffzero[THEN sym, of K x ·r (-a x ± 1r) -a 1r],
        assumption+, rule aGroup.ag-mOp-closed, assumption+)
apply (simp add:aGroup.ag-inv-inv[of K 1r],
        frule eq-elems-eq-val[of x ·r (-a x ± 1r) -a 1r v],
        thin-tac x ·r (-a x ± 1r) = -a 1r,
        simp add:val-minus-eq value-of-one,
        simp add:val-t2p,
        frule aadd-pos-poss[of v x v (-a x ± 1r)], assumption+,
        simp)
done

lemma (in Corps) OstrowskiTr9:[valuation K v; x ∈ carrier K; 0 < (v x)] ==>
0 < (v (x ·r ((1r ± x ·r (1r ± -a x))⁻K)))
apply (subgoal-tac 1r ± x ·r (1r ± -a x) ≠ 0)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
        frule aGroup.ag-mOp-closed[of K x], assumption+,
        frule Ring.ring-one[of K],
        frule aGroup.ag-pOp-closed[of K 1r -a x], assumption+,
        subst val-t2p, assumption+,
        cut-tac invf-closed1[of 1r ± x ·r (1r ± -a x)], simp)
apply simp
apply (rule aGroup.ag-pOp-closed, assumption+,
        rule Ring.ring-tOp-closed, assumption+)

apply (subst value-of-inv[of v 1r ± x ·r (1r ± -a x)], assumption+,
        rule aGroup.ag-pOp-closed, assumption+,
        rule Ring.ring-tOp-closed, assumption+,
        frule value-less-eq[THEN sym, of v 1r -a x], assumption+,
        simp add:value-of-one, simp add:val-minus-eq,
        subst value-less-eq[THEN sym, of v 1r x ·r (1r ± -a x)],
        assumption+, rule Ring.ring-tOp-closed, assumption+,
        simp add:value-of-one, subst val-t2p, assumption+,
        subst aadd-commute,
        rule aadd-pos-poss[of v (1r ± -a x) v x],
        simp, assumption, simp add:value-of-one,
        simp add:aadd-0-r)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
        frule Ring.ring-one,
        rule contrapos-pp, simp+,
        frule Ring.ring-tOp-closed[of K x (1r ± -a x)], assumption+,
        rule aGroup.ag-pOp-closed, assumption+,
        rule aGroup.ag-mOp-closed, assumption+,
        frule aGroup.ag-mOp-closed[of K x], assumption+)

apply (simp add:aGroup.ag-pOp-commute[of K 1r],
        frule aGroup.ag-eq-diffzero[THEN sym, of K x ·r (-a x ± 1r) -a 1r],

```

```

simp add:aGroup.ag-pOp-commute,
rule aGroup.ag-mOp-closed, assumption+,
simp add:aGroup.ag-inv-inv,
frule eq-elems-eq-val[of x ·r (-a x ± 1r) -a 1r v],
thin-tac x ·r (-a x ± 1r) = -a 1r,
simp add:val-minus-eq value-of-one,
frule-tac aGroup.ag-pOp-closed[of K -a x 1r], assumption+,
simp add:val-t2p)

apply (simp add:aadd-commute[of v x],
cut-tac aadd-pos-poss[of v (-a x ± 1r) v x], simp,
subst aGroup.ag-pOp-commute, assumption+,
subst value-less-eq[THEN sym, of v 1r -a x], assumption+,
simp add:value-of-one val-minus-eq, simp add:value-of-one)

apply assumption
done

lemma (in Corps) OstrowskiTr10:[valuation K v; x ∈ carrier K;
¬ 0 ≤ v x] ⇒ 0 < (v (x ·r ((1r ± x ·r (1r ± -a x))⁻K)))
apply (frule OstrowskiTr6[of v x], assumption+,
cut-tac invf-closed1[of 1r ± x ·r (1r ± -a x)], simp,
erule conjE, simp add:aneg-le, frule val-neg-nonzero[of v x],
(erule conjE)+, assumption+, erule conjE)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule aGroup.ag-mOp-closed[of K x], assumption+,
frule Ring.ring-one[of K],
frule aGroup.ag-pOp-closed[of K 1r -a x], assumption+,
subst val-t2p, assumption+)
apply (subst value-of-inv[of v 1r ± x ·r (1r ± -a x)],
assumption+, subst aGroup.ag-pOp-commute[of K 1r], assumption+,
rule Ring.ring-tOp-closed, assumption+,
subst value-less-eq[THEN sym, of v
x ·r (1r ± -a x) 1r], assumption+)
apply (rule Ring.ring-tOp-closed, assumption+, simp add:value-of-one,
frule one-plus-x-nonzero[of v -a x], assumption,
simp add:val-minus-eq, erule conjE, simp,
subst val-t2p[of v], assumption+, simp add:aadd-two-negg)

apply (simp add:val-t2p,
frule value-less-eq[THEN sym, of v -a x 1r], assumption+,
simp add:value-of-one, simp add:val-minus-eq,
simp add:val-minus-eq, simp add:aGroup.ag-pOp-commute[of K -a x],
frule val-nonzero-z[of v x], assumption+, erule exE,
simp add:a-zpz aminus, simp add:ant-0[THEN sym] aless-zless,
assumption)
done

lemma (in Corps) Ostrowski-first:vals-nonequiv K (Suc 0) vv
⇒ ∃x∈carrier K. Ostrowski-elem K (Suc 0) vv x

```

```

apply (simp add:vals-nonequiv-def,
      cut-tac Nset-Suc0, (erule conjE)+,
      simp add:valuations-def)
apply (rotate-tac -1,
      frule-tac a = 0 in forall-spec, simp,
      rotate-tac -1,
      drule-tac a = Suc 0 in forall-spec, simp)
apply (drule-tac a = Suc 0 in forall-spec, simp,
      rotate-tac -1,
      drule-tac a = 0 in forall-spec, simp, simp)
apply (frule-tac a = 0 in forall-spec, simp,
      drule-tac a = Suc 0 in forall-spec, simp,
      frule-tac v = vv 0 and v' = vv (Suc 0) in
      nonequiv-ex-Ostrowski-elem, assumption+,
      erule bxE)

apply (erule conjE,
      frule-tac v = vv (Suc 0) and v' = vv 0 in
      nonequiv-ex-Ostrowski-elem, assumption+,
      erule bxE,
      thin-tac ⊢ v-equiv K (vv (Suc 0)) (vv 0),
      thin-tac ⊢ v-equiv K (vv 0) (vv (Suc 0)))

apply (rename-tac s t)
apply (erule conjE,
      frule-tac x = t and v = vv 0 in val-neg-nonzero, assumption+)
apply (simp add:less-ant-def, (erule conjE)+,
      frule-tac x = s and v = vv (Suc 0) in val-neg-nonzero,
      assumption+, unfold less-ant-def)
apply (rule conjI, assumption+)

apply (frule-tac s = s and t = t and v = vv 0 in OstrowskiTr2,
      assumption+, rule ale-neq-less, assumption+)
apply (frule-tac s = s and t = t and v = vv (Suc 0) in OstrowskiTr3,
      assumption+, rule ale-neq-less, assumption+)
apply (subgoal-tac t ⋅r ((s ± t)^-K) ∈ carrier K,
      simp only:Ostrowski-elem-def,
      simp only: nset-m-m[of Suc 0], blast)

apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
      rule Ring.ring-tOp-closed, assumption+,
      frule-tac s = s and t = t and v = vv 0 in OstrowskiTr1,
      assumption+, rule ale-neq-less, assumption+,
      frule-tac x = s and y = t in aGroup.ag-pOp-closed[of K], assumption+)
apply (cut-tac x = s ± t in invf-closed, blast)
apply assumption
done

```

```

lemma (in Corps) Ostrowski: $\forall vv. \text{vals-nonequiv } K (\text{Suc } n) vv \longrightarrow$ 
 $(\exists x \in \text{carrier } K. \text{Ostrowski-elem } K (\text{Suc } n) vv x)$ 
apply (induct-tac n,
rule allI, rule impI, simp add:Ostrowski-first)

apply (rule allI, rule impI,
frule-tac n = n and vv = vv in restrict-vals-nonequiv1,
frule-tac n = n and vv = vv in restrict-vals-nonequiv2,
frule-tac a = compose {h. h ≤ Suc n} vv (skip 1) in forall-spec,
assumption,
drule-tac a = compose {h. h ≤ Suc n} vv (skip 2) in forall-spec,
assumption+, (erule bexE)+)
apply (rename-tac n vv s t,
cut-tac field-is-ring, frule Ring.ring-is-ag[of K])

apply (frule-tac x = s and y = t in Ring.ring-tOp-closed[of K], assumption+,
case-tac 0 ≤ vv (Suc 0) s ∧ 0 ≤ vv (Suc (Suc 0)) t,
frule-tac vv = vv and s = s and t = t in OstrowskiTr5, assumption+)
apply blast

apply (simp,
case-tac 0 ≤ (vv (Suc 0) s), simp,
frule-tac n = Suc (Suc n) and m = Suc (Suc 0) and vv = vv in
vals-nonequiv-valuation,
simp,
frule-tac v = vv (Suc (Suc 0)) and x = t in OstrowskiTr6,
assumption+,
frule-tac x = 1_r ± t ·_r (1_r ± -_a t) in invf-closed1,
frule-tac x = t and y = (1_r ± t ·_r (1_r ± -_a t))^-K in
Ring.ring-tOp-closed, assumption+, simp)
apply (subgoal-tac Ostrowski-elem K (Suc (Suc n)) vv
(t ·_r ((1_r ± t ·_r (1_r ± (-_a t)))^-K)),
blast)
apply (subst Ostrowski-elem-def,
rule conjI,
thin-tac Ostrowski-elem K (Suc n)
(compose {h. h ≤ Suc n} vv (skip (Suc 0))) s,
thin-tac vals-nonequiv K (Suc n)
(compose {h. h ≤ Suc n} vv (skip (Suc 0))),
thin-tac vals-nonequiv K (Suc n) (compose {h. h ≤ Suc n} vv (skip 2)),
thin-tac 0 ≤ (vv (Suc 0) s),
frule-tac n = Suc (Suc n) and vv = vv and m = 0 in
vals-nonequiv-valuation, simp,
rule-tac v = vv 0 and x = t in
OstrowskiTr8, assumption+)

```

```

apply (simp add:Ostrowski-elem-def, (erule conjE)+,
  thin-tac  $\forall j \in nset (Suc 0) (Suc n).$ 
     $0 < (\text{compose } \{h. h \leq (Suc n)\} vv (\text{skip } 2) j t),$ 
    simp add:compose-def skip-def,
    rule ballI,
    thin-tac  $0 \leq (vv (Suc 0) s),$ 
    thin-tac Ostrowski-elem  $K (Suc n)$ 
       $(\text{compose } \{h. h \leq (Suc n)\} vv (\text{skip } (Suc 0))) s,$ 
    frule-tac  $n = Suc (Suc n)$  and  $vv = vv$  and  $m = j$  in
      vals-nonequiv-valuation,
    simp add:nset-def, simp add:Ostrowski-elem-def, (erule conjE)+)

apply (case-tac  $j = Suc 0$ , simp,
  drule-tac  $x = Suc 0$  in bspec,
  simp add:nset-def,
  simp add:compose-def skip-def,
  rule-tac  $v = vv (Suc 0)$  and  $x = t$  in
    OstrowskiTr9, assumption+
  frule-tac  $j = j$  and  $n = n$  in nset-Tr51, assumption+
  drule-tac  $x = j - Suc 0$  in bspec, assumption+
  simp add:compose-def skip-def)

apply (case-tac  $j = Suc (Suc 0)$ , simp) apply (
  rule-tac  $v = vv (Suc (Suc 0))$  and  $x = t$  in OstrowskiTr10,
  assumption+) apply (
    subgoal-tac  $\neg j - Suc 0 \leq Suc 0$ , simp add:nset-def) apply (
      rule-tac  $v = vv j$  and  $x = t$  in
        OstrowskiTr9) apply (simp add:nset-def, assumption+)
apply (simp add:nset-def, (erule conjE)+, rule nset-Tr52, assumption+
  thin-tac vals-nonequiv  $K (Suc n)$ 
     $(\text{compose } \{h. h \leq (Suc n)\} vv (\text{skip } (Suc 0))),$ 
  thin-tac vals-nonequiv  $K (Suc n)$ 
     $(\text{compose } \{h. h \leq (Suc n)\} vv (\text{skip } 2)),$ 
  thin-tac Ostrowski-elem  $K (Suc n)$ 
     $(\text{compose } \{h. h \leq (Suc n)\} vv (\text{skip } 2)) t)$ 
apply (subgoal-tac  $s \cdot_r ((1_r \pm s \cdot_r (1_r \pm -_a s))^{-K}) \in \text{carrier } K$ ,
  subgoal-tac Ostrowski-elem  $K (Suc (Suc n)) vv$ 
     $(s \cdot_r ((1_r \pm s \cdot_r (1_r \pm -_a s))^{-K})), blast$ )
prefer 2
apply (frule-tac  $n = Suc (Suc n)$  and  $m = Suc 0$  and  $vv = vv$  in
  vals-nonequiv-valuation, simp,
  frule-tac  $v = vv (Suc 0)$  and  $x = s$  in OstrowskiTr6, assumption+
  rule Ring.ring-tOp-closed, assumption+
  frule-tac  $x = 1_r \pm s \cdot_r (1_r \pm -_a s)$  in invf-closed1, simp,
  simp add:Ostrowski-elem-def)
apply (rule conjI)
apply (rule-tac  $v = vv 0$  and  $x = s$  in OstrowskiTr8,
  simp add:vals-nonequiv-valuation, assumption+)
apply (

```

```

thin-tac vals-nonequiv K (Suc (Suc n)) vv,
(erule conjE)+,
thin-tac  $\forall j \in nset (Suc 0) (Suc n).$ 
 $0 < (\text{compose } \{h. h \leq (\text{Suc } n)\} vv (\text{skip } (\text{Suc } 0)) j s),$ 
simp add:compose-def skip-def, rule ballI)

```

```

apply (case-tac  $j = \text{Suc } 0$ , simp,
rule-tac  $v = vv (\text{Suc } 0)$  and  $x = s$  in OstrowskiTr10,
simp add:vals-nonequiv-valuation, assumption+,
rule-tac  $v = vv j$  and  $x = s$  in OstrowskiTr9,
simp add:vals-nonequiv-valuation nset-def, assumption,
(erule conjE)+, simp add:compose-def skip-def,
frule-tac  $j = j$  in nset-Tr51, assumption+,
drule-tac  $x = j - \text{Suc } 0$  in bspec, assumption+)
apply (simp add:nset-def)
done

```

```

lemma (in Corps) val-1-nonzero:[valuation K v;  $x \in \text{carrier } K; v x = 1$ ]  $\implies$ 
 $x \neq 0$ 
apply (rule contrapos-pp, simp+,
simp add:value-of-zero,
rotate-tac 3, drule sym, simp only:ant-1[THEN sym],
simp del:ant-1)
done

```

```

lemma (in Corps) Approximation1-5Tr1:[vals-nonequiv K (Suc n) vv;
n-val K (vv 0) = vv 0;  $a \in \text{carrier } K; vv 0 a = 1; x \in \text{carrier } K;$ 
Ostrowski-elem K (Suc n) vv x]  $\implies$ 
 $\forall m. 2 \leq m \longrightarrow vv 0 ((1_r \pm -_a x)^{\wedge K m} \pm a \cdot_r (x^{\wedge K m})) = 1$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule Ring.ring-one[of K],
rule allI, rule impI,
frule vals-nonequiv-valuation[of Suc n vv 0],
simp,
simp add:Ostrowski-elem-def, frule conjunct1, fold Ostrowski-elem-def,
frule val-1-nonzero[of vv 0 a], assumption+)
apply (frule vals-nonequiv-valuation[of Suc n vv 0], simp,
frule val nonzero noninf[of vv 0 a], assumption+,
frule val-unit-cond[of vv 0 x], assumption+,
frule-tac  $n = m$  in Ring,npClose[of K x], assumption+,
frule aGroup.ag-mOp-closed[of K x], assumption+,
frule aGroup.ag-pOp-closed[of K 1_r -_a x], assumption+)
apply (subgoal-tac  $0 < m$ ,
frule-tac  $x = a \cdot_r (x^{\wedge K m})$  and  $y = (1_r \pm -_a x)^{\wedge K m}$  in
value-less-eq[of vv 0],
rule Ring.ring-tOp-closed, assumption+,
rule Ring,npClose, assumption+, simp add: val-t2p,
frule value-zero-nonzero[of vv 0 x], assumption+,
simp add:val-exp-ring[THEN sym], simp add:asprod-n-0 aadd-0-r)

```

```

apply (case-tac  $x = 1_r K$ , simp add:aGroup.ag-r-inv1,
      frule-tac  $n = m$  in Ring,npZero-sub[of  $K$ ], simp,
      simp add:value-of-zero)
apply (cut-tac inf-ge-any[of 1], simp add: less-le)
apply (rotate-tac -1, drule not-sym,
      frule aGroup.ag-neq-diffnonzero[of  $K$   $1_r x$ ],
      simp add:Ring.ring-one[of  $K$ ], assumption+, simp,
      simp add:val-exp-ring[THEN sym],
      cut-tac  $n1 = m$  in of-nat-0-less-iff[THEN sym])
apply (cut-tac  $a = 0 < m$  and  $b = 0 < int m$  in a-b-exchange, simp,
      assumption)
apply (thin-tac  $(0 < m) = (0 < int m)$ ,
      frule val-nonzero-z[of vv 0  $1_r \pm -_a x$ ], assumption+,
      erule exE, simp, simp add:asprod-amult a-z-z,
      simp only:ant-1[THEN sym], simp only:aless-zless, simp add:ge2-zmult-pos)

apply (subst aGroup.ag-pOp-commute[of  $K$ ], assumption+,
      rule Ring,npClose, assumption+, rule Ring.ring-tOp-closed[of  $K$ ],
      assumption+,
      rotate-tac -1, drule sym, simp,
      thin-tac vv 0  $(a \cdot_r x \cdot_K^m \pm (1_r \pm -_a x) \cdot_K^m) = vv 0 (a \cdot_r x \cdot_K^m)$ )
apply (simp add:val-t2p,
      frule value-zero-nonzero[of vv 0  $x$ ], assumption+,
      simp add:val-exp-ring[THEN sym], simp add:asprod-n-0,
      simp add:aadd-0-r,
      cut-tac  $z = m$  in less-le-trans[of 0 2], simp, assumption+)
done

lemma (in Corps) Approximation1-5Tr3:[vals-nonequiv  $K$  ( $Suc n$ ) vv;
 $x \in carrier K$ ; Ostrowski-elem  $K$  ( $Suc n$ ) vv  $x$ ;  $j \in nset (Suc 0)$  ( $Suc n$ )]
 $\implies vv j ((1_r \pm -_a x) \cdot_K^m) = 0$ 
apply (frule Ostrowski-elem-not-one[of  $n$  vv  $x$ ], assumption+,
      cut-tac field-is-ring, frule Ring.ring-is-ag[of  $K$ ],
      frule Ring.ring-one[of  $K$ ],
      frule aGroup.ag-pOp-closed[of  $K$   $1_r -_a x$ ], assumption+)
apply (simp add:aGroup.ag-mOp-closed, simp add:nset-def,
      frule-tac  $m = j$  in vals-nonequiv-valuation[of  $Suc n$  vv],
      simp,
      frule-tac  $v1 = vv j$  and  $x1 = 1_r \pm -_a x$  and  $n1 = m$  in
      val-exp-ring[THEN sym], assumption+)

apply (frule-tac  $v = vv j$  and  $x = 1_r$  and  $y = -_a x$  in
      value-less-eq, assumption+, simp add:aGroup.ag-mOp-closed)
apply (simp add:value-of-one, simp add:val-minus-eq,
      simp add:Ostrowski-elem-def nset-def)
apply (simp add:value-of-one, rotate-tac -1, drule sym,
      simp add:asprod-n-0)
done

```

lemma (in Corps) Approximation1-5Tr4: $\llbracket \text{vals-nonequiv } K (\text{Suc } n) vv; aa \in \text{carrier } K; x \in \text{carrier } K; \text{Ostrowski-elem } K (\text{Suc } n) vv x; j \leq (\text{Suc } n) \rrbracket \implies vv j (aa \cdot_r (x^K m)) = vv j aa + (\text{int } m) *_a (vv j x)$

apply (frule Ostrowski-elem-nonzero[of n vv x],
assumption+,
cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (frule-tac $m = j$ in vals-nonequiv-valuation[of $\text{Suc } n$ vv], *assumption*)
apply (subst val-t2p[of $vv j$], *assumption+*,
rule Ring.npClose, *assumption+*,
cut-tac field-is-idom,
*frule-tac $v1 = vv j$ and $x1 = x$ and $n1 = m$ in
 val-exp-ring[THEN sym]*, *assumption+*, *simp*)
done

lemma (in Corps) Approximation1-5Tr5: $\llbracket \text{vals-nonequiv } K (\text{Suc } n) vv; a \in \text{carrier } K; a \neq \mathbf{0}; x \in \text{carrier } K; \text{Ostrowski-elem } K (\text{Suc } n) vv x; j \in \text{nset } (\text{Suc } 0) (\text{Suc } n) \rrbracket \implies \exists l. \forall m. l < m \longrightarrow 0 < (vv j (a \cdot_r (x^K m)))$

apply (frule Ostrowski-elem-nonzero[of n vv x], *assumption+*,
*subgoal-tac $\forall n. vv j (a \cdot_r (x^K n)) = vv j a + (\text{int } n) *_a (vv j x)$,*
simp,
*thin-tac $\forall n. vv j (a \cdot_r x^K n) = vv j a + \text{int } n *_a vv j x$*)
prefer 2
apply (rule allI, rule Approximation1-5Tr4[of - $vv a x j$],
assumption+, *simp add:nset-def*)
apply (frule-tac $m = j$ in vals-nonequiv-valuation[of $\text{Suc } n$ vv],
simp add:nset-def,
*frule val-nonzero-z[of $vv j a$], *assumption+*, erule exE,*
simp add:Ostrowski-elem-def,
frule conjunct2, fold Ostrowski-elem-def,
*drule-tac $x = j$ in bspec, *assumption*)
apply (frule Ostrowski-elem-nonzero[of n vv x], *assumption+*,
*frule val-nonzero-z[of $vv j x$], *assumption+*, erule exE, *simp*,*
frule-tac $a = za$ and $x = z$ in zmult-pos-bignumTr,
simp add:asprod-amult a-z-z a-zpz)
done*

lemma (in Corps) Approximation1-5Tr6: $\llbracket \text{vals-nonequiv } K (\text{Suc } n) vv; a \in \text{carrier } K; a \neq \mathbf{0}; x \in \text{carrier } K; \text{Ostrowski-elem } K (\text{Suc } n) vv x; j \in \text{nset } (\text{Suc } 0) (\text{Suc } n) \rrbracket \implies \exists l. \forall m. l < m \longrightarrow vv j ((1_r \pm -_a x)^K m \pm a \cdot_r (x^K m)) = 0$

apply (frule vals-nonequiv-valuation[of $\text{Suc } n$ vv j],
simp add:nset-def,
*frule Approximation1-5Tr5[of n vv a x j],
 i*assumption+*, erule exE,
 cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
 subgoal-tac $\forall m. l < m \longrightarrow vv j ((1_r \pm -_a x)^K m \pm a \cdot_r (x^K m)) = vv j ((1_r \pm -_a x)^K m)$)*

```

apply (simp add:Approximation1-5Tr3, blast)
apply (rule allI, rule impI,
       drule-tac a = m in forall-spec, assumption,
       frule-tac x = (1_r ± -_a x) ^K m and y = a ·_r (x ^K m) in
         value-less-eq[of vv j],
       rule Ring,npClose, assumption+,
       rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,
       simp add:aGroup.ag-mOp-closed)
apply (rule Ring.ring-tOp-closed, assumption+,
       rule Ring,npClose, assumption+,
       simp add:Approximation1-5Tr3,
       frule sym, assumption)
done

lemma (in Corps) Approximation1-5Tr7:[a ∈ carrier K; vv 0 a = 1;
x ∈ carrier K] ==>
  vals-nonequiv K (Suc n) vv ∧ Ostrowski-elem K (Suc n) vv x →
  (∃l. ∀m. l < m → (∀j∈nset (Suc 0) (Suc n).
    (vv j ((1_r ± -_a x) ^K m ± a ·_r (x ^K m)) = 0)))
apply (induct-tac n,
       rule impI, erule conjE, simp add:nset-m-m[of Suc 0],
       frule vals-nonequiv-valuation[of Suc 0 vv Suc 0], simp,
       frule Approximation1-5Tr6[of 0 vv a x Suc 0], assumption+)
apply (frule vals-nonequiv-valuation[of Suc 0 vv 0], simp,
       frule val-1-nonzero[of vv 0 a], assumption+, simp add:nset-def,
       assumption)

apply (rule impI, erule conjE,
       frule-tac n = n in restrict-vals-nonequiv[of - vv],
       frule-tac n = n in restrict-Ostrowski-elem[of x - vv],
         assumption, simp,
       erule exE,
       frule-tac n = Suc n and j = Suc (Suc n) in Approximation1-5Tr6
         [of - vv a x], assumption+,
       frule-tac n = Suc (Suc n) in vals-nonequiv-valuation[of - vv
         0], simp,
       rule val-1-nonzero[of vv 0 a], assumption+,
       simp add:nset-def)
apply (erule exE,
       subgoal-tac ∀m. (max l la) < m → (∀j∈nset (Suc 0) (Suc (Suc n)).
        vv j ((1_r ± -_a x) ^K m ± a ·_r (x ^K m)) = 0),
       blast,
       simp add:nset-Suc)
done

lemma (in Corps) Approximation1-5P:[vals-nonequiv K (Suc n) vv;
n-val K (vv 0) = vv 0] ==>
  ∃x∈carrier K. ((vv 0 x = 1) ∧ (∀j∈nset (Suc 0) (Suc n). (vv j x) = 0))
apply (frule vals-nonequiv-valuation[of Suc n vv 0], simp) apply (

```

```

frule n-val-surj[of vv 0], erule bxE) apply (
  rename-tac aa) apply (
    cut-tac n = n in Ostrowski) apply (
      drule-tac a = vv in forall-spec[of vals-nonequiv K (Suc n)], simp)
apply (
  erule bxE,
  frule-tac a = aa and x = x in Approximation1-5Tr1[of n vv],
  assumption+,
  simp, assumption+)
apply (frule-tac a = aa and x = x in Approximation1-5Tr7[of - vv - n],
  simp, assumption,
  simp, erule exE,
  cut-tac b = Suc l in max.cobounded1[of 2],
  cut-tac b = Suc l in max.cobounded2[of - 2],
  cut-tac n = l in lessI,
  frule-tac x = l and y = Suc l and z = max 2 (Suc l) in
    less-le-trans, assumption+,
  thin-tac Suc l ≤ max 2 (Suc l), thin-tac l < Suc l,
  drule-tac a = max 2 (Suc l) in forall-spec, simp,
  drule-tac a = max 2 (Suc l) in forall-spec, assumption)
apply (subgoal-tac (1_r ± -_a x) ^K (max 2 (Suc l)) ± aa ·_r (x ^K (max 2 (Suc l))) ∈
  carrier K,
  blast,
  cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
  rule aGroup.ag-pOp-closed, assumption+, rule Ring,npClose, assumption+,
  rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,
  simp add:aGroup.ag-mOp-closed, rule Ring.ring-tOp-closed, assumption+,
  rule Ring,npClose, assumption+)
done

lemma K-gamma-hom:k ≤ n ⇒ ∀ j ≤ n. (λl. γ_k l) j ∈ Zset
apply (simp add:Zset-def)
done

lemma transpos-eq:(τ_0 0) k = k
by (simp add:transpos-def)

lemma (in Corps) transpos-vals-nonequiv:[vals-nonequiv K (Suc n) vv;
  j ≤ (Suc n)] ⇒ vals-nonequiv K (Suc n) (vv ∘ (τ_0 j))
apply (simp add:vals-nonequiv-def)
apply (frule conjunct1, fold vals-nonequiv-def)
apply (simp add:valuations-def, rule conjI)
apply (rule allI, rule impI)
apply (case-tac ja = 0, simp,
  case-tac j = 0, simp add:transpos-eq)
apply (subst transpos-ij-1[of 0 Suc n j], simp, assumption+,
  rule not-sym, assumption, simp)

```

```

apply (case-tac ja = j, simp)
apply (subst transpos-ij-2[of 0 Suc n j], simp, assumption, simp,
       simp add:vals-nonequiv-valuation)
apply (case-tac j = 0, simp add:transpos-eq)
apply (cut-tac x = ja in transpos-id-1[of 0 Suc n j], simp, assumption+,
       rule not-sym, assumption+)
apply (simp add:vals-nonequiv-valuation,
       (rule allI, rule impI)+, rule impI)
apply (case-tac j = 0, simp add:transpos-eq,
       simp add:vals-nonequiv-def,
       cut-tac transpos-inj[of 0 Suc n j], simp)
apply (frule-tac x = ja and y = l in injective[of transpos 0 j
       {j. j ≤ (Suc n)}], simp, simp, assumption+)
apply (cut-tac l = ja in transpos-mem[of 0 Suc n j], simp, assumption+,
       simp, assumption,
       cut-tac l = l in transpos-mem[of 0 Suc n j], simp, assumption+,
       simp, assumption)
apply (simp add:vals-nonequiv-def,
       simp, assumption, rule not-sym, assumption)
done

```

definition

```

Ostrowski-base :: [-, nat ⇒ 'b ⇒ ant, nat] ⇒ (nat ⇒ 'b)
((Ω_ _ _) [90,90,91]90) where
Ostrowski-base K vv n = (λj∈{h. h ≤ n}. (SOME x. x ∈ carrier K ∧
(Ostrowski-elem K n (vv ∘ (τ₀ j)) x)))

```

definition

```

App-base :: [-, nat ⇒ 'b ⇒ ant, nat] ⇒ (nat ⇒ 'b) where
App-base K vv n = (λj∈{h. h ≤ n}. (SOME x. x ∈ carrier K ∧ (((vv ∘ τ₀ j) 0 x
= 1) ∧ (∀k∈nset (Suc 0) n. ((vv ∘ τ₀ j) k x) = 0))))

```

lemma (in Corps) Ostrowski-base-hom:vals-nonequiv K (Suc n) vv ==>
 $\text{Ostrowski-base } K \text{ vv } (\text{Suc } n) \in \{h. h \leq (\text{Suc } n)\} \rightarrow \text{carrier } K$
apply (rule Pi-I, rename-tac l,
 $\text{simp add:Ostrowski-base-def},$
 $\text{frule-tac } j = l \text{ in transpos-vals-nonequiv[of } n \text{ vv]}, \text{ simp},$
 $\text{cut-tac Ostrowski[of } n\text{]}$
apply (drule-tac a = vv ∘ τ₀ l in forall-spec, simp,
 $\text{rule someI2-ex, blast, simp})$
done

lemma (in Corps) Ostrowski-base-mem:vals-nonequiv K (Suc n) vv ==>
 $\forall j \leq (\text{Suc } n). \text{Ostrowski-base } K \text{ vv } (\text{Suc } n) j \in \text{carrier } K$
by (rule allI, rule impI,
 $\text{frule Ostrowski-base-hom[of } n \text{ vv]},$
 $\text{simp add:funcset-mem del:Pi-I'})$

```

lemma (in Corps) Ostrowski-base-mem-1: vals-nonequiv K (Suc n) vv;
  j ≤ (Suc n) ] ==> Ostrowski-base K vv (Suc n) j ∈ carrier K
by (simp add:Ostrowski-base-mem)

lemma (in Corps) Ostrowski-base-nonzero: vals-nonequiv K (Suc n) vv;
  j ≤ Suc n ] ==> (Ω_K vv (Suc n)) j ≠ 0
apply (simp add:Ostrowski-base-def,
  frule-tac j = j in transpos-vals-nonequiv[of n vv],
  assumption+,
  cut-tac Ostrowski[of n],
  drule-tac a = vv ∘ τ₀ j in forall-spec, assumption,
  rule someI2-ex, blast)
apply (thin-tac ∃x∈carrier K. Ostrowski-elem K (Suc n) (vv ∘ τ₀ j) x,
  erule conjE)
apply (rule-tac vv = vv ∘ τ₀ j and x = x in Ostrowski-elem-nonzero[of n],
  assumption+)
done

lemma (in Corps) Ostrowski-base-pos: vals-nonequiv K (Suc n) vv;
  j ≤ Suc n; ja ≤ Suc n; ja ≠ j ] ==> 0 < ((vv j) ((Ω_K vv (Suc n)) ja))
apply (simp add:Ostrowski-base-def,
  frule-tac j = ja in transpos-vals-nonequiv[of n vv],
  assumption+,
  cut-tac Ostrowski[of n],
  drule-tac a = vv ∘ τ₀ ja in forall-spec, assumption+)
apply (rule someI2-ex, blast,
  thin-tac ∃x∈carrier K. Ostrowski-elem K (Suc n) (vv ∘ τ₀ ja) x,
  simp add:Ostrowski-elem-def, (erule conjE)+)
apply (case-tac ja = 0, simp, cut-tac transpos-eq[of j],
  simp add:nset-def, frule Suc-leI[of 0 j],
  frule-tac a = j in forall-spec, simp, simp)
apply (case-tac j = 0, simp,
  frule-tac x = ja in bspec, simp add:nset-def,
  cut-tac transpos-ij-2[of 0 Suc n ja], simp, simp+)
apply (frule-tac x = j in bspec, simp add:nset-def,
  cut-tac transpos-id[of 0 Suc n ja j], simp+)
done

lemma (in Corps) App-base-hom: vals-nonequiv K (Suc n) vv;
  ∀j ≤ (Suc n). n-val K (vv j) = vv j ] ==>
  ∀j ≤ (Suc n). App-base K vv (Suc n) j ∈ carrier K
apply (rule allI, rule impI,
  rename-tac k,
  subst App-base-def)
apply (case-tac k = 0, simp, simp add:transpos-eq,
  frule Approximation1-5P[of n vv], simp,
  rule someI2-ex, blast, simp)
apply (frule-tac j = k in transpos-vals-nonequiv[of n vv],
  simp add:nset-def,

```

```

frule-tac  $vv = vv \circ \tau_0 k$  in Approximation1-5P[of  $n$ ])
apply (simp add:cmp-def, subst transpos-ij-1[of 0 Suc  $n$ ], simp+,
       subst transpos-ij-1[of 0 Suc  $n$   $k$ ], simp+)
apply (rule someI2-ex, blast, simp)
done

lemma (in Corps) Approximation1-5P2:[vals-nonequiv  $K$  (Suc  $n$ )  $vv;$ 
   $\forall l \in \{h. h \leq \text{Suc } n\}. n\text{-val } K (vv l) = vv l; i \leq \text{Suc } n; j \leq \text{Suc } n \] \implies vv i (\text{App-base } K vv (\text{Suc } n) j) = \delta_{i j}$ 
apply (simp add:App-base-def)
apply (case-tac  $j = 0$ , simp add:transpos-eq,
       rule someI2-ex,
       frule Approximation1-5P[of  $n$   $vv$ ], simp , blast,
       simp add:Kronecker-delta-def, rule impI, (erule conjE)+,
       frule-tac  $x = i$  in bspec, simp add:nset-def, assumption)

apply (frule-tac  $j = j$  in transpos-vals-nonequiv[of  $n$   $vv$ ], simp,
       frule Approximation1-5P[of  $n$   $vv \circ \tau_0 j$ ],
       simp add:cmp-def, simp add:transpos-ij-1[of 0 Suc  $n$   $j$ ])

apply (simp add:cmp-def,
       case-tac  $i = 0$ , simp add:transpos-eq,
       simp add:transpos-ij-1, simp add:Kronecker-delta-def,
       rule someI2-ex, blast,
       thin-tac  $\exists x \in \text{carrier } K.$ 
            $vv j x = 1 \wedge (\forall ja \in \text{nset } (\text{Suc } 0) (\text{Suc } n). vv ((\tau_0 j) ja) x = 0),$ 
           (erule conjE)+,
       drule-tac  $x = j$  in bspec, simp add:nset-def,
       simp add:transpos-ij-2)

apply (simp add:Kronecker-delta-def,
       case-tac  $i = j$ , simp add:transpos-ij-1, rule someI2-ex, blast, simp)

apply (simp, rule someI2-ex, blast,
       thin-tac  $\exists x \in \text{carrier } K. vv ((\tau_0 j) 0) x = 1 \wedge$ 
            $(\forall ja \in \text{nset } (\text{Suc } 0) (\text{Suc } n). vv ((\tau_0 j) ja) x = 0),$ 
           (erule conjE)+,
       drule-tac  $x = i$  in bspec, simp add:nset-def,
       cut-tac transpos-id[of 0 Suc  $n$   $j$   $i$ ], simp+)
done

lemma (in Corps) Approximation1-5:[vals-nonequiv  $K$  (Suc  $n$ )  $vv;$ 
   $\forall j \leq (\text{Suc } n). n\text{-val } K (vv j) = vv j \] \implies$ 
   $\exists x. (\forall j \leq (\text{Suc } n). x j \in \text{carrier } K) \wedge (\forall i \leq (\text{Suc } n). \forall j \leq (\text{Suc } n).$ 
     $((vv i) (x j) = \delta_{i j}))$ 
apply (frule App-base-hom[of  $n$   $vv$ ], rule allI, simp)
apply (subgoal-tac  $(\forall i \leq (\text{Suc } n). \forall j \leq (\text{Suc } n).$ 

```

```


$$(vv i) ((App\text{-}base K vv (Suc n)) j) = (\delta_{i j}),$$

blast)
apply (rule allI, rule impI) +
apply (rule Approximation1\text{--}5P2, assumption+, simp+)
done

lemma (in Corps) Ostrowski-baseTr0:[vals\text{-}nonequiv K (Suc n) vv; l \leq (Suc n)]

$$\implies 0 < ((vv l) (1_r \pm -_a (Ostrowski\text{-}base K vv (Suc n) l))) \wedge$$


$$(\forall m \in \{h. h \leq (Suc n)\} - \{l\}. 0 < ((vv m) (Ostrowski\text{-}base K vv (Suc n) l)))$$

apply (simp add:Ostrowski-base-def,
frule-tac  $j = l$  in transpos-vals-nonequiv[of n vv], assumption,
cut-tac Ostrowski[of n],
drule-tac  $a = vv \circ \tau_0 l$  in forall-spec, assumption)
apply (erule bexE,
unfold Ostrowski-elem-def, frule conjunct1,
fold Ostrowski-elem-def,
rule conjI, simp add:Ostrowski-elem-def)
apply (case-tac  $l = 0$ , simp, simp add:transpos-eq,
rule someI2-ex, blast, simp,
simp add:transpos-ij-1,
rule someI2-ex, blast, simp)

apply (simp add:Ostrowski-elem-def,
case-tac  $l = 0$ , simp, simp add:transpos-eq,
rule someI2-ex, blast,
thin-tac  $0 < vv 0 (1_r \pm -_a x) \wedge$ 
 $(\forall j \in nset (Suc 0) (Suc n). 0 < vv j x),$ 
rule ballI, simp add:nset-def)

apply (rule ballI, erule conjE,
rule someI2-ex, blast,
thin-tac  $\forall j \in nset (Suc 0) (Suc n). 0 < vv ((\tau_0 l) j) x,$ 
(erule conjE)+)

apply (case-tac  $m = 0$ , simp,
drule-tac  $x = l$  in bspec, simp add:nset-def,
simp add:transpos-ij-2,
drule-tac  $x = m$  in bspec, simp add:nset-def,
simp add:transpos-id)
done

lemma (in Corps) Ostrowski-baseTr1:[vals\text{-}nonequiv K (Suc n) vv; l \leq (Suc n)]

$$\implies 0 < ((vv l) (1_r \pm -_a (Ostrowski\text{-}base K vv (Suc n) l)))$$

by (simp add:Ostrowski-baseTr0)

lemma (in Corps) Ostrowski-baseTr2:[vals\text{-}nonequiv K (Suc n) vv;
 $l \leq (Suc n); m \leq (Suc n); l \neq m] \implies$ 
 $0 < ((vv m) (Ostrowski\text{-}base K vv (Suc n) l))$ 
apply (frule Ostrowski-baseTr0[of n vv l], assumption+)

```

```

apply simp
done

lemma Nset-have-two: $j \in \{h. h \leq (\text{Suc } n)\} \implies \exists m \in \{h. h \leq (\text{Suc } n)\}. j \neq m$ 
apply (rule contrapos-pp, simp+,
       case-tac  $j = \text{Suc } n$ , simp,
       drule-tac  $a = 0$  in forall-spec, simp, arith)
apply (drule-tac  $a = \text{Suc } n$  in forall-spec, simp, simp)
done

lemma (in Corps) Ostrowski-base-npow-not-one: $\llbracket 0 < N; j \leq \text{Suc } n;$ 
vals-nonequiv  $K (\text{Suc } n) vv \rrbracket \implies$ 
 $1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) j^{\sim K N}) \neq \mathbf{0}$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       rule contrapos-pp, simp+,
       frule Ostrowski-base-mem-1[of n vv j], assumption,
       frule Ring.npClose[of K  $(\Omega_K vv (\text{Suc } n)) j N$ ], assumption+,
       frule Ring.ring-one[of K],
       frule aGroup.ag-mOp-closed[of K  $(\Omega_K vv (\text{Suc } n)) j^{\sim K N}$ ], assumption+,
       frule aGroup.ag-pOp-closed[of K  $1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) j^{\sim K N})$ ],
       assumption+)
apply (frule aGroup.ag-pOp-add-r[of K  $1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) j^{\sim K N}) \mathbf{0}$ 
                                          $(\Omega_K vv (\text{Suc } n)) j^{\sim K N}$ ], assumption+,
       simp add:aGroup.ag-inc-zero, assumption+,
       thin-tac  $1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) j^{\sim K N}) = \mathbf{0}$ )
apply (simp add:aGroup.ag-pOp-assoc[of K  $1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) j^{\sim K N})$ ])
apply (simp add:aGroup.ag-l-inv1, simp add:aGroup.ag-r-zero aGroup.ag-l-zero)
apply (subgoal-tac  $\forall m \leq (\text{Suc } n). (j \neq m \longrightarrow$ 
           $0 < (vv m ((\Omega_K vv (\text{Suc } n)) j)))$ )
apply (cut-tac Nset-have-two[of j n],
       erule bxE, drule-tac  $a = m$  in forall-spec, simp,
       thin-tac  $(\Omega_K vv (\text{Suc } n)) j^{\sim K N} \pm -_a ((\Omega_K vv (\text{Suc } n)) j^{\sim K N}) \in \text{carrier } K$ ,
       frule-tac  $f = vv m$  in eq-elems-eq-val[of  $1_r (\Omega_K vv (\text{Suc } n)) j^{\sim K N}$ ],
       thin-tac  $1_r = (\Omega_K vv (\text{Suc } n)) j^{\sim K N}$ , simp)
apply (frule-tac  $m = m$  in vals-nonequiv-valuation[of Suc n vv],
       assumption+,
       frule-tac  $v1 = vv m$  and  $n1 = N$  in val-exp-ring[THEN sym,
                                         of  $- (\Omega_K vv (\text{Suc } n)) j$ ], assumption+,
                                         simp add:Ostrowski-base-nonzero, simp, simp add:value-of-one)
apply (subgoal-tac  $\text{int } N \neq 0$ ,
       frule-tac  $x = vv m ((\Omega_K vv (\text{Suc } n)) j)$  in asprod-0[of int N],
       assumption, simp add:less-ant-def, simp, simp,
       rule allI, rule impI, rule impI,
       rule Ostrowski-baseTr2, assumption+)
done

```

abbreviation

CHOOSE :: [nat, nat] \Rightarrow nat ((*C*_.) [80, 81]80) **where**

nC_i == *n choose i*

lemma (**in** *Ring*) *expansion-of-sum1*:*x* \in carrier *R* \Rightarrow

$$(1_r \pm x)^{\sim R} n = nsum R (\lambda i. nC_i \times_R x^{\sim R} i) n$$

apply (*cut-tac ring-one*, *frule npeSum2*[of *1_r x n*], *assumption+*,

simp add:npOne, subgoal-tac $\forall (j::nat). (x^{\sim R} j) \in$ carrier *R*)

apply (*simp add:ring-l-one, rule allI, simp add:npClose*)

done

lemma (**in** *Ring*) *tail-of-expansion*:*x* \in carrier *R* \Rightarrow $(1_r \pm x)^{\sim R} (Suc n) =$

$$(nsum R (\lambda i. ((Suc n)C_{(Suc i)} \times_R x^{\sim R} (Suc i))) n) \pm 1_r$$

apply (*cut-tac ring-is-ag*)

apply (*frule expansion-of-sum1*[of *x Suc n*],

simp del:nsum-suc binomial-Suc-Suc npow-suc,

thin-tac $(1_r \pm x)^{\sim R} (Suc n) = \Sigma_e R (\lambda i. (Suc n)C_i \times_R x^{\sim R} i) (Suc n))$

apply (*subst aGroup.nsumTail*[of *R n λi. (Suc n)C_i ×_R x^{~R} i*], *assumption*,

rule allI, rule impI, rule nsClose, rule npClose, assumption)

apply (*cut-tac ring-one*,

simp del:nsum-suc binomial-Suc-Suc npow-suc add:aGroup.ag-l-zero)

done

lemma (**in** *Ring*) *tail-of-expansion1*:*x* \in carrier *R* \Rightarrow

$$(1_r \pm x)^{\sim R} (Suc n) = x \cdot_r (nsum R (\lambda i. ((Suc n)C_{(Suc i)} \times_R x^{\sim R} i)) n) \pm 1_r$$

apply (*frule tail-of-expansion*[of *x n*],

simp del:nsum-suc binomial-Suc-Suc npow-suc,

subgoal-tac $\forall i. Suc n C_{Suc i} \times_R x^{\sim R} i \in$ carrier *R*,

cut-tac ring-one, cut-tac ring-is-ag)

prefer 2 apply(*simp add: nsClose npClose*)

apply (*rule aGroup.ag-pOp-add-r*[of *R - - 1_r*], *assumption+*,

rule aGroup.nsum-mem, assumption+, rule allI, rule impI,

rule nsClose, rule npClose, assumption)

apply (*rule ring-tOp-closed, assumption+*,

rule aGroup.nsum-mem, assumption+, blast, simp add:ring-one)

apply (*subst nsumMulEleL*[of *λi. Suc n C_{Suc i} ×_R x^{~R} i x*], *assumption+*)

apply (*rule aGroup.nsum-eq, assumption, rule allI, rule impI, rule nsClose,*

rule npClose, assumption, rule allI, rule impI,

rule ring-tOp-closed, assumption+, rule nsClose, rule npClose,

assumption)

apply (*rule allI, rule impI*)

apply (*subst nsMulDistrL, assumption, simp add:npClose,*

frule-tac n = j in npClose[of *x*], *simp add:ring-tOp-commute*[of *x*])

done

lemma (**in** *Corps*) *nsum-in-VrTr:valuation K v* \Rightarrow

$(\forall j \leq n. f j \in$ carrier *K*) $\wedge (\forall j \leq n.$

$0 \leq (v(f j))) \longrightarrow (nsum K f n) \in$ carrier *(Vr K v)*

```

apply (induct-tac n)
  apply (rule impI, erule conjE, simp add:val-pos-mem-Vr)
  apply (rule impI, erule conjE)
  apply (frule Vr-ring[of v], frule Ring.ring-is-ag[of Vr K v],
    frule-tac x = f (Suc n) and y = nsum K f n in
      aGroup.ag-pOp-closed[of Vr K v],
      subst val-pos-mem-Vr[THEN sym, of v], assumption+,
      simp, simp, simp)
  apply (simp, subst Vr-pOp-f-pOp[of v, THEN sym], assumption+,
    subst val-pos-mem-Vr[THEN sym, of v], assumption+,
    simp+)
  apply (subst aGroup.ag-pOp-commute, assumption+, simp add:val-pos-mem-Vr,
    assumption)
done

lemma (in Corps) nsum-in-Vr:[valuation K v; ∀ j ≤ n. f j ∈ carrier K;
  ∀ j ≤ n. 0 ≤ (v (f j))] ⇒ (nsum K f n) ∈ carrier (Vr K v)
apply (simp add:nsum-in-VrTr)
done

lemma (in Corps) nsum-mem-in-Vr:[valuation K v;
  ∀ j ≤ n. (f j) ∈ carrier K; ∀ j ≤ n. 0 ≤ (v (f j))] ⇒
  (nsum K f n) ∈ carrier (Vr K v)
by (rule nsum-in-Vr)

lemma (in Corps) val-nscal-ge-selfTr:[valuation K v; x ∈ carrier K; 0 ≤ v x]
  ⇒ v x ≤ v (n ×K x)
apply (cut-tac field-is-ring, induct-tac n, simp)
apply (simp add:value-of-zero,
  simp,
  frule-tac y = n ×K x in amin-le-plus[of v x], assumption+,
  rule Ring.nsClose, assumption+)
apply (simp add:amin-def,
  frule Ring.ring-is-ag[of K],
  frule-tac n = n in Ring.nsClose[of K x], assumption+,
  simp add:aGroup.ag-pOp-commute)
done

lemma (in Corps) ApproximationTr:[valuation K v; x ∈ carrier K; 0 ≤ (v x)] ⇒
  v x ≤ (v (1r ± −a ((1r ± x) ^K (Suc n))))
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
  case-tac x = 0K,
  simp, frule Ring.ring-one[of K], simp add:aGroup.ag-r-zero,
  simp add:Ring.npOne, simp add:Ring.ring-l-one,simp add:aGroup.ag-r-inv1,
  subst Ring.tail-of-expansion1[of K x], assumption+,
  frule Ring.ring-one[of K])
apply (subgoal-tac (nsum K (λ i. Suc n C Suc i ×K x ^K i) n) ∈ carrier (Vr K v),
  frule Vr-mem-f-mem[of v (nsum K (λ i. Suc n C Suc i ×K x ^K i) n)],
```

```

assumption+,
frule-tac  $x = x$  and  $y = nsum K (\lambda i. Suc n C_{Suc i} \times_K x^{\wedge K i}) n$  in
Ring.ring-tOp-closed[of  $K$ ], assumption+,
subst aGroup.ag-pOp-commute[of  $K - 1_r$ ], assumption+,
subst aGroup.ag-p-inv[of  $K 1_r$ ], assumption+,
subst aGroup.ag-pOp-assoc[THEN sym], assumption+,
simp add:aGroup.ag-mOp-closed, rule aGroup.ag-mOp-closed, assumption+,
simp del:binomial-Suc-Suc add:aGroup.ag-r-inv1, subst aGroup.ag-l-zero,
assumption+,
rule aGroup.ag-mOp-closed, assumption+, simp add:val-minus-eq)

apply (subst val-t2p[of  $v$ ], assumption+) apply (
simp add:val-pos-mem-Vr[THEN sym, of  $v$ 
 $nsum K (\lambda i. (nC_i + nC_{Suc i}) \times_K x^{\wedge K i}) n$ ],
frule aadd-le-mono[of  $0 v (nsum K (\lambda i. (nC_i + nC_{Suc i}) \times_K x^{\wedge K i}) n)$ 
 $v x$ ], simp add:aadd-0-l, simp add:aadd-commute[of  $v x$ ])

apply (rule nsum-mem-in-Vr[of  $v n \lambda i. Suc n C_{Suc i} \times_K x^{\wedge K i}$ ], assumption,
rule allI, rule impI) apply (rule Ring.nsClose, assumption+) apply (simp
add:Ring.npClose)

apply (rule allI, rule impI)
apply (cut-tac  $i = 0$  and  $j = v (x^{\wedge K j})$  and  $k = v (Suc n C_{Suc j} \times_K x^{\wedge K j})$ 
in ale-trans)
apply (case-tac  $j = 0$ , simp add:value-of-one)
apply (simp add: val-exp-ring[THEN sym],
frule val-nonzero-z[of  $v x$ ], assumption+,
erule exE,
cut-tac  $m1 = 0$  and  $n1 = j$  in of-nat-less-iff[THEN sym],
frule-tac  $a = 0 < j$  and  $b = int 0 < int j$  in a-b-exchange,
assumption, thin-tac  $0 < j$ , thin-tac  $(0 < j) = (int 0 < int j)$ )
apply (simp del: of-nat-0-less-iff)

apply (frule-tac  $w1 = int j$  and  $x1 = 0$  and  $y1 = ant z$  in
asprod-pos-mono[THEN sym],
simp only:asprod-n-0)

apply(rule-tac  $x = x^{\wedge K j}$  and  $n = Suc n C_{Suc j}$  in
val-nscal-ge-selfTr[of  $v$ ], assumption+,
simp add:Ring.npClose, simp add:val-exp-ring[THEN sym],
frule val-nonzero-z[of  $v x$ ], assumption+, erule exE, simp)
apply (case-tac  $j = 0$ , simp)
apply (subst asprod-amult, simp, simp add:a-z-z)
apply(
simp only:ant-0[THEN sym], simp only:ale-zle,
cut-tac  $m1 = 0$  and  $n1 = j$  in of-nat-less-iff[THEN sym])
apply (frule-tac  $a = 0 < j$  and  $b = int 0 < int j$  in a-b-exchange,
assumption+, thin-tac  $0 < j$ , thin-tac  $(0 < j) = (int 0 < int j)$ ,
frule-tac  $z = int 0$  and  $z' = int j$  in zless-imp-zle,

```

```

frule-tac  $i = \text{int } 0 \text{ and } j = \text{int } j \text{ and } k = z$  in  $\text{int-mult-le}$ ,
assumption+, simp add:mult.commute )
apply assumption
done

lemma (in Corps) ApproximationTr0: $aa \in \text{carrier } K \implies$ 
 $(1_r \pm -_a (aa \wedge^K N)) \wedge^K N \in \text{carrier } K$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
rule Ring.npClose, assumption+,
rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,
rule aGroup.ag-mOp-closed, assumption+, rule Ring.npClose, assumption+)
done

lemma (in Corps) ApproximationTr1: $aa \in \text{carrier } K \implies$ 
 $1_r \pm -_a ((1_r \pm -_a (aa \wedge^K N)) \wedge^K N) \in \text{carrier } K$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule ApproximationTr0[of aa N],
frule Ring.ring-one[of K], rule aGroup.ag-pOp-closed, assumption+,
rule aGroup.ag-mOp-closed, assumption+)
done

lemma (in Corps) ApproximationTr2:[valuation K v; aa ∈ carrier K; aa ≠ 0;
 $0 \leq (v aa) \implies (\text{int } N) *_a (v aa) \leq (v (1_r \pm -_a ((1_r \pm -_a (aa \wedge^K N)) \wedge^K N)))$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
case-tac N = 0,
frule val-nonzero-z[of v aa], assumption+, erule exE, simp)
apply (frule Ring.ring-one[of K], simp add:aGroup.ag-r-inv1,
simp add:value-of-zero)

apply (frule-tac n = N in Ring.npClose[of K aa], assumption+,
frule ApproximationTr[of v -_a (aa \wedge^K N) N - Suc 0],
rule aGroup.ag-mOp-closed, assumption+, simp add:val-minus-eq,
subst val-exp-ring[THEN sym, of v], assumption+,
simp add:asprod-pos-pos)
apply (simp add:val-minus-eq, simp add:val-exp-ring[THEN sym])
done

lemma (in Corps) eSum-tr:
(  $\forall j \leq n. (x j) \in \text{carrier } K$  )  $\wedge$ 
(  $\forall j \leq n. (b j) \in \text{carrier } K$  )  $\wedge$   $l \leq n$   $\wedge$ 
(  $\forall j \in (\{h. h \leq n\} - \{l\}). (g j = (x j) \cdot_r (1_r \pm -_a (b j)))$  )  $\wedge$ 
 $g l = (x l) \cdot_r (-_a (b l))$ 
 $\longrightarrow (nsum K (\lambda j \in \{h. h \leq n\}. (x j) \cdot_r (1_r \pm -_a (b j))) n) \pm (-_a (x l)) =$ 
 $nsum K g n$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (induct-tac n)
apply (simp, rule impI, (erule conjE)+,
simp, frule Ring.ring-one[of K], subst Ring.ring-distrib1,
assumption+,

```

```

simp add:aGroup.ag-mOp-closed, simp add:Ring.ring-r-one,
frule aGroup.ag-mOp-closed[of K b 0], assumption+,
frule Ring.ring-tOp-closed[of K x 0 -a (b 0)], assumption+,
subst aGroup.ag-pOp-commute[of K x 0 -], assumption+,
subst aGroup.ag-pOp-assoc, assumption+,
frule aGroup.ag-mOp-closed[of K],
assumption+
apply (simp add:aGroup.ag-r-inv1, subst aGroup.ag-r-zero, assumption+, simp)
apply (rule impI, (erule conjE)+)
apply (subgoal-tac  $\forall j \leq (\text{Suc } n). ((x j) \cdot_r (1_r \pm -_a (b j))) \in \text{carrier } K$ )
apply (case-tac l = Suc n, simp)
apply (subgoal-tac  $\Sigma_e K g n \in \text{carrier } K$ ,
subgoal-tac  $\{h. h \leq (\text{Suc } n)\} - \{\text{Suc } n\} = \{h. h \leq n\}$ , simp,
subgoal-tac  $\forall j. j \leq n \longrightarrow j \leq (\text{Suc } n)$ ,
frule-tac  $f = \lambda u. \text{if } u \leq \text{Suc } n \text{ then } (x u) \cdot_r (1_r \pm -_a (b u)) \text{ else }$ 
undefined and  $n = n$  in aGroup.nsum-eq[of K - - g])
apply (rule allI, rule impI, simp,
rule allI, simp, rule allI, rule impI, simp, simp)

apply (cut-tac a = x (Suc n)  $\cdot_r (1_r \pm -_a (b (\text{Suc } n))) \pm -_a (x (\text{Suc } n))$  and
 $b = x (\text{Suc } n) \cdot_r (-_a (b (\text{Suc } n)))$  and
 $c = \Sigma_e K g n$  in aGroup.ag-pOp-add-l[of K], assumption)
apply (rule aGroup.ag-pOp-closed, assumption+,
rule Ring.ring-tOp-closed, assumption+, simp,
rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,
rule aGroup.ag-mOp-closed, assumption, simp,
rule aGroup.ag-mOp-closed, assumption, simp,
rule Ring.ring-tOp-closed, assumption+, simp,
rule aGroup.ag-mOp-closed, assumption+, simp, assumption)

apply (subst Ring.ring-distrib1, assumption+, simp, simp add:Ring.ring-one,
simp add:aGroup.ag-mOp-closed,
simp add:Ring.ring-r-one) apply (
frule-tac  $x = x (\text{Suc } n)$  and  $y = x (\text{Suc } n) \cdot_r (-_a (b (\text{Suc } n)))$  in
aGroup.ag-pOp-commute [of K], simp,
simp add:Ring.ring-tOp-closed aGroup.ag-mOp-closed,
simp) apply (
subst aGroup.ag-pOp-assoc[of K], assumption+,
rule Ring.ring-tOp-closed, assumption+, simp,
(simp add:aGroup.ag-mOp-closed)+,
subst aGroup.ag-r-inv1, assumption+, simp,
subst aGroup.ag-r-zero, assumption+,
simp add:Ring.ring-tOp-closed aGroup.ag-mOp-closed, simp,
rotate-tac -1, drule sym, simp) apply (
thin-tac  $\Sigma_e K g n \pm x (\text{Suc } n) \cdot_r (-_a (b (\text{Suc } n))) =$ 
 $\Sigma_e K g n \pm (x (\text{Suc } n) \cdot_r (1_r \pm -_a (b (\text{Suc } n))) \pm -_a (x (\text{Suc } n)))$ )
apply (subst aGroup.ag-pOp-assoc[THEN sym], assumption+,
rule Ring.ring-tOp-closed, assumption+, simp,
rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,

```

```

rule aGroup.ag-mOp-closed, assumption+, simp,
rule aGroup.ag-mOp-closed, assumption+, simp, simp,
simp, rule equalityI, rule subsetI, simp, rule subsetI, simp)
apply (rule aGroup.nsum-mem, assumption+,
rule allI, rule impI, simp)
defer
apply (rule allI, rule impI)
apply (case-tac j = l, simp,
rule Ring.ring-tOp-closed, assumption, simp,
rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,
rule aGroup.ag-mOp-closed, assumption, simp, simp,
rule Ring.ring-tOp-closed, assumption, simp,
rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,
rule aGroup.ag-mOp-closed, assumption, simp, simp)

apply (subst aGroup.ag-pOp-assoc, assumption+,
rule aGroup.nsum-mem, assumption+,
rule allI, simp, rule Ring.ring-tOp-closed, assumption+, simp,
rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,
rule aGroup.ag-mOp-closed, assumption, simp,
rule aGroup.ag-mOp-closed, assumption, simp,
subst aGroup.ag-pOp-commute[of K - -a (x l)], assumption+,
rule Ring.ring-tOp-closed, assumption, simp,
rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one)
apply (rule aGroup.ag-mOp-closed, assumption+, simp,
rule aGroup.ag-mOp-closed, assumption+, simp,
subst aGroup.ag-pOp-assoc[THEN sym], assumption+,
rule aGroup.nsum-mem, assumption+,
rule allI, rule impI, simp,
rule aGroup.ag-mOp-closed, assumption, simp,
rule Ring.ring-tOp-closed, assumption, simp,
rule aGroup.ag-pOp-closed, assumption+, simp add:Ring.ring-one,
rule aGroup.ag-mOp-closed, assumption, simp)
apply (subgoal-tac  $\Sigma_e$  K ( $\lambda a.$  if  $a \leq (\text{Suc } n)$  then  $x a \cdot_r (1_r \pm -_a (b a))$ 
else undefined)  $n \pm -_a (x l) =$ 
 $\Sigma_e$  K ( $\lambda a.$  if  $a \leq n$  then  $x a \cdot_r (1_r \pm -_a (b a))$  else undefined)  $n \pm$ 
 $-_a (x l)$ , simp,
rule aGroup.ag-pOp-add-r[of K - - -a (x l)], assumption+,
rule aGroup.nsum-mem, assumption+,
rule allI, rule impI, simp,
rule aGroup.nsum-mem, assumption+,
rule allI, rule impI, simp,
rule aGroup.ag-mOp-closed, assumption, simp,
rule aGroup.nsum-eq, assumption+,
rule allI, rule impI, simp, rule allI, rule impI)
apply simp
apply (rule allI, rule impI, simp)
done

```

lemma (in Corps) eSum-minus-x: $\llbracket \forall j \leq n. (x j) \in \text{carrier } K; \forall j \leq n. (b j) \in \text{carrier } K; l \leq n; \forall j \in (\{h. h \leq n\} - \{l\}). (g j = (x j) \cdot_r (1_r \pm -_a (b j))); g l = (x l) \cdot_r (-_a (b l)) \rrbracket \implies (nsum K (\lambda j \in \{h. h \leq n\}. (x j) \cdot_r (1_r \pm -_a (b j))) n) \pm (-_a (x l)) = nsum K g n$
by (cut-tac eSum-tr[of n x b l g], simp)

lemma (in Ring) one-m-x-times:x $\in \text{carrier } R \implies (1_r \pm -_a x) \cdot_r (nsum R (\lambda j. x^R j) n) = 1_r \pm -_a (x^R (Suc n))$
apply (cut-tac ring-one, cut-tac ring-is-ag,
frule aGroup.ag-mOp-closed[of R x], assumption+,
frule aGroup.ag-pOp-closed[of R 1_r -_a x], assumption+)

apply (induct-tac n)
apply (simp add:ring-r-one ring-l-one)
apply (simp del:npow-suc,
frule-tac n = Suc n in npClose[of x],
subst ring-distrib1, assumption+)

apply (rule aGroup.nsum-mem, assumption, rule allI, rule impI,
simp add:npClose, rule npClose, assumption+,
simp del:npow-suc,
thin-tac $(1_r \pm -_a x) \cdot_r \Sigma_e R (npow R x) n = 1_r \pm -_a (x^R (Suc n))$)

apply (subst ring-distrib2, assumption+,
simp del:npow-suc add:ring-l-one,
subst aGroup.pOp-assocTr43[of R], assumption+,
rule ring-tOp-closed, rule aGroup.ag-mOp-closed, assumption+)

apply (subst aGroup.ag-l-inv1, assumption+, simp del:npow-suc
add:aGroup.ag-r-zero,
frule-tac x = -_a x and y = x^R (Suc n) in ring-tOp-closed,
assumption+)

apply (rule aGroup.ag-pOp-add-l[of R - - 1_r], assumption+,
rule aGroup.ag-mOp-closed, assumption+,
rule npClose, assumption+,
subst ring-inv1-1[THEN sym, of x], assumption,
rule npClose, assumption,
simp,
subst ring-tOp-commute[of x], assumption+, simp)

done

lemma (in Corps) x-pow-fSum-in-Vr: $\llbracket \text{valuation } K v; x \in \text{carrier } (\text{Vr } K v) \rrbracket \implies (nsum K (npow K x) n) \in \text{carrier } (\text{Vr } K v)$
apply (frule Vr-ring[of v])
apply (induct-tac n)
apply simp
apply (frule Ring.ring-one[of Vr K v])
apply (simp add:Vr-1-f-1)
apply (simp del:npow-suc)

```

apply (frule Ring.ring-is-ag[of Vr K v])

apply (subst Vr-pOp-f-pOp[THEN sym, of v], assumption+)
apply (subst Vr-exp-f-exp[THEN sym, of v], assumption+)
apply (rule Ring.npClose[of Vr K v], assumption+)
apply (rule aGroup.ag-pOp-closed[of Vr K v], assumption+)
apply (subst Vr-exp-f-exp[THEN sym, of v], assumption+)
apply (rule Ring.npClose[of Vr K v], assumption+)
done

lemma (in Corps) val-1mx-pos:[valuation K v; x ∈ carrier K;
  0 < (v (1r ± -a x))] ⇒ v x = 0
apply (cut-tac field-is-ring, frule Ring.ring-one[of K],
  frule Ring.ring-is-ag[of K])
apply (frule aGroup.ag-mOp-closed[of K x], assumption+)
apply (frule aGroup.ag-pOp-closed[of K 1r -a x], assumption+)
apply (frule aGroup.ag-mOp-closed[of K 1r ± -a x], assumption+)
apply (cut-tac x = x and y = 1r ± -a (1r ± -a x) and f = v in
  eq-elems-eq-val)
apply (subst aGroup.ag-p-inv, assumption+,
  subst aGroup.ag-pOp-assoc[THEN sym], assumption+,
  rule aGroup.ag-mOp-closed, assumption+,
  subst aGroup.ag-inv-inv, assumption+,
  subst aGroup.ag-r-inv1, assumption+,
  subst aGroup.ag-l-zero, assumption+,
  (simp add:aGroup.ag-inv-inv)+,
  frule value-less-eq[of v 1r -a (1r ± -a x)],
  assumption+)
apply (simp add:val-minus-eq value-of-one,
  simp add:value-of-one)
done

lemma (in Corps) val-1mx-pow:[valuation K v; x ∈ carrier K;
  0 < (v (1r ± -a x))] ⇒ 0 < (v (1r ± -a x^K (Suc n)))
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (subst Ring.one-m-x-times[THEN sym, of K x n], assumption+)
apply (frule Ring.ring-one[of K],
  frule x-pow-fSum-in-Vr[of v x n],
  subst val-pos-mem-Vr[THEN sym], assumption+,
  frule val-1mx-pos[of v x], assumption+,
  simp)

apply (subst val-t2p, assumption+,
  rule aGroup.ag-pOp-closed, assumption+,
  simp add:aGroup.ag-mOp-closed, simp add:Vr-mem-f-mem,
  frule val-pos-mem-Vr[THEN sym, of v nsum K (npow K x) n],
  simp add:Vr-mem-f-mem, simp)
apply (frule aadd-le-mono[of 0 v (nsum K (npow K x) n) v (1r ± -a x)],
  simp add:aadd-0-l, simp add:aadd-commute)

```

done

```

lemma (in Corps) ApproximationTr3: vals-nonequiv K (Suc n) vv;
   $\forall l \leq (\text{Suc } n). x l \in \text{carrier } K; j \leq (\text{Suc } n) \Rightarrow$ 
   $\exists L. (\forall N. L < N \rightarrow (an m) \leq (vv j ((\Sigma_e K (\lambda h. h \leq (\text{Suc } n)) . (x k) \cdot_r (1_r \pm -_a ((1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) k) \wedge^K N) \wedge^K N)))$ 
   $(\text{Suc } n)) \pm -_a (x j))))$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (frule-tac vals-nonequiv-valuation[of Suc n vv j], assumption+)
apply (subgoal-tac  $\forall N. \Sigma_e K (\lambda j. h \leq (\text{Suc } n)). (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) j) \wedge^K N) \wedge^K N)) (\text{Suc } n) \pm -_a (x j) =$ 
 $\Sigma_e K (\lambda l. h \leq (\text{Suc } n)). \text{if } l \neq j \text{ then } (x l) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) l) \wedge^K N) \wedge^K N) \text{ else } (x j) \cdot_r (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) l) \wedge^K N) \wedge^K N \pm -_a (1_r)) (\text{Suc } n))$ 
apply (simp del:nsum-suc)
apply (thin-tac  $\forall N. \Sigma_e K (\lambda j. h \leq (\text{Suc } n)). (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) j) \wedge^K N) \wedge^K N)) (\text{Suc } n) \pm -_a (x j) = \Sigma_e K (\lambda l. h \leq (\text{Suc } n)). \text{if } l \neq j \text{ then } (x l) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) l) \wedge^K N) \wedge^K N) \text{ else } (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) l) \wedge^K N) \wedge^K N \pm -_a (1_r)) (\text{Suc } n))$ 
prefer 2 apply (rule allI)
apply (rule eSum-minus-x, assumption+)
apply (rule allI, rule impI) apply (rule ApproximationTr0)
apply (simp add:Ostrowski-base-mem) apply assumption
apply (rule ballI, simp)
apply simp
apply (frule Ring.ring-one[of K])
apply (cut-tac aa =  $(\Omega_K vv (\text{Suc } n)) j$  and  $N = N$  in ApproximationTr0)
apply (simp add:Ostrowski-base-mem)
apply (subst aGroup.ag-pOp-assoc, assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+)+
apply (subst aGroup.ag-pOp-commute[of K - -_a 1_r], assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+)+

apply (subst aGroup.ag-pOp-assoc[THEN sym], assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+)+
apply (simp add:aGroup.ag-r-inv1)
apply (subst aGroup.ag-l-zero, assumption+) apply (simp add:aGroup.ag-mOp-closed)
apply simp

apply (subgoal-tac  $\exists L. \forall N. L < N \rightarrow$ 
 $(\forall ja \leq (\text{Suc } n). (an m) \leq ((vv j \circ (\lambda l. h \leq (\text{Suc } n)). \text{if } l \neq j \text{ then } (x l) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) l) \wedge^K N) \wedge^K N) \text{ else } (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) l) \wedge^K N) \wedge^K N \pm -_a (1_r))) ja)))$ 

```

```

apply (erule exE)
apply (rename-tac M)
apply (subgoal-tac  $\forall N. M < (N::nat) \rightarrow$ 
 $(an m) \leq (vv j (\Sigma_e K (\lambda l \in \{h. h \leq (Suc n)\}. (if l \neq j then$ 
 $(x l) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (Suc n)) l) \wedge^K N) \wedge^K N)$ 
 $else (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (Suc n)) l) \wedge^K N) \wedge^K N$ 
 $\pm -_a 1_r))) (Suc n))))$ 
apply blast
apply (rule allI, rule impI)
apply (drule-tac a = N in forall-spec, assumption)
apply (rule value-ge-add[of vv j Suc n - an m], assumption+)

apply (rule allI, rule impI)
apply (frule Ring.ring-one[of K])
apply (case-tac ja = j, simp)
apply (rule Ring.ring-tOp-closed, assumption+, simp)
apply (rule aGroup.ag-pOp-closed, assumption+)+
apply (rule aGroup.ag-mOp-closed, assumption+)
apply (rule Ring.npClose, assumption)
apply (rule aGroup.ag-pOp-closed, assumption+)
apply (rule aGroup.ag-mOp-closed, assumption)
apply (rule Ring.npClose, assumption)
apply (simp add:Ostrowski-base-mem)
apply (rule aGroup.ag-mOp-closed, assumption+)

apply simp
apply (rule Ring.ring-tOp-closed, assumption+, simp)
apply (rule aGroup.ag-pOp-closed, assumption+)+
apply (rule aGroup.ag-mOp-closed, assumption+)
apply (rule Ring.npClose, assumption)
apply (rule aGroup.ag-pOp-closed, assumption+)
apply (rule aGroup.ag-mOp-closed, assumption)
apply (rule Ring.npClose, assumption)
apply (simp add:Ostrowski-base-mem)

apply assumption

apply (subgoal-tac  $\forall N. \forall ja \leq (Suc n). (1_r \pm -_a (1_r \pm -_a$ 
 $((\Omega_K vv (Suc n)) ja) \wedge^K N) \wedge^K N) \in carrier K)$ 
apply (subgoal-tac  $\forall N. (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (Suc n)) j) \wedge^K N) \wedge^K N$ 
 $\pm -_a 1_r) \in carrier K)$ 
apply (simp add:val-t2p)
apply (cut-tac multi-inequalityTr0[of Suc n (vv j) o x m])
apply (subgoal-tac  $\forall ja \leq (Suc n). (vv j o x) ja \neq -\infty, simp$ )
apply (erule exE)
apply (subgoal-tac  $\forall N. L < N \rightarrow (\forall ja \leq (Suc n). (ja \neq j \rightarrow$ 
 $an m \leq vv j (x ja) + (vv j (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (Suc n)) ja) \wedge^K N) \wedge^K N)))$ )

```

```

 $\wedge (ja = j \longrightarrow (an m) \leq vv j (x j) + (vv j (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (Suc n)) j) \sim^K N) \sim^K N \pm -_a (1_r))))$ 
apply blast
apply (rule allI, rule impI) +

apply (case-tac ja = j, simp)
apply (thin-tac  $\forall N. 1_r \pm -_a (1_r \pm -_a (\Omega_K vv (Suc n)) j) \sim^K N \sim^K N \pm -_a 1_r$ 
 $\in carrier K)$ 
apply (thin-tac  $\forall l \leq Suc n. x l \in carrier K$ )
apply (drule-tac  $x = N$  in spec)
apply (drule-tac  $a = j$  in forall-spec, assumption,
 $thin-tac \forall ja \leq Suc n. 1_r \pm -_a (1_r \pm -_a (\Omega_K vv (Suc n)) ja \sim^K N) \sim^K N$ 
 $\in carrier K)$ 
apply (cut-tac  $N = N$  in ApproximationTr0 [of  $(\Omega_K vv (Suc n)) j$ ])
apply (simp add:Ostrowski-base-mem)
apply (frule Ring.ring-one[of K], frule aGroup.ag-mOp-closed[of K 1_r],
assumption) apply (
frule-tac  $x = (1_r \pm -_a ((\Omega_K vv (Suc n)) j) \sim^K N) \sim^K N$  in
aGroup.ag-mOp-closed[of K], assumption+)
apply (simp only:aGroup.ag-pOp-assoc)
apply (simp only:aGroup.ag-pOp-commute[of K - -_a 1_r])
apply (simp only:aGroup.ag-pOp-assoc[THEN sym])
apply (simp add:aGroup.ag-r-inv1)
apply (simp add:aGroup.ag-l-zero) apply (simp only:val-minus-eq)
apply (thin-tac  $(1_r \pm -_a (\Omega_K vv (Suc n)) j) \sim^K N \sim^K N \in carrier K$ ,
 $thin-tac -_a (1_r \pm -_a (\Omega_K vv (Suc n)) j) \sim^K N \sim^K N \in carrier K)$ 
apply (subst val-exp-ring[THEN sym, of vv j], assumption+)
apply (rule aGroup.ag-pOp-closed[of K], assumption+)
apply (rule aGroup.ag-mOp-closed[of K], assumption)
apply (rule Ring.npClose, assumption+) apply (simp add:Ostrowski-base-mem)
apply (rule Ostrowski-base-npow-not-one) apply simp apply assumption+
apply (drule-tac  $a = N$  in forall-spec, assumption)
apply (drule-tac  $a = j$  in forall-spec, assumption)
apply (frule Ostrowski-baseTr1[of n vv j], assumption+)
apply (frule-tac  $n = N - Suc 0$  in val-1mx-pow[of vv j  $(\Omega_K vv (Suc n)) j$ ])
apply (simp add:Ostrowski-base-mem) apply assumption
apply (thin-tac  $vv j (x j) \neq -\infty$ ) apply (simp only:Suc-pred)
apply (thin-tac  $0 < vv j (1_r \pm -_a ((\Omega_K vv (Suc n)) j))$ )
apply (cut-tac  $b = vv j (1_r \pm -_a ((\Omega_K vv (Suc n)) j) \sim^K N)$  and  $N = N$  in
asprod-ge) apply assumption apply simp
apply (cut-tac  $x = an N$  and  $y = int N *_a vv j (1_r \pm -_a ((\Omega_K vv (Suc n))$ 
 $j) \sim^K N)$  in aadd-le-mono[of - - vv j (x j)], assumption)
apply (simp add:aadd-commute)

apply simp

```

```

apply (frule-tac aa = ( $\Omega_K vv (Suc n)$ ) ja and N = N in
      ApproximationTr2[of vv j])
apply (simp add:Ostrowski-base-mem)
apply (rule Ostrowski-base-nonzero, assumption+)
apply (frule-tac l = ja in Ostrowski-baseTr0[of n vv], assumption+,
       erule conjE)
apply (rotate-tac -1, frule-tac a = j in forall-spec) apply assumption
apply (frule-tac x = j in bspec, simp)
apply (rule aless-imp-le) apply blast
apply (rotate-tac -5,
       drule-tac a = N in forall-spec, assumption)
apply (rotate-tac -2,
       drule-tac a = ja in forall-spec, assumption) apply (
       drule-tac a = ja in forall-spec, assumption)
apply (frule-tac l = ja in Ostrowski-baseTr0[of n vv], assumption+)
apply (erule conjE, rotate-tac -1,
       frule-tac a = j in forall-spec, assumption+)
apply (thin-tac vv j (x ja) ≠ -∞)
apply (cut-tac b = vv j (( $\Omega_K vv (Suc n)$ ) ja) and N = N in asprod-ge)
apply simp apply simp
apply (frule-tac x = an N and y = int N *a vv j (( $\Omega_K vv (Suc n)$ ) ja) and
      z = vv j (x ja) in aadd-le-mono)
apply (frule-tac x = int N *a vv j (( $\Omega_K vv (Suc n)$ ) ja) and y = (vv j)
      (1r ± -a (1r ± -a (( $\Omega_K vv (Suc n)$ ) ja) ^K N) ^K N) and z = vv j (x ja)
      in aadd-le-mono)
apply (frule-tac i = an N + vv j (x ja) and
      j = int N *a vv j (( $\Omega_K vv (Suc n)$ ) ja) + vv j (x ja) and
      k = vv j (1r ± -a (1r ± -a (( $\Omega_K vv (Suc n)$ ) ja) ^K N) ^K N) +
          vv j (x ja) in ale-trans, assumption+)
apply (subst aadd-commute)
apply (frule-tac x = an m and y = vv j (x ja) + an N in aless-imp-le)
apply (rule-tac j = vv j (x ja) + an N in ale-trans[of an m],
       assumption)
apply (simp add:aadd-commute)
apply (rule allI, rule impI, subst comp-def)
apply (frule-tac a = ja in forall-spec, assumption)
apply (frule-tac x = x ja in value-in-aug-inf[of vv j], assumption+)
apply (simp add:aug-inf-def)

apply (rule allI)
apply (rule aGroup.ag-pOp-closed, assumption+) apply blast
apply (rule aGroup.ag-mOp-closed, assumption, rule Ring.ring-one, assumption)

apply ((rule allI)+, rule impI)
apply (rule-tac aa = ( $\Omega_K vv (Suc n)$ ) ja in ApproximationTr1,
       simp add:Ostrowski-base-mem)
done

```

definition

```

app-lb :: [- , nat, nat ⇒ 'b ⇒ ant, nat ⇒ 'b, nat] ⇒
          (nat ⇒ nat) ((5Ψ_ . . . .) [98,98,98,98,99]98) where
Ψ_K n vv x m = (λj∈{h. h ≤ n}. (SOME L. (∀N. L < N →
(an m) ≤ (vv j (Σ_e K (λj∈{h. h ≤ n}. (x j) ·_r (1_r ±_K (1_r ±_K
(1_r ±_K −_a K ((Ω_K vv n) j) ^_K N) ^_K N)) n ±_K −_a K (x j)))))))

```

lemma (in Corps) app-LB:[vals-nonequiv K (Suc n) vv;

$$\forall l \leq (\text{Suc } n). x l \in \text{carrier } K; j \leq (\text{Suc } n) \implies$$

$$\forall N. (\Psi_K (\text{Suc } n) vv x m) j < N \longrightarrow (an m) \leq$$

$$(vv j (\Sigma_e K (\lambda j \in \{h. h \leq (\text{Suc } n)\}. (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n) j) ^_K N) ^_K N)) (\text{Suc } n) \pm -_a (x j))))$$

apply (frule ApproximationTr3[of n vv x j m],
assumption+)

apply (simp del:nsum-suc add:app-lb-def) **apply** (rule allI)
apply (rule someI2-ex) **apply** assumption+
apply (rule impI) **apply** blast
done

lemma (in Corps) ApplicationTr4:[vals-nonequiv K (Suc n) vv;

$$\forall j \in \{h. h \leq (\text{Suc } n)\}. x j \in \text{carrier } K \implies$$

$$\exists l. \forall N. l < N \longrightarrow (\forall j \leq (\text{Suc } n). (an m) \leq$$

$$(vv j (\Sigma_e K (\lambda j \in \{h. h \leq (\text{Suc } n)\}. (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n) j) ^_K N) ^_K N)) (\text{Suc } n) \pm -_a (x j))))$$

apply (subgoal-tac $\forall N. (m\text{-max} (\text{Suc } n) (\Psi_K (\text{Suc } n) vv x m)) < N \longrightarrow$
 $(\forall j \leq (\text{Suc } n). (an m) \leq$
 $(vv j (\Sigma_e K (\lambda j \in \{h. h \leq (\text{Suc } n)\}. (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n) j) ^_K N) ^_K N)) (\text{Suc } n) \pm -_a (x j))))$)

apply blast
apply (rule allI, rule impI)+
apply (frule-tac $j = j$ in app-LB[of n vv x - m],
simp, assumption,
subgoal-tac $(\Psi_K (\text{Suc } n) vv x m) j < N$, blast)
apply (frule-tac $l = j$ and $n = \text{Suc } n$ and $f = \Psi_K (\text{Suc } n) vv x m$ in m-max-gt,
frule-tac $x = (\Psi_K (\text{Suc } n) vv x m) j$ and
 $y = m\text{-max} (\text{Suc } n) (\Psi_K (\text{Suc } n) vv x m)$ and $z = N$ in le-less-trans,
assumption+)

done

theorem (in Corps) Approximation-thm:[vals-nonequiv K (Suc n) vv;
 $\forall j \leq (\text{Suc } n). (x j) \in \text{carrier } K \implies$
 $\exists y \in \text{carrier } K. \forall j \leq (\text{Suc } n). (an m) \leq (vv j (y \pm -_a (x j)))$
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (subgoal-tac $\exists l. (\forall N. l < N \longrightarrow (\forall j \leq (\text{Suc } n). (an m) \leq ((vv j) ((nsum K$
 $(\lambda j \in \{h. h \leq (\text{Suc } n)\}. (x j) \cdot_r (1_r \pm -_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n) j) ^_K N) ^_K N)) (\text{Suc } n) \pm -_a (x j))))))$

```

apply (erule exE)
apply (rename-tac M)
apply (subgoal-tac  $\forall j \leq (\text{Suc } n)$ .  $(an m) \leq$ 
 $(vv j ( (\Sigma_e K (\lambda j \in \{h. h \leq (\text{Suc } n)\}. (x j) \cdot_r (1_r \pm -_a (1_r \pm$ 
 $-_a ((\Omega_K vv (\text{Suc } n)) j) \wedge^K (\text{Suc } M)) \wedge^K (\text{Suc } M))) (\text{Suc } n)) \pm -_a (x j)))$ )
apply (subgoal-tac  $\Sigma_e K (\lambda j \in \{h. h \leq (\text{Suc } n)\}. (x j) \cdot_r (1_r \pm$ 
 $-_a (1_r \pm -_a ((\Omega_K vv (\text{Suc } n)) j) \wedge^K (\text{Suc } M)) \wedge^K (\text{Suc } M))) (\text{Suc } n) \in \text{carrier } K$ )
apply blast
apply (rule aGroup.nsum-mem[of K Suc n], assumption+)
apply (rule allI, rule impI, simp del:nsum-suc npow-suc)
apply (rule Ring.ring-tOp-closed, assumption+, simp,
rule ApproximationTr1, simp add:Ostrowski-base-mem)

```

```

apply (subgoal-tac M < Suc M) apply blast
apply simp
apply (rule ApplicationTr4[of n vv x], assumption+)
apply simp
done

```

definition

```

distinct-pds :: [-, nat, nat  $\Rightarrow$  ('b  $\Rightarrow$  ant) set]  $\Rightarrow$  bool where
distinct-pds K n P  $\longleftrightarrow$   $(\forall j \leq n. P j \in \text{Pds}_K) \wedge$ 
 $(\forall l \leq n. \forall m \leq n. l \neq m \longrightarrow P l \neq P m)$ 

```

```

lemma (in Corps) distinct-pds-restriction:  $\llbracket \text{distinct-pds } K (\text{Suc } n) P \rrbracket \implies$ 
 $\text{distinct-pds } K n P$ 
apply (simp add:distinct-pds-def)
done

```

```

lemma (in Corps) ring-n-distinct-prime-divisors:distinct-pds K n P  $\implies$ 
 $\text{Ring } (\text{Sr } K \{x. x \in \text{carrier } K \wedge (\forall j \leq n. 0 \leq ((\nu_K (P j)) x)))\}$ 
apply (simp add:distinct-pds-def) apply (erule conjE)
apply (cut-tac field-is-ring)
apply (rule Ring.Sr-ring, assumption+)
apply (subst sr-def)
apply (rule conjI)
apply (rule subsetI) apply simp
apply (rule conjI)
apply (simp add:Ring.ring-one)
apply (rule allI, rule impI)
apply (cut-tac P = P j in representative-of-pd-valuation, simp,
simp add:value-of-one)
apply (rule ballI)
apply simp
apply (frule Ring.ring-is-ag[of K]) apply (erule conjE)
apply (frule-tac x = y in aGroup.ag-mOp-closed[of K], assumption+)
apply (frule-tac x = x and y = -_a y in aGroup.ag-pOp-closed[of K],

```

```

assumption+)
apply simp
apply (rule conjI)
apply (rule allI, rule impI)
apply (rotate-tac -4, frule-tac a = j in forall-spec, assumption,
      rotate-tac -3,
      drule-tac a = j in forall-spec, assumption)
apply (cut-tac P = P j in representative-of-pd-valuation, simp)
apply (frule-tac v = νK(P j) and x = x and y = -ay in amin-le-plus,
      assumption+)
apply (simp add:val-minus-eq)
apply (frule-tac x = (νK(P j)) x and y = (νK(P j)) y in amin-ge1[of 0])
apply simp
apply (rule-tac j = amin ((νK(P j)) x) ((νK(P j)) y) and k = (νK(P j)) (x ±
      -ay) in ale-trans[of 0], assumption+)
apply (simp add:Ring.ring-tOp-closed)

apply (rule allI, rule impI,
      cut-tac P = P j in representative-of-pd-valuation, simp,
      subst val-t2p [where v=νKP j], assumption+,
      rule aadd-two-pos, simp+)
done

lemma (in Corps) distinct-pds-valuation:[j ≤ (Suc n);
      distinct-pds K (Suc n) P] ⇒ valuation K (νK(P j))
apply (rule-tac P = P j in representative-of-pd-valuation)
apply (simp add:distinct-pds-def)
done

lemma (in Corps) distinct-pds-valuation1:[0 < n; j ≤ n; distinct-pds K n P]
      ⇒ valuation K (νK(P j))
apply (rule distinct-pds-valuation[of j n - Suc 0 P])
apply simp+
done

lemma (in Corps) distinct-pds-valuation2:[j ≤ n; distinct-pds K n P] ⇒
      valuation K (νK(P j))
apply (case-tac n = 0,
      simp add:distinct-pds-def,
      subgoal-tac 0 ∈ {0::nat},
      simp add:representative-of-pd-valuation[of P 0],
      simp)

apply (simp add:distinct-pds-valuation1[of n])
done

definition
ring-n-pd :: [(b, m) Ring-scheme, nat ⇒ (b ⇒ ant) set,

```

$nat \Rightarrow ('b, 'm) \text{ Ring-scheme}$
 $((\exists O_{-} -) [98, 98, 99] 98) \text{ where}$
 $O_{K P n} = Sr K \{x. x \in carrier K \wedge$
 $(\forall j \leq n. 0 \leq ((\nu_{K (P j)}) x))\}$

lemma (in Corps) $ring-n-pd:distinct-pds K n P \Rightarrow Ring (O_{K P n})$
by (*simp add:ring-n-pd-def, simp add:ring-n-distinct-prime-divisors*)

lemma (in Corps) $ring-n-pd-Suc:distinct-pds K (Suc n) P \Rightarrow$
 $carrier (O_{K P (Suc n)}) \subseteq carrier (O_{K P n})$
apply (*rule subsetI*)
apply (*simp add:ring-n-pd-def Sr-def*)
done

lemma (in Corps) $ring-n-pd-pOp-K-pOp:distinct-pds K n P; x \in carrier (O_{K P n});$
 $y \in carrier (O_{K P n}) \Rightarrow x \pm_{(O_{K P n})} y = x \pm y$
apply (*simp add:ring-n-pd-def Sr-def*)
done

lemma (in Corps) $ring-n-pd-tOp-K-tOp:distinct-pds K n P; x \in carrier (O_{K P n});$
 $y \in carrier (O_{K P n}) \Rightarrow x \cdot_r_{(O_{K P n})} y = x \cdot_r y$
apply (*simp add:ring-n-pd-def Sr-def*)
done

lemma (in Corps) $ring-n-eSum-K-eSumTr:distinct-pds K n P \Rightarrow$
 $(\forall j \leq m. f j \in carrier (O_{K P n})) \rightarrow nsum (O_{K P n}) f m = nsum K f m$
apply (*induct-tac m*)
apply (*rule impI, simp*)

apply (*rule impI, simp,*
subst ring-n-pd-pOp-K-pOp, assumption+,
frule-tac n = n in ring-n-pd[of - P],
frule-tac Ring.ring-is-ag, drule sym, simp)
apply (*rule aGroup.nsum-mem, assumption+, simp+*)
done

lemma (in Corps) $ring-n-eSum-K-eSum:distinct-pds K n P;$
 $\forall j \leq m. f j \in carrier (O_{K P n}) \Rightarrow nsum (O_{K P n}) f m = nsum K f m$
apply (*simp add:ring-n-eSum-K-eSumTr*)
done

lemma (in Corps) $ideal-eSum-closed:distinct-pds K n P; ideal (O_{K P n}) I;$
 $\forall j \leq m. f j \in I \Rightarrow nsum K f m \in I$
apply (*frule ring-n-pd[of n P]) thm Ring.ideal-nsum-closed*)
apply (*frule-tac n = m in Ring.ideal-nsum-closed[of (O_{K P n}) I - f], assumption+*)
apply (*subst ring-n-eSum-K-eSum [THEN sym, of n P m f], assumption+, rule allI, simp add:Ring.ideal-subset*)

```

apply assumption
done

definition
prime-n-pd :: [-, nat ⇒ ('b ⇒ ant) set,
                  nat, nat] ⇒ 'b set
      ((4P_ _ _ _) [98,98,98,99]98) where
      P_K P n j = {x. x ∈ (carrier (O_K P n)) ∧ 0 < ((ν_K (P j)) x) }

lemma (in Corps) zero-in-ring-n-pd-zero-K:distinct-pds K n P ==>
      0_(O_K P n) = 0_K
apply (simp add:ring-n-pd-def Sr-def)
done

lemma (in Corps) one-in-ring-n-pd-one-K:distinct-pds K n P ==>
      1_r(O_K P n) = 1_r
apply (simp add:ring-n-pd-def Sr-def)
done

lemma (in Corps) mem-ring-n-pd-mem-K:[distinct-pds K n P; x ∈ carrier (O_K P n)]
      ==> x ∈ carrier K
apply (simp add:ring-n-pd-def Sr-def)
done

lemma (in Corps) ring-n-tOp-K-tOp:[distinct-pds K n P; x ∈ carrier (O_K P n);
      y ∈ carrier (O_K P n)] ==> x ·_r(O_K P n) y = x ·_r y
apply (simp add:ring-n-pd-def Sr-def)
done

lemma (in Corps) ring-n-exp-K-exp:[distinct-pds K n P; x ∈ carrier (O_K P n)]
      ==> x^K m = x^(O_K P n) m
apply (frule ring-n-pd[of n P])
apply (induct-tac m) apply simp
apply (simp add:one-in-ring-n-pd-one-K)

apply simp
apply (frule-tac n = na in Ring.npClose[of O_K P n x], assumption+)
apply (simp add:ring-n-tOp-K-tOp)
done

lemma (in Corps) prime-n-pd-prime:[distinct-pds K n P; j ≤ n] ==>
      prime-ideal (O_K P n) (P_K P n j)
apply (subst prime-ideal-def)
apply (rule conjI)
apply (simp add:ideal-def)
apply (rule conjI)
apply (rule aGroup.asubg-test)
apply (frule ring-n-pd[of n P], simp add:Ring.ring-is-ag)

```

```

apply (rule subsetI, simp add:prime-n-pd-def)
apply (subgoal-tac  $0_{(O_K P_n)} \in P_{K P n} j$ )
apply blast

apply (simp add:zero-in-ring-n-pd-zero-K)
apply (simp add:prime-n-pd-def)
apply (simp add: ring-n-pd-def Sr-def)
apply (cut-tac field-is-ring, simp add:Ring.ring-zero)
apply (rule conjI) apply (rule allI, rule impI)
apply (cut-tac  $P = P j$  in representative-of-pd-valuation,
       simp add:distinct-pds-def, simp add:value-of-zero)
apply (cut-tac  $P = P j$  in representative-of-pd-valuation,
       simp add:distinct-pds-def, simp add:value-of-zero)
apply (simp add:ant-0[THEN sym])

apply (rule ballI)+
apply (simp add:prime-n-pd-def) apply (erule conjE)+
apply (frule ring-n-pd [of n P], frule Ring.ring-is-ag[of  $O_K P_n$ ])
apply (frule-tac  $x = b$  in aGroup.ag-mOp-closed[of  $O_K P_n$ ], assumption+)
apply (simp add:aGroup.ag-pOp-closed)
apply (thin-tac Ring ( $O_K P_n$ )) apply (thin-tac aGroup ( $O_K P_n$ ))
apply (simp add:ring-n-pd-def Sr-def)
apply (erule conjE)+
apply (cut-tac  $v = \nu_K (P j)$  and  $x = a$  and  $y = -_a b$  in
       amin-le-plus)
apply (rule-tac  $P = P j$  in representative-of-pd-valuation,
       simp add:distinct-pds-def)
apply assumption+
apply (cut-tac  $P = P j$  in representative-of-pd-valuation)
apply (simp add:distinct-pds-def)
apply (frule-tac  $x = (\nu_K (P j)) a$  and  $y = (\nu_K (P j)) (-_a b)$  in
       amin-gt[of 0])
apply (simp add:val-minus-eq)

apply (frule-tac  $y = amin ((\nu_K (P j)) a) ((\nu_K (P j)) (-_a b))$  and
 $z = (\nu_K (P j)) (a \pm -_a b)$  in aless-le-trans[of 0], assumption+)

apply (rule ballI)+
apply (frule ring-n-pd [of n P])
apply (frule-tac  $x = r$  and  $y = x$  in Ring.ring-tOp-closed[of  $O_K P_n$ ],
       assumption+)
apply (simp add:prime-n-pd-def)
apply (cut-tac  $P = P j$  in representative-of-pd-valuation,
       simp add:distinct-pds-def)
apply (thin-tac Ring ( $O_K P_n$ ))
apply (simp add:prime-n-pd-def ring-n-pd-def Sr-def, (erule conjE)+,
       simp add:val-t2p)
apply (subgoal-tac  $0 \leq ((\nu_K (P j)) r)$ )

```

```

apply (simp add:aadd-pos-poss, simp)

apply (rule conjI,
      rule contrapos-pp, simp+,
      simp add:prime-n-pd-def,
      (erule conjE)+, simp add: one-in-ring-n-pd-one-K,
      simp add:distinct-pds-def, (erule conjE)+,
      cut-tac representative-of-pd-valuation[of P j],
      simp add:value-of-one, simp)

apply ((rule ballI)+, rule impI)
apply (rule contrapos-pp, simp+, erule conjE,
       simp add:prime-n-pd-def, (erule conjE)+,
       simp add:ring-n-pd-def Sr-def, (erule conjE)+,
       simp add:aneg-less,
       frule-tac  $x = (\nu_K(P_j)) x$  in ale-antisym[of - 0], simp,
       frule-tac  $x = (\nu_K(P_j)) y$  in ale-antisym[of - 0], simp)

apply (simp add:distinct-pds-def, (erule conjE)+,
       cut-tac representative-of-pd-valuation[of P j],
       simp add:val-t2p aadd-0-l,
       simp)
done

lemma (in Corps) n-eq-val-eq-idealTr:
   $\llbracket \text{distinct-pds } K \ n \ P; x \in \text{carrier } (O_{K \ P \ n}); y \in \text{carrier } (O_{K \ P \ n}); \forall j \leq n. ((\nu_K(P_j)) x) \leq ((\nu_K(P_j)) y) \rrbracket \implies Rxa(O_{K \ P \ n}) y \subseteq Rxa(O_{K \ P \ n}) x$ 
apply (subgoal-tac  $\forall j \leq n.$  valuation  $K(\nu_K(P_j))$ )
apply (case-tac  $x = \mathbf{0}_{(O_{K \ P \ n})}$ ,
           simp add:zero-in-ring-n-pd-zero-K)
apply (simp add:value-of-zero)
apply (subgoal-tac  $y = \mathbf{0}$ , simp,
           drule-tac  $a = n$  in forall-spec, simp,
           drule-tac  $a=n$  in forall-spec, simp)
apply (cut-tac inf-ge-any[of  $(\nu_K(P_n)) y$ ],
       frule ale-antisym[of  $(\nu_K(P_n)) y \infty$ ], assumption+)
apply (rule value-inf-zero, assumption+)
apply (simp add:mem-ring-n-pd-mem-K, assumption)

apply (frule ring-n-pd[of n P])
apply (subgoal-tac  $\forall j \leq n.$   $0 \leq ((\nu_K(P_j)) (y \cdot_r (x^K)))$ )
apply (subgoal-tac  $(y \cdot_r (x^K)) \in \text{carrier } (O_{K \ P \ n})$ )
apply (cut-tac field-frac-mul[of y x],
       frule Ring.rxa-in-Rxa[of  $O_{K \ P \ n} x y \cdot_r (x^K)$ ], assumption+,
       simp add:ring-n-pd-tOp-K-tOp[THEN sym],
       frule Ring.principal-ideal[of  $O_{K \ P \ n} x$ ], assumption+)

apply (cut-tac Ring.ideal-cont-Rxa[of  $O_{K \ P \ n} (O_{K \ P \ n}) \diamond_p x y$ ],

```

```

assumption+,
simp add:mem-ring-n-pd-mem-K,
simp add:mem-ring-n-pd-mem-K,
simp add:zero-in-ring-n-pd-zero-K)
apply (frule Ring.rxa-in-Rxa[of  $O_K P n x y \cdot_r (x^K)$ ], assumption+,
      simp add:ring-n-pd-def Sr-def,
      (erule conjE)+,
      cut-tac field-is-ring, rule Ring.ring-tOp-closed, assumption+,
      cut-tac invf-closed1[of x], simp, simp,
      simp add:ring-n-pd-def Sr-def)
apply (cut-tac Ring.ring-tOp-closed, assumption+,
       cut-tac field-is-ring, assumption+, simp+,
       cut-tac invf-closed1[of x], simp, simp)

apply (rule allI, rule impI, drule-tac a = j in forall-spec, assumption+,
       cut-tac invf-closed1[of x], simp, erule conjE)
apply (subst val-t2p [where v= $\nu_K P j$ ], simp,
       rule mem-ring-n-pd-mem-K[of n P y], assumption+,
       frule-tac x = j in spec, simp,
       simp add:zero-in-ring-n-pd-zero-K)
apply (subst value-of-inv [where v= $\nu_K P j$ ], simp,
       simp add:ring-n-pd-def Sr-def, assumption+)
apply (frule-tac x = ( $\nu_K (P j)$ ) x and y = ( $\nu_K (P j)$ ) y in ale-diff-pos,
       simp add:diff-ant-def,
       simp add:mem-ring-n-pd-mem-K[of n P x] zero-in-ring-n-pd-zero-K)

apply (rule allI, rule impI,
      simp add:distinct-pds-def, (erule conjE)+,
      rule-tac P = P j in representative-of-pd-valuation, simp)
done

lemma (in Corps) n-eq-val-eq-ideal: [| distinct-pds K n P; x ∈ carrier (O_K P n);
y ∈ carrier (O_K P n); ∀ j ≤ n. (( $\nu_K (P j)$ ) x) = (( $\nu_K (P j)$ ) y) |] ==>
Rxa (O_K P n) x = Rxa (O_K P n) y
apply (rule equalityI)
apply (subgoal-tac ∀ j ≤ n. ( $\nu_K (P j)$ ) y ≤ (( $\nu_K (P j)$ ) x))
apply (rule n-eq-val-eq-idealTr, assumption+)
apply (rule allI, rule impI, simp)

apply (subgoal-tac ∀ j ≤ n. ( $\nu_K (P j)$ ) x ≤ (( $\nu_K (P j)$ ) y))
apply (rule n-eq-val-eq-idealTr, assumption+)
apply (rule allI, rule impI)
apply simp
done

definition
mI-gen :: [- , nat ⇒ ('r ⇒ ant) set, nat, 'r set] ⇒ 'r where
mI-gen K P n I = (SOME x. x ∈ I ∧

```

$$(\forall j \leq n. (\nu_{K(Pj)}) x = LI K (\nu_{K(Pj)}) I))$$

definition

$mL :: [-, nat \Rightarrow ('r \Rightarrow ant) set, 'r set, nat] \Rightarrow int$ **where**
 $mL K P I j = tna (LI K (\nu_{K(Pj)}) I)$

lemma (in Corps) mI-vals-nonempty: $\llbracket \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I; j \leq n \rrbracket \implies (\nu_{K(Pj)})^{\cdot} I \neq \{\}$

apply (frule ring-n-pd[of n P])

apply (frule Ring.ideal-zero [of $O_{K P n} I$], assumption+)

apply (simp add:image-def)

apply blast

done

lemma (in Corps) mI-vals-LB: $\llbracket \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I; j \leq n \rrbracket \implies ((\nu_{K(Pj)})^{\cdot} I) \subseteq LBset (ant 0)$

apply (rule subsetI)

apply (simp add:image-def, erule bxE)

apply (frule ring-n-pd[of n P])

apply (frule-tac $h = xa$ in Ring.ideal-subset[of $O_{K P n} I$], assumption+)

apply (thin-tac ideal $(O_{K P n}) I$)

apply (thin-tac Ring $(O_{K P n})$)

apply (simp add: ring-n-pd-def Sr-def) apply (erule conjE)+

apply (drule-tac $a = j$ in forall-spec, simp)

apply (simp add:LBset-def ant-0)

done

lemma (in Corps) mL-hom: $\llbracket \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I;$

$I \neq \{\mathbf{0}_{(O_{K P n})}\}; I \neq \text{carrier } (O_{K P n}) \rrbracket \implies$

$\forall j \leq n. mL K P I j \in Zset$

apply (rule allI, rule impI)

apply (simp add:mL-def LI-def)

apply (simp add:Zset-def)

done

lemma (in Corps) ex-Zleast-in-mI: $\llbracket \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I; j \leq n \rrbracket \implies \exists x \in I. (\nu_{K(Pj)}) x = LI K (\nu_{K(Pj)}) I$

apply (frule-tac $j = j$ in mI-vals-nonempty[of n P I], assumption+)

apply (frule-tac $j = j$ in mI-vals-LB[of n P I], assumption+)

apply (frule-tac $A = (\nu_{K(Pj)})^{\cdot} I$ and $z = 0$ in AMin-mem, assumption+)

apply (simp add:LI-def)

apply (thin-tac $(\nu_{K(Pj)})^{\cdot} I \subseteq LBset (ant 0)$)

apply (simp add:image-def, erule bxE)

apply (drule sym)

apply blast

done

```

lemma (in Corps) val-LI-pos: [| distinct-pds K n P; ideal (O_K P n) I;
  I ≠ {0_{O_K P n}}; j ≤ n] ==> 0 ≤ LI K (ν_K (P j)) I
apply (frule-tac j = j in mI-vals-nonempty[of n P I], assumption+)
apply (frule-tac j = j in mI-vals-LB[of n P I], assumption+)
apply (frule-tac A = (ν_K (P j)) ` I and z = 0 in AMin-mem, assumption+)
apply (simp add:LI-def)
apply (frule subsetD[of (ν_K (P j)) ` I LBset (ant 0) AMin ((ν_K (P j)) ` I)], as-
umption+)
apply (simp add:LBset-def ant-0)
done

lemma (in Corps) val-LI-noninf: [| distinct-pds K n P; ideal (O_K P n) I;
  I ≠ {0_{O_K P n}}; j ≤ n] ==> LI K (ν_K (P j)) I ≠ ∞
apply (frule-tac j = j in mI-vals-nonempty[of n P I], assumption+)
apply (frule-tac j = j in mI-vals-LB[of n P I], assumption+)
apply (frule-tac A = (ν_K (P j)) ` I and z = 0 in AMin, assumption+)
apply (thin-tac (ν_K (P j)) ` I ⊆ LBset (ant 0),
  thin-tac (ν_K (P j)) ` I ≠ {})
apply (frule ring-n-pd[of n P])
apply (frule Ring.ideal-zero[of O_K P n I], assumption+)
apply (erule conjE, simp add:LI-def)
apply (frule singleton-sub[of 0_{O_K P n} I])
apply (frule sets-not-eq[of I {0_{O_K P n}}],
  assumption+, erule bxE)
apply (simp add:zero-in-ring-n-pd-zero-K)
apply (subgoal-tac ∃ x ∈ I. AMin ((ν_K (P j)) ` I) = (ν_K (P j)) x,
  erule bxE) apply simp
apply (drule-tac x = a in bspec, assumption)
apply (thin-tac AMin ((ν_K (P j)) ` I) = (ν_K (P j)) x)

apply (frule-tac h = a in Ring.ideal-subset[of O_K P n I], assumption+)
apply (frule-tac x = a in mem-ring-n-pd-mem-K[of n P], assumption+)
apply (simp add:distinct-pds-def, (erule conjE)+)
apply (cut-tac representative-of-pd-valuation[of P j])
defer apply simp apply blast
apply (frule-tac x = a in val-nonzero-z[of ν_K (P j)], assumption+,
  erule exE, simp)
apply (thin-tac ∀ l ≤ n. ∀ m ≤ n. l ≠ m —> P l ≠ P m,
  thin-tac (ν_K (P j)) a = ant z)

apply (rule contrapos-pp, simp+)
apply (cut-tac x = ant z in inf-ge-any)
apply (frule-tac x = ant z in ale-antisym[of - ∞], assumption+)
apply simp
done

```

```

lemma (in Corps) Zleast-in-mI-pos: distinct-pds K n P; ideal (O_K P n) I;
  I ≠ {0_{(O_K P n)}}; j ≤ n] ⇒ 0 ≤ mL K P I j
apply (simp add:mL-def)
apply (frule ex-Zleast-in-mI[of n P I j], assumption+)
  erule bxE, frule sym, thin-tac (ν_K (P j)) x = LI K (ν_K (P j)) I
apply (subgoal-tac LI K (ν_K (P j)) I ≠ ∞, simp)
apply (thin-tac LI K (ν_K (P j)) I = (ν_K (P j)) x)

apply (frule ring-n-pd[of n P])
apply (frule-tac h = x in Ring.ideal-subset[of O_K P n I], assumption+)
apply (thin-tac ideal (O_K P n) I)
apply (thin-tac Ring (O_K P n))
apply (simp add: ring-n-pd-def Sr-def) apply (erule conjE)
apply (drule-tac a = j in forall-spec, assumption)
apply (simp add:apos-tna-pos)
apply (rule val-LI-noninf, assumption+)
done

lemma (in Corps) Zleast-mL-I: distinct-pds K n P; ideal (O_K P n) I; j ≤ n;
  I ≠ {0_{(O_K P n)}}; x ∈ I] ⇒ ant (mL K P I j) ≤ ((ν_K (P j)) x)
apply (frule val-LI-pos[of n P I j], assumption+)
apply (frule apos-neq-minf[of LI K (ν_K (P j)) I])
apply (frule val-LI-noninf[of n P I j], assumption+)
apply (simp add:mL-def LI-def)
apply (simp add:ant-tna)
apply (frule Zleast-in-mI-pos[of n P I j], assumption+)

apply (frule mI-vals-nonempty[of n P I j], assumption+)
apply (frule mI-vals-LB[of n P I j], assumption+)
apply (frule AMin[of (ν_K (P j)) 'I 0], assumption+)
apply (erule conjE)
apply (frule Zleast-in-mI-pos[of n P I j], assumption+)
apply (simp add:mL-def LI-def)
done

lemma (in Corps) Zleast-LI: distinct-pds K n P; ideal (O_K P n) I; j ≤ n;
  I ≠ {0_{(O_K P n)}}; x ∈ I] ⇒ (LI K (ν_K (P j)) I) ≤ ((ν_K (P j)) x)
apply (frule mI-vals-nonempty[of n P I j], assumption+)
apply (frule mI-vals-LB[of n P I j], assumption+)
apply (frule AMin[of (ν_K (P j)) 'I 0], assumption+)
apply (erule conjE)
apply (simp add:LI-def)
done

lemma (in Corps) mpdiv-vals-nonequiv: distinct-pds K n P ⇒
  vals-nonequiv K n (λj. ν_K (P j))
apply (simp add:vals-nonequiv-def)

```

```

apply (rule conjI)
apply (simp add:valuations-def)
apply (rule allI, rule impI)
apply (rule representative-of-pd-valuation,
       simp add:distinct-pds-def)
apply ((rule allI, rule impI)+, rule impI)
apply (simp add:distinct-pds-def, erule conjE)
apply (rotate-tac 4) apply (
  drule-tac a = j in forall-spec, assumption)
apply (rotate-tac -1,
       drule-tac a = l in forall-spec, assumption, simp)
apply (simp add:distinct-p-divisors)
done

definition
KbaseP :: [-, nat ⇒ ('r ⇒ ant) set, nat] ⇒
            (nat ⇒ 'r) ⇒ bool where
  KbaseP K P n f ↔ (∀ j ≤ n. f j ∈ carrier K) ∧
    (∀ j ≤ n. ∀ l ≤ n. (νK(P j)) (f l) = (δj l))

definition
Kbase :: [-, nat, nat ⇒ ('r ⇒ ant) set]
        ⇒ (nat ⇒ 'r) ((3Kb_ __) [95,95,96]95) where
  KbK n P = (SOME f. KbaseP K P n f)

lemma (in Corps) KbaseTr:distinct-pds K n P ==> ∃f. KbaseP K P n f
apply (simp add: KbaseP-def)
apply (frule mpdiv-vals-nonequiv[of n P])
apply (case-tac n = 0)
apply (simp add:vals-nonequiv-def valuations-def)
apply (simp add:distinct-pds-def)
apply (frule n-val-n-val1[of P 0])
apply (frule n-val-surj[of νK(P 0)])
apply (erule bexE)
apply (subgoal-tac ((λj∈{0::nat}. x) (0::nat)) ∈ carrier K ∧
  (νK(P 0)) ((λj∈{0::nat}. x) (0::nat)) = (δ0 0))
apply blast
apply (rule conjI)
apply simp apply (simp add:Kronecker-delta-def)
apply (cut-tac Approximation1-5[of n - Suc 0 λj. νK(P j)])
apply simp
apply simp+
apply (rule allI, rule impI)
apply (rule n-val-n-val1)
apply (simp add:distinct-pds-def)
done

lemma (in Corps) KbaseTr1:distinct-pds K n P ==> KbaseP K P n (KbK n P )
apply (subst Kbase-def)

```

```

apply (frule KbaseTr[of n P])
apply (erule exE)
apply (simp add:someI)
done

lemma (in Corps) Kbase-hom:distinct-pds K n P ==>
  ∀ j ≤ n. (KbK n P) j ∈ carrier K
apply (frule KbaseTr1[of n P])
apply (simp add:KbaseP-def)
done

lemma (in Corps) Kbase-Kronecker:distinct-pds K n P ==>
  ∀ j ≤ n. ∀ l ≤ n. (νK (P j)) ((KbK n P) l) = δj l
apply (frule KbaseTr1[of n P])
apply (simp add:KbaseP-def)
done

lemma (in Corps) Kbase-nonzero:distinct-pds K n P ==>
  ∀ j ≤ n. (KbK n P) j ≠ 0
apply (rule allI, rule impI)
apply (frule Kbase-Kronecker[of n P])
apply (subgoal-tac (νK (P j)) ((KbK n P) j) = δj j)
apply (thin-tac ∀ j≤n. (∀ l≤n. ((νK P j) ((KbK n P) l)) = δj l))
apply (simp add:Kronecker-delta-def)
apply (rule contrapos-pp, simp+)
apply (cut-tac P = P j in representative-of-pd-valuation)
apply (simp add:distinct-pds-def)
apply (simp only:value-of-zero, simp only:ant-1[THEN sym],
       frule sym, thin-tac ∞ = ant 1, simp del:ant-1)
apply simp
done

lemma (in Corps) Kbase-hom1:distinct-pds K n P ==>
  ∀ j ≤ n. (KbK n P) j ∈ carrier K - {0}
by(simp add:Kbase-nonzero Kbase-hom)

definition
  Zl-mI :: [-, nat ⇒ ('b ⇒ ant) set, 'b set]
  ⇒ nat ⇒ 'b where
  Zl-mI K P I j = (SOME x. (x ∈ I ∧ (νK (P j)) x = LI K (νK (P j)) I)))

```

lemma (in Corps) value-Zl-mI:distinct-pds K n P; ideal (O_{K P n}) I; j ≤ n] ==> (Zl-mI K P I j ∈ I) ∧ (νK (P j)) (Zl-mI K P I j) = LI K (νK (P j)) I

apply (subgoal-tac ∃ x. (x ∈ I ∧ ((νK (P j)) x = LI K (νK (P j)) I)))

apply (subst Zl-mI-def)+

apply (rule someI2-ex, assumption+)

apply (frule ex-Zleast-in-mI[of n P I j], assumption+)

apply (erule bxE, blast)

done

```

lemma (in Corps) Zl-mI-nonzero:⟦distinct-pds K n P; ideal (O_K P n) I;
  I ≠ {0_(O_K P n)}; j ≤ n⟧ ⇒ Zl-mI K P I j ≠ 0
apply (case-tac n = 0)
apply (simp add:distinct-pds-def)
apply (frule representative-of-pd-valuation[of P 0])
apply (subgoal-tac O_K P 0 = Vr K (ν_K (P 0)))
apply (subgoal-tac Zl-mI K P I 0 = Ig K (ν_K (P 0)) I)
apply simp apply (simp add:Ig-nonzero)
apply (simp add:Ig-def Zl-mI-def)
apply (simp add:ring-n-pd-def Vr-def)

apply (simp)
apply (frule value-Zl-mI[of n P I j], assumption+)
apply (erule conjE)
apply (rule contrapos-pp, simp+)
apply (frule distinct-pds-valuation1[of n j P], assumption+)
apply (simp add:value-of-zero)
apply (simp add:zero-in-ring-n-pd-zero-K)
apply (frule singleton-sub[of 0 I],
       frule sets-not-eq[of I {0}], assumption,
       erule bexE, simp)
apply (frule-tac x = a in Zleast-mL-I[of n P I j], assumption+)
apply (frule-tac x = a in val-nonzero-z[of ν_K (P j)])
apply (frule ring-n-pd[of n P])
apply (frule-tac h = a in Ring.ideal-subset[of O_K P n I], assumption+)
apply (simp add:mem-ring-n-pd-mem-K) apply assumption

apply (simp add:zero-in-ring-n-pd-zero-K) apply assumption
apply (frule val-LI-noninf[THEN not-sym, of n P I j], assumption+)
apply (simp add:zero-in-ring-n-pd-zero-K) apply assumption
apply simp
done

```

```

lemma (in Corps) Zl-mI-mem-K:⟦distinct-pds K n P; ideal (O_K P n) I; l ≤ n⟧
  ⇒ (Zl-mI K P I l) ∈ carrier K
apply (frule value-Zl-mI[of n P I l], assumption+)
apply (erule conjE)
apply (frule ring-n-pd[of n P])
apply (frule Ring.ideal-subset[of O_K P n I Zl-mI K P I l], assumption+)
apply (simp add:mem-ring-n-pd-mem-K[of n P Zl-mI K P I l])
done

```

definition

```

mprod-exp :: [-, nat ⇒ int, nat ⇒ 'b, nat]
  ⇒ 'b where
mprod-exp K e f n = nprod K (λj. ((f j)_K^(e j))) n

```

```

lemma (in Corps) mprod-expR-memTr:( $\forall j \leq n. f j \in \text{carrier } K$ )  $\rightarrow$ 
 $mprod\text{-}expR K e f n \in \text{carrier } K$ 
apply (cut-tac field-is-ring)
apply (induct-tac n)
apply (rule impI, simp)
apply (simp add:mprod-expR-def)
apply (cut-tac Ring.npClose[of K f 0 e 0], assumption+)

apply (rule impI)
apply simp
apply (subst Ring.mprodR-Suc, assumption+)
apply (simp)
apply (simp)
apply (rule Ring.ring-tOp-closed[of K], assumption+)
apply (rule Ring.npClose, assumption+)
apply simp
done

lemma (in Corps) mprod-expR-mem: $\forall j \leq n. f j \in \text{carrier } K \implies$ 
 $mprod\text{-}expR K e f n \in \text{carrier } K$ 
apply (cut-tac field-is-ring)
apply (cut-tac Ring.mprod-expR-memTr[of K e n f])
apply simp
apply (subgoal-tac  $f \in \{j. j \leq n\} \rightarrow \text{carrier } K$ , simp+)
done

lemma (in Corps) mprod-Suc: $\llbracket \forall j \leq (\text{Suc } n). e j \in Zset;$ 
 $\forall j \leq (\text{Suc } n). f j \in (\text{carrier } K - \{\mathbf{0}\}) \rrbracket \implies$ 
 $mprod\text{-}exp K e f (\text{Suc } n) = (mprod\text{-}exp K e f n) \cdot_r ((f (\text{Suc } n))_K^{(e (\text{Suc } n))})$ 
apply (simp add:mprod-exp-def)
done

lemma (in Corps) mprod-memTr:
 $(\forall j \leq n. e j \in Zset) \wedge (\forall j \leq n. f j \in ((\text{carrier } K) - \{\mathbf{0}\})) \rightarrow$ 
 $(mprod\text{-}exp K e f n) \in ((\text{carrier } K) - \{\mathbf{0}\})$ 
apply (induct-tac n)
apply (simp, rule impI, (erule conjE)+,
simp add:mprod-exp-def, simp add:npowf-mem,
simp add:field-potent-nonzero1)
apply (rule impI, simp, erule conjE,
cut-tac field-is-ring, cut-tac field-is-idom,
erule conjE, simp add:mprod-Suc)
apply (rule conjI)
apply (rule Ring.ring-tOp-closed[of K], assumption+,
simp add:npowf-mem)
apply (rule Idomain.idom-tOp-nonzeros, assumption+,
simp add:npowf-mem, assumption,
simp add:field-potent-nonzero1)

```

done

```
lemma (in Corps) mprod-mem:[ $\forall j \leq n. e j \in Zset; \forall j \leq n. f j \in ((carrier K) - \{\mathbf{0}\})$ ]  $\implies$  (mprod-exp K e f n)  $\in ((carrier K) - \{\mathbf{0}\})$ 
apply (cut-tac mprod-memTr[of n e f]) apply simp
done
```

```
lemma (in Corps) mprod-mprodR:[ $\forall j \leq n. e j \in Zset; \forall j \leq n. 0 \leq (e j); \forall j \leq n. f j \in ((carrier K) - \{\mathbf{0}\})$ ]  $\implies$ 
mprod-exp K e f n = mprod-expR K (nat o e) f n
apply (cut-tac field-is-ring)
apply (simp add:mprod-exp-def mprod-expR-def)
apply (rule Ring.nprod-eq, assumption+)
apply (rule allI, rule impI, simp add:npowf-mem)
apply (rule allI, rule impI, rule Ring.npClose, assumption+, simp)
apply (rule allI, rule impI)
apply (simp add:npowf-def)
done
```

2.8.1 Representation of an ideal I as a product of prime ideals

```
lemma (in Corps) ring-n-mprod-mprodRTr:distinct-pds K n P  $\implies$ 
( $\forall j \leq m. e j \in Zset$ )  $\wedge$  ( $\forall j \leq m. 0 \leq (e j)$ )  $\wedge$ 
( $\forall j \leq m. f j \in carrier (O_{K P n} - \{\mathbf{0}_{(O_{K P n})}\})$ )  $\longrightarrow$ 
mprod-exp K e f m = mprod-expR (O_{K P n}) (nat o e) f m
apply (frule ring-n-pd[of n P])
apply (induct-tac m)
apply (rule impI, (erule conjE)+,
simp add:mprod-exp-def mprod-expR-def)
apply (erule conjE, simp add:npowf-def, simp add:ring-n-exp-K-exp)

apply (rule impI, (erule conjE)+, simp)
apply (subst mprod-Suc, assumption+,
rule allI, rule impI,
simp add:mem-ring-n-pd-mem-K,
simp add:zero-in-ring-n-pd-zero-K)
apply (subst Ring.mprodR-Suc, assumption+,
simp add:cmp-def,
simp)
apply (simp add:ring-n-pd, simp add:npowf-def,
simp add:ring-n-exp-K-exp)
apply (subst ring-n-tOp-K-tOp, assumption+,
rule Ring.mprod-expR-mem, simp add:ring-n-pd,
simp,
simp)
apply (rule Ring.npClose, simp add:ring-n-pd, simp, simp)
done
```

```

lemma (in Corps) ring-n-mprod-mprodR:[[distinct-pds K n P;  $\forall j \leq m. e j \in Zset;$ 
 $\forall j \leq m. 0 \leq (e j); \forall j \leq m. f j \in carrier (O_{K P n}) - \{\mathbf{0}_{(O_{K P n})}\}]$ ]
 $\implies mprod-exp K e f m = mprod-expR (O_{K P n}) (nat o e) f m$ 
apply (simp add:ring-n-mprod-mprodRTr)
done

lemma (in Corps) value-mprod-expTr:valuation K v  $\implies$ 
 $(\forall j \leq n. e j \in Zset) \wedge (\forall j \leq n. f j \in (carrier K - \{\mathbf{0}\})) \rightarrow$ 
 $v (mprod-exp K e f n) = ASum (\lambda j. (e j) *_a (v (f j))) n$ 
apply (induct-tac n)
apply simp
apply (rule impI, erule conjE)
apply(simp add:mprod-exp-def val-exp)

apply (rule impI, erule conjE)
apply simp
apply (subst mprod-Suc, assumption+)
apply (rule allI, rule impI, simp)
apply (subst val-t2p[of v], assumption+)
apply (cut-tac n = n in mprod-mem[of - e f],
 $(rule allI, rule impI, simp)+, simp)$ 
apply (simp add:npowf-mem, simp add:field-potent-nonzero1)
apply (simp add:val-exp[THEN sym, of v])
done

lemma (in Corps) value-mprod-exp:[[valuation K v;  $\forall j \leq n. e j \in Zset;$ 
 $\forall j \leq n. f j \in (carrier K - \{\mathbf{0}\})] \implies$ 
 $v (mprod-exp K e f n) = ASum (\lambda j. (e j) *_a (v (f j))) n$ 
apply (simp add:value-mprod-expTr)
done

lemma (in Corps) mgenerator0-1:[[distinct-pds K (Suc n) P;
ideal (O_{K P (Suc n)}) I;  $I \neq \{\mathbf{0}_{(O_{K P (Suc n)})}\};$ 
 $I \neq carrier (O_{K P (Suc n)}); j \leq (Suc n)] \implies$ 
 $((\nu_{K (P j)}) (mprod-exp K (mL K P I) (Kb_{K (Suc n) P} (Suc n))) =$ 
 $((\nu_{K (P j)}) (Zl-mI K P I j))$ 
apply (frule distinct-pds-valuation[of j n P], assumption+)
apply (frule mL-hom[of Suc n P I], assumption+)
apply (frule Kbase-hom1[of Suc n P])
apply (frule value-mprod-exp[of \nu_{K (P j)} Suc n mL K P I
 $Kb_{K (Suc n) P}], assumption+)$ 

apply (simp del:ASum-Suc)
apply (thin-tac (\nu_{K (P j)}) (mprod-exp K (mL K P I) (Kb_{K (Suc n) P} (Suc n)))
 $= ASum (\lambda ja. (mL K P I ja) *_a (\nu_{K (P j)}) ((Kb_{K (Suc n) P} ja)) (Suc n))$ 
apply (subgoal-tac ASum (\lambda ja. (mL K P I ja) *_a
 $((\nu_{K (P j)}) ((Kb_{K (Suc n) P} ja)))) (Suc n) =$ 

```

```

ASum (λja. (mL K P I ja) *a (δj ja)) (Suc n))
apply (simp del:ASum-Suc)
apply (subgoal-tac ∀ h ≤ (Suc n). (λja. (mL K P I ja) *a (δj ja)) h ∈ Z∞)
apply (cut-tac eSum-single[of Suc n λja. (mL K P I ja) *a (δj ja) j])
apply simp
apply (simp add:Kronecker-delta-def asprod-n-0)
apply (rotate-tac -1, drule not-sym)
apply (simp add:mL-def[of K P I j])

apply (frule val-LI-noninf[of Suc n P I j], assumption+)
apply (rule not-sym, simp, simp)
apply (frule val-LI-pos[of Suc n P I j], assumption+,
      rotate-tac -2, frule not-sym, simp, simp)

apply (frule apos-neq-minf[of LI K (νK(P j)) I])
apply (simp add:ant-tna)
apply (simp add:value-Zl-mI[of Suc n P I j])
apply (rule allI, rule impI)
apply (simp add:Kdelta-in-Zinf, simp)
apply (rule ballI, simp)
apply (simp add:Kronecker-delta-def, erule conjE)
apply (simp add:asprod-n-0)

apply (rule allI, rule impI)
apply (simp add:Kdelta-in-Zinf)

apply (frule Kbase-Kronecker[of Suc n P])
apply (rule ASum-eq,
       rule allI, rule impI,
       simp add:Kdelta-in-Zinf,
       rule allI, rule impI,
       simp add:Kdelta-in-Zinf)
apply (rule allI, rule impI) apply simp
done

lemma (in Corps) mgenerator0-2:[ 0 < n; distinct-pds K n P; ideal (OK P n) I;
I ≠ {0(O_K P n)}; I ≠ carrier (OK P n); j ≤ n] ⇒
((νK (P j)) (mprod-exp K (mL K P I) (KbK n P n)) = ((νK (P j)) (Zl-mI K P I j)))
apply (cut-tac mgenerator0-1[of n - Suc 0 P I j])
apply simp+
done

lemma (in Corps) mgenerator1:[distinct-pds K n P; ideal (OK P n) I;
I ≠ {0(O_K P n)}; I ≠ carrier (OK P n); j ≤ n] ⇒
((νK (P j)) (mprod-exp K (mL K P I) (KbK n P n)) = ((νK (P j)) (Zl-mI K P I j)))
apply (case-tac n = 0,

```

```

frule value-Zl-mI[of n P I j], assumption+,
frule val-LI-noninf[of n P I j], assumption+,
frule val-LI-pos[of n P I j], assumption+,
frule apos-neq-minf[of LI K ( $\nu_K(P_j)$ ) I],
simp add:distinct-pds-def, erule conjE)
apply (cut-tac representative-of-pd-valuation[of P j], simp+,
simp add:mprod-exp-def,
subst val-exp[THEN sym, of  $\nu_K(P_0)$  ( $Kb_K(0_P)$ ) 0], assumption+,
cut-tac Kbase-hom[of 0 P], simp,
simp add:distinct-pds-def,
cut-tac Kbase-nonzero[of 0 P], simp+,
simp add:distinct-pds-def)
apply (cut-tac Kbase-nonzero[of 0 P], simp add:distinct-pds-def)
apply (cut-tac Kbase-Kronecker[of 0 P], simp add:distinct-pds-def)
apply (simp add:Kronecker-delta-def, simp add:mL-def, simp add:ant-tna)
apply (simp add:distinct-pds-def)+
apply (cut-tac mgenerator0-2[of n P I j], simp+)
apply (simp add:distinct-pds-def) apply simp+
done

lemma (in Corps) mgenerator2Tr1: [| 0 < n; j ≤ n; k ≤ n; distinct-pds K n P |] ==>
 $(\nu_K(P_j)) (mprod-exp K (\lambda l. \gamma_{k l}) (Kb_K n P) n) = (\gamma_{k j}) *_a (\delta_{j j})$ 
apply (frule distinct-pds-valuation1[of n j P], assumption+)
apply (frule K-gamma-hom[of k n])
apply (subgoal-tac ∀ j ≤ n. (Kb_K n P) j ∈ carrier K - {0})
apply (simp add:value-mprod-exp[of ν_K(P_j) n K-gamma k (Kb_K n P)])
apply (subgoal-tac ASum (λ ja. ( $\gamma_{k ja}$ ) *a ( $\nu_K(P_j)$ ) ((Kb_K n P) ja)) n
= ASum (λ ja. ((( $\gamma_{k ja}$ ) *a ( $\delta_{j ja}$ )))) n)
apply simp
apply (subgoal-tac ∀ j ≤ n. ( $\lambda ja.$  ( $\gamma_{k ja}$ ) *a ( $\delta_{j ja}$ ))) j ∈ Z∞)
apply (cut-tac eSum-single[of n λ ja. (( $\gamma_{k ja}$ ) *a ( $\delta_{j ja}$ )) j], simp)
apply (rule allI, rule impI, simp add:Kronecker-delta-def,
rule impI, simp add:asprod-n-0 Zero-in-aug-inf, assumption+)
apply (rule ballI, simp)
apply (simp add:K-gamma-def, rule impI, simp add:Kronecker-delta-def)
apply (rule allI, rule impI)
apply (simp add:Kronecker-delta-def, simp add:K-gamma-def)
apply (simp add:ant-0 Zero-in-aug-inf)
apply (cut-tac z-in-aug-inf[of 1], simp add:ant-1)

apply (rule ASum-eq)
apply (rule allI, rule impI)
apply (simp add:K-gamma-def, simp add:Zero-in-aug-inf)
apply (rule impI, rule value-in-aug-inf, assumption+, simp)
apply (simp add:K-gamma-def Zero-in-aug-inf Kdelta-in-Zinf1)
apply (rule allI, rule impI)
apply (simp add:Kbase-Kronecker[of n P])
apply (rule Kbase-hom1, assumption+)

```

done

lemma (in Corps) mgenerator2Tr2: $\llbracket 0 < n; j \leq n; k \leq n; \text{distinct-pds } K n P \rrbracket \implies (\nu_K(P_j)) ((\text{mprod-exp } K (\lambda l. \gamma_k l) (Kb_{K n P} n)_K^m) = \text{ant}(m * (\gamma_k j)))$

apply (frule K-gamma-hom[of k n])
apply (frule Kbase-hom1[of n P])
apply (frule mprod-mem[of n K-gamma k Kb_{K n P}], assumption+)
apply (frule distinct-pds-valuation1[of n j P], assumption+)
apply (simp, erule conjE)
apply (simp add:val-exp[THEN sym])
apply (simp add:mgenerator2Tr1)
apply (simp add:K-gamma-def Kronecker-delta-def)
apply (rule impI)
apply (simp add:asprod-def a-z-z)
done

lemma (in Corps) mgenerator2Tr3-1: $\llbracket 0 < n; j \leq n; k \leq n; j = k; \text{distinct-pds } K n P \rrbracket \implies (\nu_K(P_j)) ((\text{mprod-exp } K (\lambda l. (\gamma_k l)) (Kb_{K n P} n)_K^m) = 0)$
apply (simp add:mgenerator2Tr2) apply (simp add:K-gamma-def)
done

lemma (in Corps) mgenerator2Tr3-2: $\llbracket 0 < n; j \leq n; k \leq n; j \neq k; \text{distinct-pds } K n P \rrbracket \implies (\nu_K(P_j)) ((\text{mprod-exp } K (\lambda l. (\gamma_k l)) (Kb_{K n P} n)_K^m) = \text{ant } m)$
apply (simp add:mgenerator2Tr2) apply (simp add:K-gamma-def)
done

lemma (in Corps) mgeneratorTr4: $\llbracket 0 < n; \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I; I \neq \{\mathbf{0}_{O_{K P n}}\}; I \neq \text{carrier } (O_{K P n}) \rrbracket \implies (\text{mprod-exp } K (mL K P I) (Kb_{K n P} n) \in \text{carrier } (O_{K P n}))$
apply (subst ring-n-pd-def)
apply (simp add:Sr-def)
apply (frule mL-hom[of n P I], assumption+)
apply (frule mprod-mem[of n mL K P I Kb_{K n P}])
apply (rule Kbase-hom1, assumption+)

apply (simp add:mprod-mem)
apply (rule allI, rule impI)
apply (simp add:mgenerator1)
apply (simp add:value-Zl-mI)
apply (simp add:val-LI-pos)
done

definition

m-zmax-pdsI-hom :: $[-, \text{nat} \Rightarrow ('b \Rightarrow \text{ant}) \text{ set}, 'b \text{ set}] \Rightarrow \text{nat} \Rightarrow \text{int}$ **where**
 $m\text{-zmax-pdsI-hom } K P I = (\lambda j. \text{tna}(\text{AMin}((\nu_K(P_j)) ` I)))$

definition

m-zmax-pdsI :: [-, nat, nat \Rightarrow ('b \Rightarrow ant) set, 'b set] \Rightarrow int **where**
 $m\text{-zmax-pdsI } K n P I = (m\text{-zmax } n \text{ (} m\text{-zmax-pdsI-hom } K P I \text{)}) + 1$

lemma (in Corps) *value-Zl-mI-pos*: $\llbracket 0 < n; \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I; I \neq \{\mathbf{0}_{(O_{K P n})}\}; I \neq \text{carrier } (O_{K P n}); j \leq n; l \leq n \rrbracket \implies 0 \leq ((\nu_{K (P j)}) (\text{Zl-mI } K P I l))$
apply (*frule value-Zl-mI*[*of n P I l*], *assumption+*)
apply (*erule conjE*)
apply (*frule ring-n-pd*[*of n P*])
apply (*frule Ring.ideal-subset*[*of O_{K P n} I Zl-mI K P I l*], *assumption+*)
apply (*thin-tac ideal* (*O_{K P n}*) *I*)
apply (*thin-tac I* $\neq \{\mathbf{0}_{O_{K P n}}\}$)
apply (*thin-tac I* $\neq \text{carrier } (O_{K P n})$)
apply (*thin-tac Ring* (*O_{K P n}*))
apply (*simp add:ring-n-pd-def Sr-def*)
done

lemma (in Corps) *value-mI-genTr1*: $\llbracket 0 < n; \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I; I \neq \{\mathbf{0}_{O_{K P n}}\}; I \neq \text{carrier } (O_{K P n}); j \leq n \rrbracket \implies (mprod-exp K (K-gamma j) (Kb_{K n P} n)_K^{(m\text{-zmax-pdsI } K n P I)} \in \text{carrier } K)$
apply (*frule K-gamma-hom*[*of j n*])
apply (*frule mprod-mem*[*of n K-gamma j Kb_{K n P}*])
apply (*rule Kbase-hom1*, *assumption+*)
apply (*rule npowf-mem*)
apply *simp+*
done

lemma (in Corps) *value-mI-genTr1-0*: $\llbracket 0 < n; \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I; I \neq \{\mathbf{0}_{O_{K P n}}\}; I \neq \text{carrier } (O_{K P n}); j \leq n \rrbracket \implies (mprod-exp K (K-gamma j) (Kb_{K n P} n) \in \text{carrier } K)$
apply (*frule K-gamma-hom*[*of j n*])
apply (*frule mprod-mem*[*of n K-gamma j Kb_{K n P}*])
apply (*rule Kbase-hom1*, *assumption+*)
apply *simp*
done

lemma (in Corps) *value-mI-genTr2*: $\llbracket 0 < n; \text{distinct-pds } K n P; \text{ideal } (O_{K P n}) I; I \neq \{\mathbf{0}_{O_{K P n}}\}; I \neq \text{carrier } (O_{K P n}); j \leq n \rrbracket \implies (mprod-exp K (K-gamma j) (Kb_{K n P} n)_K^{(m\text{-zmax-pdsI } K n P I)} \neq \mathbf{0})$
apply (*frule K-gamma-hom*[*of j n*])
apply (*frule mprod-mem*[*of n K-gamma j Kb_{K n P}*])
apply (*rule Kbase-hom1*, *assumption+*) **apply** *simp* **apply** (*erule conjE*)
apply (*simp add: field-potent-nonzero1*)

done

```

lemma (in Corps) value-mI-genTr3: [| 0 < n; distinct-pds K n P; ideal (O_K P n)
I;
I ≠ {0_O_K P n}; I ≠ carrier (O_K P n); j ≤ n] ==>
(Zl-mI K P I j) ·_r ((mprod-exp K (K-gamma j) (Kb_K n P) n)_K^{(m-zmax-pdsI K n P I)})
∈ carrier K
apply (cut-tac field-is-ring)
apply (rule Ring.ring-tOp-closed, assumption+)
apply (simp add:Zl-mI-mem-K)
apply (simp add:value-mI-genTr1)
done

lemma (in Corps) value-mI-gen: [| 0 < n; distinct-pds K n P; ideal (O_K P n) I;
I ≠ {0_O_K P n}; I ≠ carrier (O_K P n); j ≤ n] ==>
(ν_K (P j)) (nsum K (λk. ((Zl-mI K P I k) ·_r ((mprod-exp K (λl. (γ_k l)) (Kb_K n P)
n)_K^{(m-zmax-pdsI K n P I)}))) n) = LI K (ν_K (P j)) I
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (case-tac j = n, simp)
apply (cut-tac nsum-suc[of K λk. Zl-mI K P I k ·_r
mprod-exp K (K-gamma k) (Kb_K n P) n_K^{m-zmax-pdsI K n P I} n - Suc 0],
simp,
thin-tac Σ_e K (λk. Zl-mI K P I k ·_r
mprod-exp K (K-gamma k) (Kb_K n P) n_K^{m-zmax-pdsI K n P I}) n =
Σ_e K (λk. Zl-mI K P I k ·_r
mprod-exp K (K-gamma k) (Kb_K n P)
n_K^{m-zmax-pdsI K n P I} (n - Suc 0) ±
Zl-mI K P I n ·_r
mprod-exp K (K-gamma n) (Kb_K n P) n_K^{m-zmax-pdsI K n P I})
apply (cut-tac distinct-pds-valuation[of n n - Suc 0 P])
prefer 2 apply simp
prefer 2 apply simp
apply (subst value-less-eq1[THEN sym, of ν_K (P n)
(Zl-mI K P I n) ·_r (mprod-exp K (K-gamma n) (Kb_K n P) n_K^{m-zmax-pdsI K n P I})
nsum K (λk. (Zl-mI K P I k) ·_r (mprod-exp K (K-gamma k) (Kb_K n P) n_K^{m-zmax-pdsI K n P I}))
(n - Suc 0)], assumption+)

apply (simp add:value-mI-genTr3)
apply (frule Ring.ring-is-ag[of K])
apply (rule aGroup.nsum-mem[of - n - Suc 0], assumption+)
apply (rule allI, rule impI)
apply (simp add:value-mI-genTr3)

apply (subst val-t2p[of ν_K (P n)], assumption+)
apply (simp add:Zl-mI-mem-K)
apply (simp add:value-mI-genTr1)

```

```

apply (simp add:mgenerator2Tr3-1[of n n n P])
apply (simp add:aadd-0-r)
apply (frule value-Zl-mI[of n P I n], assumption+, simp)
apply (erule conjE)
apply (frule-tac f = λk. (Zl-mI K P I k) ·r
      (mprod-exp K (K-gamma k) (KbK n P) nKm-zmax-pdsI K n P I) in
      value-ge-add[of νK (P n) n - Suc 0 -
      ant (m-zmax-pdsI K n P I)])
apply (rule allI, rule impI)
apply (rule Ring.ring-tOp-closed, assumption+)
apply (simp add:Zl-mI-mem-K)
apply (simp add:value-mI-genTr1)

apply (rule allI, rule impI) apply (simp add:cmp-def)
apply (subst val-t2p [where v=νK P n], assumption+)
apply (simp add:Zl-mI-mem-K)
apply (simp add:value-mI-genTr1)

apply (cut-tac e = K-gamma ja in mprod-mem[of n - KbK n P])
apply (simp add:Zset-def) apply (rule Kbase-hom1, assumption+)
apply (subst val-exp[of νK (P n), THEN sym], assumption+)
apply simp+

apply (subst mgenerator2Tr1[of n n - P], assumption+, simp, simp,
       assumption+)
apply (simp add:K-gamma-def Kronecker-delta-def)
apply (frule-tac l = ja in value-Zl-mI-pos[of n P I n],
       assumption+, simp, simp)
apply (simp add:Nset-preTr1)
apply (frule-tac y = (νK (P n)) (Zl-mI K P I ja) in
       aadd-le-mono[of 0 - ant (m-zmax-pdsI K n P I)]) apply (simp add:aadd-0-l)
apply (subgoal-tac LI K (νK (P n)) I < ant (m-zmax-pdsI K n P I))
apply simp
apply (rule aless-le-trans[of LI K (νK (P n)) I
       ant (m-zmax-pdsI K n P I)])
apply (simp add:m-zmax-pdsI-def)
apply (cut-tac aless-zless[of tna (LI K (νK (P n)) I)
       m-zmax n (m-zmax-pdsI-hom K P I) + 1])
apply (frule val-LI-noninf[of n P I n], assumption+, simp, simp)
apply (frule val-LI-pos[of n P I n], assumption+, simp,
       frule apos-neq-minf[of LI K (νK (P n)) I], simp add:ant-tna)
apply (subst m-zmax-pdsI-hom-def)
apply (subst LI-def)
apply (cut-tac m-zmax-gt-each[of n λu.(tna (AMin ((νK (P u)) ` I)))])
apply simp

apply (rule allI, rule impI)

```

```

apply (simp add:Zset-def, simp)
apply (subst val-t2p[of  $\nu_K(P n)$ ], assumption+)
apply (rule Zl-mI-mem-K, assumption+, simp)
apply (simp add:value-mI-genTr1)
apply (simp add:mgenerator2Tr3-1[of n n n P m-zmax-pdsI K n P I])
apply (simp add:aadd-0-r)
apply (simp add:value-Zl-mI[of n P I n])

apply (frule aGroup.addition3[of K n - Suc 0  $\lambda k. (Zl-mI K P I k) \cdot_r ((mprod-exp K (K\gamma k) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)}) j)]$ )

apply simp
apply (rule allI, rule impI)
apply (simp add:value-mI-genTr3) apply simp+

apply (thin-tac  $\Sigma_e K (\lambda k. Zl-mI K P I k) \cdot_r mprod-exp K (K\gamma k) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)}$   $n = \Sigma_e K (\text{cmp } (\lambda k. Zl-mI K P I k) \cdot_r mprod-exp K (K\gamma k) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)} (\tau_j n)) n$ )
apply (cut-tac nsum-suc[of K cmp (\lambda k. Zl-mI K P I k) \cdot_r mprod-exp K (K\gamma k) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)} (\tau_j n)] n - Suc 0])
apply (simp del:nsum-suc) apply (thin-tac  $\Sigma_e K (\text{cmp } (\lambda k. Zl-mI K P I k) \cdot_r mprod-exp K (K\gamma k) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)} (\tau_j n)) n = \Sigma_e K (\text{cmp } (\lambda k. Zl-mI K P I k) \cdot_r mprod-exp K (K\gamma k) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)} (\tau_j n)) (n - Suc 0) \pm (\text{cmp } (\lambda k. Zl-mI K P I k) \cdot_r mprod-exp K (K\gamma k) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)} (\tau_j n)) n$ )
apply (cut-tac distinct-pds-valuation[of j n - Suc 0 P])
prefer 2 apply simp prefer 2 apply simp
apply (simp add:cmp-def)

apply (cut-tac n-in-Nsetn[of n])
apply (simp add:transpos-ij-2)
apply (subst value-less-eq1[THEN sym, of  $\nu_K(P j)$ ])
 $(Zl-mI K P I j) \cdot_r (mprod-exp K (K\gamma j) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)}) \Sigma_e K (\lambda x. (Zl-mI K P I ((\tau_j n) x)) \cdot_r (mprod-exp K (K\gamma ((\tau_j n) x)) (Kb_{K n} P) n_K^{(m-zmax-pdsI K n P I)})) (n - Suc 0)], assumption+)$ 
apply (simp add:value-mI-genTr3)
apply (rule aGroup.nsum-mem[of K n - Suc 0], assumption+)
apply (rule allI, rule impI)
apply (frule-tac l = ja in transpos-mem[of j n n], simp+)
apply (simp add:value-mI-genTr3)

```

```

apply (subst val-t2p[of  $\nu_K(P_j)$ ], assumption+)
apply (simp add:Zl-mI-mem-K)
apply (simp add:value-mI-genTr1)

apply (simp add:mgenerator2Tr3-1[of n j j P])

apply (frule value-Zl-mI[of n P I j], assumption+)
apply (erule conjE)
apply (simp add:aadd-0-r)
apply (cut-tac f =  $\lambda x. (Zl\text{-}mI K P I ((\tau_j n) x)) \cdot_r$ 
      (mprod-exp K (K-gamma ((\tau_j n) x)) (Kb_K n P)  $n_K^{m\text{-}zmax\text{-}pdsI K n P I}$ ) in
      value-ge-add[of  $\nu_K(P_j)$ ]
      n - Suc 0 - ant (m-zmax-pdsI K n P I)], assumption+)
apply (rule allI, rule impI)
apply (frule-tac l = ja in transpos-mem[of j n n], simp+)
apply (simp add:value-mI-genTr3)
apply (rule allI, rule impI) apply (simp add:cmp-def)

apply (frule-tac l = ja in transpos-mem[of j n n], simp+)

apply (subst val-t2p [where v= $\nu_K(P_j)$ ], assumption+)
apply (simp add:Zl-mI-mem-K)
apply (simp add:value-mI-genTr1)
apply (cut-tac k = ja in transpos-noteqTr[of n - j], simp+)
apply (subst mgenerator2Tr3-2[of n j - P], simp+)
apply (cut-tac l = ( $\tau_j n$ ) ja in value-Zl-mI-pos[of n P I j],
      simp+)
apply (frule-tac y = ( $\nu_K(P_j)$ ) (Zl-mI K P I ((\tau_j n) ja)) in
      aadd-le-mono[of 0 - ant (m-zmax-pdsI K n P I)])
apply (simp add:aadd-0-l)
apply (subgoal-tac LI K ( $\nu_K(P_j)$ ) I < ant (m-zmax-pdsI K n P I))
apply (rule aless-le-trans[of LI K ( $\nu_K(P_j)$ ) I
      ant (m-zmax-pdsI K n P I)], assumption+)

apply (simp add:m-zmax-pdsI-def)
apply (cut-tac aless-zless[of tna (LI K ( $\nu_K(P_j)$ ) I)
      m-zmax n (m-zmax-pdsI-hom K P I) + 1])
apply (frule val-LI-noninf[of n P I j], assumption+,
      frule val-LI-pos[of n P I j], assumption+,
      frule apos-neq-minf[of LI K ( $\nu_K(P_j)$ ) I], simp add:ant-tna)
apply (subst m-zmax-pdsI-hom-def)
apply (subst LI-def)
apply (subgoal-tac  $\forall h \leq n. (\lambda u. (tna (AMin ((\nu_K(P_u)) \cdot I)))) h \in Zset$ )
apply (frule m-zmax-gt-each[of n  $\lambda u. (tna (AMin ((\nu_K(P_u)) \cdot I)))$ ])
apply simp
apply (rule allI, rule impI)
apply (simp add:Zset-def)

```

```

apply (subst val-t2p[of  $\nu_K(P_j)$ ], assumption+)
apply (rule Zl-mI-mem-K, assumption+)
apply (simp add:value-mI-genTr1)

apply (simp add:mgenerator2Tr3-1[of  $n j j P$ 
                                 $m\text{-zmax-pdsI } K n P I$ ])
apply (simp add:aadd-0-r)
apply (simp add:value-Zl-mI[of  $n P I j$ ])
done

lemma (in Corps) mI-gen-in-I:[ $0 < n$ ; distinct-pds  $K n P$ ; ideal  $(O_K P_n) I$ ;
 $I \neq \{0_{(O_K P_n)}\}$ ;  $I \neq \text{carrier } (O_K P_n)$ ]  $\implies$ 
  ( $\text{nsum } K (\lambda k. ((\text{Zl-mI } K P I k) \cdot_r$ 
    $((mprod-exp K (\lambda l. (\gamma_k l)) (Kb_{K n P} n)_K^{(m\text{-zmax-pdsI } K n P I)}))) n) \in I$ )
apply (cut-tac field-is-ring, frule ring-n-pd[of  $n P$ ])
apply (rule ideal-eSum-closed[of  $n P I n$ ], assumption+)
apply (rule allI, rule impI)
apply (frule-tac  $j = j$  in value-Zl-mI[of  $n P I$ ], assumption+)
apply (erule conjE)
apply (thin-tac  $(\nu_K(P_j)) (\text{Zl-mI } K P I j) = LI K (\nu_K(P_j)) I$ )
apply (subgoal-tac (mprod-exp K (K-gamma j) (Kb_{K n P} n)_K^{(m\text{-zmax-pdsI } K n P I)}
                      $\in \text{carrier } (O_K P_n)})$ 
apply (frule-tac  $x = \text{Zl-mI } K P I j$  and
   $r = (mprod-exp K (K-gamma j) (Kb_{K n P} n)_K^{(m\text{-zmax-pdsI } K n P I)}$ 
  in Ring.ideal-ring-multiple1[of  $(O_K P_n) I$ ], assumption+)
apply (frule-tac  $h = \text{Zl-mI } K P I j$  in
  Ring.ideal-subset[of  $O_K P_n I$ ], assumption+)
apply (simp add:ring-n-pd-tOp-K-tOp[of  $n P$ ])

apply (subst ring-n-pd-def) apply (simp add:Sr-def)
apply (simp add:value-mI-genTr1)

apply (rule allI, rule impI)
apply (case-tac  $j = ja$ )
apply (simp add:mgenerator2Tr3-1)

apply (simp add:mgenerator2Tr3-2)
apply (simp add:m-zmax-pdsI-def) apply (simp add:m-zmax-pdsI-hom-def)
apply (simp only:ant-0[THEN sym])
apply (simp add:aless-zless)
apply (subgoal-tac  $\forall l \leq n. (\lambda j. tna (AMin ((\nu_K(P_j)) ` I))) l \in Zset$ )
apply (frule m-zmax-gt-each[of  $n \lambda j. tna (AMin ((\nu_K(P_j)) ` I)))]$ )
apply (rotate-tac -1, drule-tac  $a = ja$  in forall-spec, simp+)
apply (frule-tac  $j = ja$  in val-LI-pos[of  $n P I$ ], assumption+)
apply (cut-tac  $j = tna (LI K (\nu_K(P_ja)) I)$  in ale-zle[of 0])
apply (frule-tac  $j = ja$  in val-LI-noninf[of  $n P I$ ], assumption+,
       frule-tac  $j = ja$  in val-LI-pos[of  $n P I$ ], assumption+,
       frule-tac  $a = LI K (\nu_K(P_ja)) I$  in apos-neq-minf, simp add:ant-tna,

```

```

simp add:ant-0) apply (unfold LI-def)
apply (frule-tac y = tna (AMin (νK(Pja) ` I)) and z = m-zmax n (λj. tna
(AMin (νK(Pj) ` I))) in order-trans[of 0], assumption+)
apply (rule-tac y = m-zmax n (λj. tna (AMin (νK(Pj) ` I))) and
z = m-zmax n (λj. tna (AMin (νK(Pj) ` I))) + 1 in order-trans[of 0],
assumption+) apply simp

apply (rule allI, rule impI) apply (simp add:Zset-def)
done

```

We write the element $e\Sigma K (\lambda k. (Zl\text{-}mI K P I k) \cdot_K ((mprod-exp K (K\text{-}gamma k) (Kb_{K n P}) n)_K^{(m\text{-}zmax-pdsI K n P I)})) n$ as $mIg_{K G a i n P I}$
definition

```

mIg :: [-, nat, nat ⇒ ('b ⇒ ant) set,
'b set] ⇒ 'b ((4mIg _ _ _ _) [82,82,82,83]82) where
mIg_{K n P I} = Σe K (\lambda k. (Zl\text{-}mI K P I k) \cdot_r K
((mprod-exp K (K\text{-}gamma k) (Kb_{K n P}) n)_K^{(m\text{-}zmax-pdsI K n P I)})) n

```

We can rewrite above two lemmas by using $mIg_{K G a i n P I}$

```

lemma (in Corps) value-mI-gen1:[0 < n; distinct-pds K n P; ideal (OK P n) I;
I ≠ {0(O K P n)} ; I ≠ carrier (OK P n)] ⇒
∀j ≤ n. (νK (P j)) (mIgK n P I) = LI K (νK (P j)) I
apply (rule allI, rule impI)
apply (simp add:mIg-def value-mI-gen)
done

```

```

lemma (in Corps) mI-gen-in-I1:[0 < n; distinct-pds K n P; ideal (OK P n) I;
I ≠ {0(O K P n)} ; I ≠ carrier (OK P n)] ⇒ (mIgK n P I) ∈ I
apply (simp add:mIg-def mI-gen-in-I)
done

```

```

lemma (in Corps) mI-principalTr:[0 < n; distinct-pds K n P; ideal (OK P n) I;
I ≠ {0(O K P n)} ; I ≠ carrier (OK P n); x ∈ I] ⇒
∀j ≤ n. ((νK (P j)) (mIgK n P I)) ≤ ((νK (P j)) x)
apply (simp add:value-mI-gen1)
apply (rule allI, rule impI)
apply (rule Zleast-LI, assumption+)
done

```

```

lemma (in Corps) mI-principal:[0 < n; distinct-pds K n P; ideal (OK P n) I;
I ≠ {0(O K P n)} ; I ≠ carrier (OK P n)] ⇒
I = Rxa (OK P n) (mIgK n P I)
apply (frule ring-n-pd[of n P])
apply (rule equalityI)
apply (rule subsetI)
apply (frule-tac x = x in mI-principalTr[of n P I],
assumption+)

```

```

apply (frule-tac y = x in n-eq-val-eq-idealTr[of n P mIg_K n P I])
apply (frule mI-gen-in-I1[of n P I], assumption+)
apply (simp add:Ring.ideal-subset)+
apply (thin-tac  $\forall j \leq n. (\nu_K(P_j)) (mIg_K n P I) \leq (\nu_K(P_j)) x$ )
apply (frule-tac h = x in Ring.ideal-subset[of O_K P n I], assumption+)
apply (frule-tac a = x in Ring.a-in-principal[of O_K P n], assumption+)
apply (simp add:subsetD)
apply (rule Ring.ideal-cont-Rxa[of O_K P n I mIg_K n P I], assumption+)
apply (rule mI-gen-in-I1[of n P I], assumption+)
done

```

2.8.2 prime-n-pd

```

lemma (in Corps) prime-n-pd-principal: [| distinct-pds K n P; j ≤ n |] ==>
  (P_K P n j) = Rxa (O_K P n) (((Kb_K n P) j))
apply (frule ring-n-pd[of n P])
apply (frule prime-n-pd-prime[of n P j], assumption+)
apply (simp add:prime-ideal-def, frule conjunct1)
apply (fold prime-ideal-def)
apply (thin-tac prime-ideal (O_K P n) (P_K P n j))
apply (rule equalityI)
apply (rule subsetI)
apply (frule-tac y = x in n-eq-val-eq-idealTr[of n P (Kb_K n P) j])
apply (thin-tac Ring (O_K P n), thin-tac ideal (O_K P n) (P_K P n j))
apply (simp add:ring-n-pd-def Sr-def)
apply (frule Kbase-hom[of n P], simp)
apply (rule allI, rule impI)
apply (frule Kbase-Kronecker[of n P])
apply (simp add:Kronecker-delta-def, rule impI)
apply (simp only:ant-0[THEN sym], simp only:ant-1[THEN sym])
apply (simp del:ant-1)
apply (simp add:prime-n-pd-def)

apply (rule allI, rule impI)
apply (frule Kbase-Kronecker[of n P])
apply simp
apply (thin-tac  $\forall j \leq n. \forall l \leq n. (\nu_K(P_j)) ((Kb_K n P) l) = \delta_{j,l}$ )
apply (case-tac ja = j, simp add:Kronecker-delta-def)
apply (thin-tac ideal (O_K P n) (P_K P n j))
apply (simp add:prime-n-pd-def, erule conjE)
apply (frule-tac x = x in mem-ring-n-pd-mem-K[of n P],
       assumption+)
apply (case-tac x = 0_K)
apply (frule distinct-pds-valuation2[of j n P], assumption+)
apply (rule gt-a0-ge-1, assumption)+

apply (simp add:Kronecker-delta-def)
apply (frule-tac j = ja in distinct-pds-valuation2[of - n P],

```

```

assumption+)
apply (simp add:prime-n-pd-def, erule conjE)
apply (thin-tac ideal ( $O_{K P n}$ ) { $x$ .  $x \in \text{carrier} (O_{K P n}) \wedge 0 < (\nu_K (P j)) x$ })
apply (simp add:ring-n-pd-def Sr-def)
apply (cut-tac  $h = x$  in Ring.ideal-subset[of  $O_{K P n} P_{K P n} j$ ])
apply (frule-tac  $a = x$  in Ring.a-in-principal[of  $O_{K P n}$ ])
apply (simp add:Ring.ideal-subset, assumption+)

apply (rule-tac  $c = x$  and  $A = (O_{K P n}) \diamondsuit_p x$  and  $B = (O_{K P n}) \diamondsuit_p (Kb_{K n P})$ 
j
    in subsetD, assumption+)
apply (simp add:Ring.a-in-principal)
apply (rule Ring.ideal-cont-Rxa[of  $O_{K P n} P_{K P n} j (Kb_{K n P}) j$ , assumption+])
apply (subst prime-n-pd-def, simp)
apply (frule Kbase-Kronecker[of  $n P$ ])
apply (simp add:Kronecker-delta-def)
apply (simp only:ant-1[THEN sym], simp only:ant-0[THEN sym])
apply (simp del:ant-1 add:aless-zless)
apply (subst ring-n-pd-def, simp add:Sr-def)
apply (frule Kbase-hom[of  $n P$ ])
apply simp
apply (rule allI)
apply (simp add:ant-0)
apply (rule impI)
apply (simp only:ant-1[THEN sym], simp only:ant-0[THEN sym])
apply (simp del:ant-1)
done

lemma (in Corps) ring-n-prod-primesTr:[ $0 < n$ ; distinct-pds  $K n P$ ;
ideal ( $O_{K P n}$ )  $I$ ;  $I \neq \{\mathbf{0}_{O_{K P n}}\}$ ;  $I \neq \text{carrier} (O_{K P n})$ ]  $\implies$ 
 $\forall j \leq n. (\nu_K (P j)) (mprod-exp K (mL K P I) (Kb_{K n P}) n) =$ 
 $(\nu_K (P j)) (mIg_{K n P I})$ 
apply (rule allI, rule impI)
apply (simp add:mgenerator1)
apply (simp add:value-mI-gen1)

apply (simp add:value-Zl-mI)
done

lemma (in Corps) ring-n-prod-primesTr1:[ $0 < n$ ; distinct-pds  $K n P$ ;
ideal ( $O_{K P n}$ )  $I$ ;  $I \neq \{\mathbf{0}_{O_{K P n}}\}$ ;  $I \neq \text{carrier} (O_{K P n})$ ]  $\implies$ 
 $I = (O_{K P n}) \diamondsuit_p (mprod-exp K (mL K P I) (Kb_{K n P}) n)$ 
apply (frule ring-n-pd[of  $n P$ ])
apply (subst n-eq-val-eq-ideal[of  $n P$  mprod-exp K (mL K P I)
 $(Kb_{K n P}) n mIg_{K n P I}$ , assumption+])
apply (simp add:mgeneratorTr4)
apply (frule mI-gen-in-I1[of  $n P I$ ], assumption+)

```

```

apply (simp add:Ring.ideal-subset)
apply (simp add:ring-n-prod-primesTr)
apply (simp add:mI-principal)
done

lemma (in Corps) ring-n-prod-primes: [| 0 < n; distinct-pds K n P;
ideal (O_K_P_n) I; I ≠ {0_O_K_P_n}; I ≠ carrier (O_K_P_n);
∀ k ≤ n. J k = (P_K_P_n k) ⊗ (O_K_P_n) (nat ((mL K P I) k)) |] ==>
I = iΠ(O_K_P_n),n J
apply (simp add:prime-n-pd-principal[of n P])
apply (subst ring-n-prod-primesTr1[of n P I], assumption+)
apply (frule ring-n-pd[of n P])
apply (frule Ring.prod-n-principal-ideal[of O_K_P_n nat o (mL K P I) n
Kb_K_n_P J])
apply (frule Kbase-hom[of n P])
apply (simp add:nat-def)
apply (subst ring-n-pd-def) apply (simp add:Sr-def)
apply (rule Pi-I, simp)
apply (simp add:Kbase-Kronecker[of n P])
apply (simp add:Kronecker-delta-def)
apply (simp only:ant-1[THEN sym], simp only:ant-0[THEN sym])
apply (simp del:ant-1)
apply (simp add:Kbase-hom) apply simp

apply simp
apply (frule ring-n-mprod-mprodR[of n P n mL K P I Kb_K_n_P])
apply (rule allI, rule impI, simp add:Zset-def)
apply (rule allI, rule impI)
apply (simp add: Zleast-in-mI-pos)

apply (rule allI, rule impI)
apply (subst ring-n-pd-def) apply (simp add:Sr-def)
apply (frule Kbase-hom1[of n P], simp)
apply (simp add:zero-in-ring-n-pd-zero-K)
apply (frule Kbase-Kronecker[of n P])
apply (simp add:Kronecker-delta-def)
apply (simp only:ant-1[THEN sym], simp only:ant-0[THEN sym])
apply (simp del:ant-1)
apply simp
done

end

```

```

theory Valuation3
imports Valuation2
begin

```

2.9 Completion

In this section we formalize "completion" of the ground field K

definition

```
limit :: [-, 'b ⇒ ant, nat ⇒ 'b, 'b]
      ⇒ bool ((4lim _ _ -) [90,90,90,91]90) where
limK v f b ←→ (forall N. ∃ M. (forall n. M < n →
      ((f n) ±K (-aK b)) ∈ (vp K v) (Vr K v) (an N)))
```

lemma *not-in-singleton-noneq*: $x \notin \{a\} \implies x \neq a$

apply *simp*

done

lemma *noneq-not-in-singleton*: $x \neq a \implies x \notin \{a\}$

apply *simp*

done

lemma *inf-neq-1*[*simp*]: $\infty \neq 1$

by (*simp only*:*ant-1*[*THEN sym*], *rule* *z-neq-inf*[*THEN not-sym*, *of 1*])

lemma *a1-neq-0*[*simp*]: $(1::ant) \neq 0$

by (*simp only*:*an-1*[*THEN sym*], *simp only*:*an-0*[*THEN sym*],
subst *aneq-natneq*[*of 1 0*], *simp*)

lemma *a1-poss*[*simp*]: $(0::ant) < 1$

by (*cut-tac* *zposs-aposss*[*of 1*], *simp*)

lemma *a-p1-gt*[*simp*]: $\llbracket a \neq \infty; a \neq -\infty \rrbracket \implies a < a + 1$

apply (*cut-tac* *aadd-poss-less*[*of a 1*],
simp add:*aadd-commute*, *assumption+*)

apply *simp*

done

lemma (**in Corps**) *vpr-pow-inter-zero*:*valuation K v* \implies

$(\bigcap \{I. \exists n. I = (vp K v) (Vr K v) (an n)\}) = \{\mathbf{0}\}$

apply (*frule* *Vr-ring*[*of v*], *frule* *vp-ideal*[*of v*])

apply (*rule* *equalityI*)

defer

apply (*rule* *subsetI*)

apply *simp*

apply (*rule* *allI*, *rule* *impI*, *erule* *exE*, *simp*)

apply (*cut-tac* *n = an n* **in** *vp-apow-ideal*[*of v*], *assumption+*)

apply *simp*

apply (*cut-tac* *I = (vp K v) (Vr K v) (an n)* **in** *Ring.ideal-zero*[*of Vr K v*],
assumption+)

```

apply (simp add:Vr-0-f-0)
apply (rule subsetI, simp)
apply (rule contrapos-pp, simp+)
apply (subgoal-tac  $x \in vp K v$ )
prefer 2
apply (drule-tac  $x = vp K v (Vr K v) (an 1)$  in spec)
apply (subgoal-tac  $\exists n. vp K v (Vr K v) (an (Suc 0)) = vp K v (Vr K v) (an n)$ ,
         simp,
         thin-tac  $\exists n. vp K v (Vr K v) (an (Suc 0)) = vp K v (Vr K v) (an n)$ )
apply (simp add:r-apow-def an-def)
apply (simp only:na-1)
apply (simp only:Ring.idealpow-1-self[of Vr K v vp K v])

apply blast

apply (frule n-val-valuation[of v])

apply (frule-tac  $x = x$  in val-nonzero-z[of n-val K v],
         frule-tac  $h = x$  in Ring.ideal-subset[of Vr K v vp K v],
         assumption+,
         simp add:Vr-mem-f-mem, assumption+) apply (
             frule-tac  $h = x$  in Ring.ideal-subset[of Vr K v vp K v],
             simp add:Vr-mem-f-mem, assumption+)
apply (cut-tac  $x = x$  in val-pos-mem-Vr[of v], assumption) apply (
         simp add:Vr-mem-f-mem, simp)
apply (frule-tac  $x = x$  in val-pos-n-val-pos[of v],
         simp add:Vr-mem-f-mem, simp)
apply (cut-tac  $x = n-val K v x$  and  $y = 1$  in aadd-pos-poss, assumption+,
         simp) apply (frule-tac  $y = n-val K v x + 1$  in aless-imp-le[of 0])
apply (cut-tac  $x1 = x$  and  $n1 = (n-val K v x) + 1$  in n-val-n-pow[THEN sym,
         of v], assumption+)
apply (drule-tac  $a = vp K v (Vr K v) (n-val K v x + 1)$  in forall-spec)
apply (erule exE, simp)
apply (simp only:ant-1[THEN sym]  $a\text{-}zpz$ ,
         cut-tac  $z = z + 1$  in z-neq-inf)
apply (subst an-na[THEN sym], assumption+, blast)
apply simp
apply (cut-tac  $a = n-val K v x$  in a-p1-gt)
apply (erule exE, simp only:ant-1[THEN sym], simp only:a-zpz z-neq-inf)
apply (cut-tac  $i = z + 1$  and  $j = z$  in ale-zle, simp)
apply (erule exE, simp add:z-neq-minf)
apply (cut-tac  $y1 = n-val K v x$  and  $x1 = n-val K v x + 1$  in
         aneg-le[THEN sym], simp)
done

lemma (in Corps) limit-diff-n-val: $\llbracket b \in carrier K; \forall j. f j \in carrier K;$ 
 $\forall N. \exists M. \forall n. M < n \longrightarrow$ 
 $(an N) \leq (n-val K v ((f n) \pm (-_a b)))$ 

```

```

apply (rule iffI)
apply (rule allI)
apply (simp add:limit-def) apply (rotate-tac -1)
apply (drule-tac x = N in spec)
apply (erule exE)
apply (subgoal-tac  $\forall n > M. (an N) \leq (n\text{-val } K v (f n \pm (-_a b)))$ )
apply blast
apply (rule allI, rule impI) apply (rotate-tac -2)
apply (drule-tac x = n in spec, simp)
apply (rule n-value-x-1[of v], assumption+,
      simp add:an-nat-pos, assumption)

apply (simp add:limit-def)
apply (rule allI, rotate-tac -1, drule-tac x = N in spec)

apply (erule exE)
apply (subgoal-tac  $\forall n > M. f n \pm (-_a b) \in vp K v (Vr K v) (an N)$ )
apply blast
apply (rule allI, rule impI)
apply (rotate-tac -2, drule-tac x = n in spec, simp)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (rule-tac x = f n  $\pm -_a b$  and n = an N in n-value-x-2[of v],
      assumption+)
apply (subst val-pos-mem-Vr[THEN sym, of v], assumption+)
apply (drule-tac x = n in spec)
apply (rule aGroup.ag-pOp-closed[of K], assumption+)
apply (simp add:aGroup.ag-mOp-closed)
apply (subst val-pos-n-val-pos[of v], assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+) apply simp
apply (simp add:aGroup.ag-mOp-closed)
apply (rule-tac j = an N and k = n-val K v (f n  $\pm -_a b$ ) in
      ale-trans[of 0], simp, assumption+)
apply simp
done

lemma (in Corps) an-na-Lv:valuation K v  $\implies$  an (na (Lv K v)) = Lv K v
apply (frule Lv-pos[of v])
apply (frule aless-imp-le[of 0 Lv K v])
apply (frule apos-neq-minf[of Lv K v])
apply (frule Lv-z[of v], erule exE)
apply (rule an-na)
apply (rule contrapos-pp, simp+)
done

lemma (in Corps) limit-diff-val: $\llbracket b \in carrier K; \forall j. f j \in carrier K;$ 
 $\text{valuation } K v \rrbracket \implies (\lim_K v f b) = (\forall N. \exists M. \forall n. M < n \longrightarrow$ 
 $(an N) \leq (v ((f n) \pm (-_a b))))$ 
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
      simp add:limit-diff-n-val[of b f v])

```

```

apply (rule iffI)
apply (rule allI,
       rotate-tac -1, drule-tac x = N in spec, erule exE)
apply (subgoal-tac  $\forall n > M. \text{an } N \leq v(f n \pm -_a b)$ , blast)
apply (rule allI, rule impI)
apply (drule-tac x = n in spec,
       drule-tac x = n in spec, simp)
apply (frule aGroup.ag-mOp-closed[of K b], assumption+,
       frule-tac x = f n and y =  $-_a b$  in aGroup.ag-pOp-closed, assumption+)
apply (frule-tac x = f n  $\pm -_a b$  in n-val-le-val[of v],
       assumption+)
apply (cut-tac n = N in an-nat-pos)
apply (frule-tac j = an N and k = n-val K v (f n  $\pm -_a b$ ) in
       ale-trans[of 0], assumption+)
apply (subst val-pos-n-val-pos, assumption+)
apply (rule-tac i = an N and j = n-val K v (f n  $\pm -_a b$ ) and
       k = v (f n  $\pm -_a b$ ) in ale-trans, assumption+)
apply (rule allI)
apply (rotate-tac 3, drule-tac x = N * (na (Lv K v)) in spec)
apply (erule exE)
apply (subgoal-tac  $\forall n. M < n \rightarrow (\text{an } N) \leq n\text{-val } K v (f n \pm -_a b)$ , blast)
apply (rule allI, rule impI)
apply (rotate-tac -2, drule-tac x = n in spec, simp)

apply (drule-tac x = n in spec)
apply (frule aGroup.ag-mOp-closed[of K b], assumption+,
       frule-tac x = f n and y =  $-_a b$  in aGroup.ag-pOp-closed, assumption+)
apply (cut-tac n = N * na (Lv K v) in an-nat-pos)
apply (frule-tac i = 0 and j = an (N * na (Lv K v)) and
       k = v (f n  $\pm -_a b$ ) in ale-trans, assumption+)
apply (simp add:amult-an-an, simp add:an-na-Lv)
apply (frule Lv-pos[of v])
apply (frule-tac x1 = f n  $\pm -_a b$  in n-val[THEN sym, of v],
       assumption+, simp)
apply (frule Lv-z[of v], erule exE, simp)
apply (simp add:amult-pos-mono-r)
done

```

uniqueness of the limit is derived from *vp-pow-inter-zero*

```

lemma (in Corps) limit-unique:[ $b \in \text{carrier } K; \forall j. f j \in \text{carrier } K;$ 
  valuation K v;  $c \in \text{carrier } K; \lim_K v f b; \lim_K v f c \Rightarrow b = c$ ]
apply (rule contrapos-pp, simp+, simp add:limit-def,
       cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       simp add:aGroup.ag-neq-diffnonzero[of K b c],
       frule vpr-pow-inter-zero[THEN sym, of v],
       frule noneq-not-in-singleton[of b  $\pm -_a c$  0])
apply simp
apply (rotate-tac -1, erule exE, erule conjE)
apply (erule exE, simp, thin-tac x = (vp K v)(Vr K v) (an n))

```

```

apply (rotate-tac 3, drule-tac  $x = n$  in spec,
      erule exE,
      drule-tac  $x = n$  in spec,
      erule exE)
apply (rename-tac  $x N M1 M2$ )
apply (subgoal-tac  $M1 < Suc(\max M1 M2)$ ,
      subgoal-tac  $M2 < Suc(\max M1 M2)$ ,
      drule-tac  $x = Suc(\max M1 M2)$  in spec,
      drule-tac  $x = Suc(\max M1 M2)$  in spec,
      drule-tac  $x = Suc(\max M1 M2)$  in spec)
apply simp

apply (frule-tac  $n = an N$  in vp-apow-ideal[of  $v$ ],
      frule-tac  $x = f(Suc(\max M1 M2)) \pm -_a b$  and  $N = an N$  in
      mem-vp-apow-mem-Vr[of  $v$ ], simp,
      frule Vr-ring[of  $v$ ], simp, simp)

apply (frule Vr-ring[of  $v$ ],
      frule-tac  $I = vp K v(Vr K v)(an N)$  and  $x = f(Suc(\max M1 M2)) \pm -_a b$ 
      in Ring.ideal-inv1-closed[of  $Vr K v$ ], assumption+)

apply (frule-tac  $I = vp K v(Vr K v)(an N)$  and  $h = f(Suc(\max M1 M2)) \pm -_a b$ 
      in Ring.ideal-subset[of  $Vr K v$ ], assumption+)
apply (simp add:Vr-mOp-f-mOp,
      cut-tac field-is-ring, frule Ring.ring-is-ag[of  $K$ ],
      frule aGroup.ag-mOp-closed[of  $K b$ ], assumption+,
      simp add:aGroup.ag-p-inv, simp add:aGroup.ag-inv-inv)

apply (frule-tac  $I = vp K v(Vr K v)(an N)$  and  $x =$ 
       $-_a(f(Suc(\max M1 M2))) \pm b$  and  $y = f(Suc(\max M1 M2)) \pm (-_a c)$  in
      Ring.ideal-pOp-closed[of  $Vr K v$ ], assumption+)
apply (frule-tac  $x = f(Suc(\max M1 M2)) \pm -_a c$  and  $N = an N$  in
      mem-vp-apow-mem-Vr[of  $v$ ], simp, assumption,
      frule-tac  $x = -_a(f(Suc(\max M1 M2))) \pm b$  and  $N = an N$  in
      mem-vp-apow-mem-Vr[of  $v$ ], simp,
      simp add:Vr-pOp-f-pOp, simp add:Vr-pOp-f-pOp,
      frule aGroup.ag-mOp-closed[of  $K c$ ], assumption+,
      frule-tac  $x = f(Suc(\max M1 M2))$  in aGroup.ag-mOp-closed[of  $K$ ],
      assumption+,
      frule-tac  $x = f(Suc(\max M1 M2))$  and  $y = -_a c$  in
      aGroup.ag-pOp-closed[of  $K$ ], assumption+)

apply (simp add:aGroup.ag-pOp-assoc[of  $K - b -$ ],
      simp add:aGroup.ag-pOp-assoc[THEN sym, of  $K b - -_a c$ ],
      simp add:aGroup.ag-pOp-commute[of  $K b$ ],
      simp add:aGroup.ag-pOp-assoc[of  $K - b -_a c$ ],
      frule aGroup.ag-pOp-closed[of  $K b -_a c$ ], assumption+,
      aGroup.ag-pOp-closed[of  $K b -_a c$ ], assumption+)

```

```

simp add:aGroup.ag-pOp-assoc[THEN sym, of K - - b ± -a c],
simp add:aGroup.ag-l-inv1, simp add:aGroup.ag-l-zero)
apply (simp add:aGroup.ag-pOp-commute[of K - b])
apply arith apply arith
done

```

lemma (in Corps) limit-n-val: $\llbracket b \in \text{carrier } K; b \neq \mathbf{0}; \text{valuation } K v;$

```

   $\forall j. f j \in \text{carrier } K; \lim_K v f b \rrbracket \implies$ 
   $\exists N. (\forall m. N < m \longrightarrow (\text{n-val } K v) (f m) = (\text{n-val } K v) b)$ 
apply (simp add:limit-def)
apply (frule n-val-valuation[of v])
apply (frule val-nonzero-z[of n-val K v b], assumption+, erule exE,
       rename-tac L)
apply (rotate-tac -3, drule-tac x = nat (abs L + 1) in spec)
apply (erule exE, rename-tac M)

```

apply (subgoal-tac $\forall n. M < n \longrightarrow \text{n-val } K v (f n) = \text{n-val } K v b$, blast)

```

apply (rule allI, rule impI)
apply (rotate-tac -2,
       drule-tac x = n in spec,
       simp)
apply (frule-tac x = f n ± -a b and n = an (nat (|L| + 1)) in
       n-value-x-1[of v],
       thin-tac f n ± -a b ∈ vp K v (Vr K v) (an (nat (|L| + 1))))
apply (simp add:an-def)
apply (simp add: zpos-apos [symmetric])
apply assumption

```

apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])

apply (frule aGroup.ag-mOp-closed[of K b], assumption+)

```

apply (drule-tac x = n in spec,
       frule-tac x = f n in aGroup.ag-pOp-closed[of K - -a b],
       assumption+,
       frule-tac x = b and y = (f n) ± (-a b) in value-less-eq[of
       n-val K v], assumption+)
apply simp

```

apply (rule-tac x = ant L and y = an (nat (|L| + 1)) and
 z = n-val K v (f n ± -_a b) in aless-le-trans)

```

apply (subst an-def)
apply (subst aless-zless) apply arith apply assumption+
apply (simp add:aGroup.ag-pOp-commute[of K b])

```

apply (simp add:aGroup.ag-pOp-assoc) apply (simp add:aGroup.ag-l-inv1)
apply (simp add:aGroup.ag-r-zero)
done

```

lemma (in Corps) limit-val: $\llbracket b \in carrier K; b \neq \mathbf{0}; \forall j. f j \in carrier K;$ 
 $\quad valuation K v; lim_K v f b \rrbracket \implies \exists N. (\forall n. N < n \longrightarrow v(f n) = v b)$ 
apply (frule limit-n-val[of b v f], assumption+)
apply (erule exE)
apply (subgoal-tac  $\forall m. N < m \longrightarrow v(f m) = v b$ )
apply blast
apply (rule allI, rule impI)
apply (drule-tac  $x = m$  in spec)
apply (drule-tac  $x = m$  in spec)
apply simp
apply (frule Lv-pos[of v])
apply (simp add:n-val[THEN sym, of v])
done

lemma (in Corps) limit-val-infinity: $\llbracket valuation K v; \forall j. f j \in carrier K;$ 
 $\quad lim_K v f \mathbf{0} \rrbracket \implies \forall N. (\exists M. (\forall m. M < m \longrightarrow (an N) \leq (n-val K v (f m))))$ 
apply (simp add:limit-def)
apply (rule allI)
apply (drule-tac  $x = N$  in spec,
      erule exE)

apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
      simp only:aGroup.ag-inv-zero[of K], simp only:aGroup.ag-r-zero)
apply (subgoal-tac  $\forall n. M < n \longrightarrow an N \leq n-val K v (f n)$ , blast)

apply (rule allI, rule impI)
apply (drule-tac  $x = n$  in spec,
      drule-tac  $x = n$  in spec, simp)
apply (simp add:n-value-x-1)
done

lemma (in Corps) not-limit-zero: $\llbracket valuation K v; \forall j. f j \in carrier K;$ 
 $\neg (lim_K v f \mathbf{0}) \rrbracket \implies \exists N. (\forall M. (\exists m. (M < m) \wedge$ 
 $\quad ((n-val K v) (f m)) < (an N)))$ 
apply (simp add:limit-def)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (simp add:aGroup.ag-inv-zero aGroup.ag-r-zero)
apply (erule exE)
apply (case-tac  $N = 0$ , simp add:r-apow-def)
apply (subgoal-tac  $\forall M. \exists n. M < n \wedge n-val K v (f n) < (an 0)$ ) apply blast
apply (rule allI,
      rotate-tac 4, frule-tac  $x = M$  in spec,
      erule exE, erule conjE)
apply (frule-tac  $x1 = f n$  in val-pos-mem-Vr[THEN sym, of v]) apply simp
apply simp
apply (frule-tac  $x = f n$  in val-pos-n-val-pos[of v])
apply simp apply simp apply (thin-tac  $\neg 0 \leq v(f n)$ )
apply (simp add:aneg-le) apply blast

```

```

apply (simp)
apply (subgoal-tac  $\forall n. ((f n) \in vp K v (Vr K v) (an N)) =$ 
           $((an N) \leq n\text{-val } K v (f n)))$ )
apply simp
apply (thin-tac  $\forall n. (f n \in vp K v (Vr K v) (an N)) = (an N \leq n\text{-val } K v (f n)))$ )
apply (simp add:aneg-le) apply blast

apply (rule allI)
apply (thin-tac  $\forall M. \exists n. M < n \wedge f n \notin vp K v (Vr K v) (an N)$ )
apply (rule iffI)
apply (frule-tac  $x1 = f n \text{ and } n1 = an N \text{ in } n\text{-val-}n\text{-pow}[THEN sym, of } v]$ )
apply (rule-tac  $N = an N \text{ and } x = f n \text{ in } mem\text{-}vp\text{-}apow\text{-}mem\text{-}Vr[of } v]$ ,
       assumption+, simp, assumption, simp, simp)
apply (frule-tac  $x = f n \text{ and } n = an N \text{ in } n\text{-val-}n\text{-pow}[of } v]$ )
apply (frule-tac  $x1 = f n \text{ in } val\text{-}pos\text{-}mem\text{-}Vr[THEN sym, of } v]$ )
apply simp
apply (cut-tac  $n = N \text{ in } an\text{-nat}\text{-}pos$ )
apply (frule-tac  $j = an N \text{ and } k = n\text{-val } K v (f n) \text{ in ale-trans}[of } 0],$ 
           assumption+)
apply (frule-tac  $x1 = f n \text{ in } val\text{-}pos\text{-}n\text{-val}\text{-}pos[THEN sym, of } v)$ )
apply simp+
done

lemma (in Corps) limit-p:[valuation K v;  $\forall j. f j \in carrier K;$ 
 $\forall j. g j \in carrier K; b \in carrier K; c \in carrier K; lim_{K v} f b; lim_{K v} g c]$ 
 $\implies lim_{K v} (\lambda j. (f j) \pm (g j)) (b \pm c)$ 
apply (simp add:limit-def)
apply (rule allI) apply (rotate-tac 3,
drule-tac  $x = N \text{ in spec,}$ 
drule-tac  $x = N \text{ in spec,}$ 
(erule exE)+)
apply (frule-tac  $x = M \text{ and } y = Ma \text{ in two-inequalities, simp,}$ 
subgoal-tac  $\forall n > (max M Ma). (f n) \pm (g n) \pm -_a (b \pm c)$ 
 $\in (vp K v) (Vr K v) (an N)$ )
apply (thin-tac  $\forall n. Ma < n \longrightarrow$ 
 $g n \pm -_a c \in (vp K v) (Vr K v) (an N),$ 
thin-tac  $\forall n. M < n \longrightarrow$ 
 $f n \pm -_a b \in (vp K v) (Vr K v) (an N), blast$ )
apply (frule Vr-ring[of v],
frule-tac  $n = an N \text{ in } vp\text{-}apow\text{-}ideal[of } v]$ )
apply simp
apply (rule allI, rule impI)
apply (thin-tac  $\forall n > M. f n \pm -_a b \in vp K v (Vr K v) (an N),$ 
thin-tac  $\forall n > Ma. g n \pm -_a c \in vp K v (Vr K v) (an N),$ 
frule-tac  $I = vp K v (Vr K v) (an N) \text{ and } x = f n \pm -_a b$ 
 $\text{and } y = g n \pm -_a c \text{ in Ring.ideal-pOp-closed[of } Vr K v],$ 
assumption+, simp, simp)
apply (frule-tac  $I = vp K v (Vr K v) (an N) \text{ and } h = f n \pm -_a b$ 

```

```

in Ring.ideal-subset[of Vr K v], assumption+, simp,
frule-tac I = vp K v(Vr K v) (an N) and h = g n ± -a c in
Ring.ideal-subset[of Vr K v], assumption+, simp)
apply (simp add:Vr-pOp-f-pOp)
apply (thin-tac f n ± -a b ∈ carrier (Vr K v),
thin-tac g n ± -a c ∈ carrier (Vr K v))

apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule aGroup.ag-mOp-closed[of K b], assumption+,
frule aGroup.ag-mOp-closed[of K c], assumption+,
frule-tac a = f n and b = -a b and c = g n and d = -a c in
aGroup.ag-add4-rel[of K], simp+)
apply (simp add:aGroup.ag-p-inv[THEN sym])
done

lemma (in Corps) Abs-ant-abs[simp]:Abs (ant z) = ant (abs z)
apply (simp add:Abs-def, simp add:aminus)
apply (simp only:ant-0[THEN sym], simp only:aless-zless)
apply (simp add:zabs-def)
done

lemma (in Corps) limit-t-nonzero:[valuation K v; ∀(j::nat). (f j) ∈ carrier K;
∀(j::nat). g j ∈ carrier K; b ∈ carrier K; c ∈ carrier K; b ≠ 0; c ≠ 0;
limK v f b; limK v g c] ⇒ limK v (λj. (f j) ·r (g j)) (b ·r c)
apply (cut-tac field-is-ring, simp add:limit-def, rule allI)
apply (subgoal-tac ∀j. (f j) ·r (g j) ± -a (b ·r c) =
((f j) ·r (g j)) ± (f j) ·r (-a c)) ± ((f j) ·r c ± -a (b ·r c)), simp,
thin-tac ∀j. f j ·r g j ± -a b ·r c =
f j ·r g j ± f j ·r (-a c) ± (f j ·r c ± -a b ·r c))
apply (frule limit-n-val[of b v f], assumption+,
simp add:limit-def)
apply (frule n-val-valuation[of v])
apply (frule val-nonzero-z[of n-val K v b], assumption+,
rotate-tac -1, erule exE,
frule val-nonzero-z[of n-val K v c], assumption+,
rotate-tac -1, erule exE, rename-tac N bz cz)
apply (rotate-tac 5,
drule-tac x = N + nat (abs cz) in spec,
drule-tac x = N + nat (abs bz) in spec)
apply (erule exE)+
apply (rename-tac N bz cz M M1 M2)

apply (subgoal-tac ∀n. (max (max M1 M2) M) < n →
(f n) ·r (g n) ± (f n) ·r (-a c) ± ((f n) ·r c ± (-a (b ·r c)))
∈ vp K v (Vr K v) (an N), blast)
apply (rule allI, rule impI) apply (simp, (erule conjE)+)
apply (rotate-tac 11, drule-tac x = n in spec,
drule-tac x = n in spec, simp,
drule-tac x = n in spec, simp)

```

```

apply (frule-tac  $b = g n \pm -_a c$  and  $n = an N$  and  $x = f n$  in
      convergenceTr1[of  $v$ ])
apply simp apply simp apply (simp add:an-def a-zpz[THEN sym]) apply simp
apply (frule-tac  $b = f n \pm -_a b$  and  $n = an N$  in convergenceTr1[of
 $v c$ ], assumption+, simp) apply (simp add:an-def)
apply (simp add:a-zpz[THEN sym]) apply simp

apply (drule-tac  $x = n$  in spec,
        drule-tac  $x = n$  in spec)
apply (simp add:Ring.ring-inv1-1[of  $K b c$ ],
        cut-tac Ring.ring-is-ag, frule aGroup.ag-mOp-closed[of  $K c$ ],
        assumption+,
        simp add:Ring.ring-distrib1[THEN sym],
        frule aGroup.ag-mOp-closed[of  $K b$ ], assumption+,
        simp add:Ring.ring-distrib2[THEN sym],
        subst Ring.ring-tOp-commute[of  $K - c$ ], assumption+,
        rule aGroup.ag-pOp-closed, assumption+)
apply (cut-tac  $n = N$  in an-nat-pos)
apply (frule-tac  $n = an N$  in vp-apow-ideal[of  $v$ ], assumption+)
apply (frule Vr-ring[of  $v$ ])

apply (frule-tac  $x = (f n) \cdot_r (g n \pm -_a c)$  and  $y = c \cdot_r (f n \pm -_a b)$ 
      and  $I = vp K v (Vr K v) (an N)$  in Ring.ideal-pOp-closed[of  $Vr K v$ ],
      assumption+)
apply (frule-tac  $R = Vr K v$  and  $I = vp K v (Vr K v) (an N)$  and
       $h = (f n) \cdot_r (g n \pm -_a c)$  in Ring.ideal-subset, assumption+,
      frule-tac  $R = Vr K v$  and  $I = vp K v (Vr K v) (an N)$  and
       $h = c \cdot_r (f n \pm -_a b)$  in Ring.ideal-subset, assumption+)
apply (simp add:Vr-pOp-f-pOp, assumption+)
apply (rule allI)
apply (thin-tac  $\forall N. \exists M. \forall n > M. f n \pm -_a b \in vp K v (Vr K v) (an N)$ ,
        thin-tac  $\forall N. \exists M. \forall n > M. g n \pm -_a c \in vp K v (Vr K v) (an N)$ )
apply (drule-tac  $x = j$  in spec,
        drule-tac  $x = j$  in spec,
        cut-tac field-is-ring, frule Ring.ring-is-ag[of  $K$ ],
        frule-tac  $x = f j$  and  $y = g j$  in Ring.ring-tOp-closed, assumption+,
        frule-tac  $x = b$  and  $y = c$  in Ring.ring-tOp-closed, assumption+,
        frule-tac  $x = f j$  and  $y = c$  in Ring.ring-tOp-closed, assumption+,
        frule-tac  $x = c$  in aGroup.ag-mOp-closed[of  $K$ ], assumption+,
        frule-tac  $x = f j$  and  $y = -_a c$  in Ring.ring-tOp-closed, assumption+,
        frule-tac  $x = b \cdot_r c$  in aGroup.ag-mOp-closed[of  $K$ ], assumption+)
apply (subst aGroup.pOp-assocTr41[THEN sym, of  $K$ ], assumption+,
        subst aGroup.pOp-assocTr42[of  $K$ ], assumption+,
        subst Ring.ring-distrib1[THEN sym, of  $K$ ], assumption+)
apply (simp add:aGroup.ag-l-inv1, simp add:Ring.ring-times-x-0,
        simp add:aGroup.ag-r-zero)
done

```

```

lemma an-npn[simp]:an (n + m) = an n + an m
by (simp add:an-def a-zpz)

lemma Abs-noninf:a ≠ -∞ ∧ a ≠ ∞ ==> Abs a ≠ ∞
by (cut-tac mem-ant[of a], simp, erule exE, simp add:Abs-def,
      simp add:aminus)

lemma (in Corps) limit-t-zero:[c ∈ carrier K; valuation K v;
  ∀(j::nat). f j ∈ carrier K; ∀(j::nat). g j ∈ carrier K;
  lim_K v f 0; lim_K v g c] ==> lim_K v (λj. (f j) ·_r (g j)) 0
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
      simp add:limit-def, simp add:aGroup.ag-inv-zero, simp add:aGroup.ag-r-zero)
apply (subgoal-tac ∀j. (f j) ·_r (g j) ∈ carrier K,
      simp add:aGroup.ag-r-zero)
prefer 2 apply (rule allI, simp add:Ring.ring-tOp-closed)
apply (case-tac c = 0_K)
apply (simp add:aGroup.ag-inv-zero, simp add:aGroup.ag-r-zero)
apply (rule allI,
      rotate-tac 4,
      drule-tac x = N in spec,
      drule-tac x = N in spec, (erule exE)+,
      rename-tac N M1 M2)
apply (subgoal-tac ∀n>(max M1 M2). (f n) ·_r (g n) ∈ (vp K v)^(Vr K v) (an N),
      blast)
apply (rule allI, rule impI)
apply (drule-tac x = M1 and y = M2 in two-inequalities, assumption+,
      thin-tac ∀n>M2. g n ∈ vp K v (Vr K v) (an N))
apply (thin-tac ∀j. f j ·_r g j ∈ carrier K,
      thin-tac ∀j. f j ∈ carrier K,
      thin-tac ∀j. g j ∈ carrier K,
      drule-tac x = n in spec, simp, erule conjE,
      erule conjE,
      frule Vr-ring[of v])
apply (cut-tac n = N in an-nat-pos)
apply (frule-tac x = f n in mem-vp-apow-mem-Vr[of v], assumption+,
      frule-tac x = g n in mem-vp-apow-mem-Vr[of v], assumption+,
      frule-tac n = an N in vp-apow-ideal[of v], assumption+)
apply (frule-tac I = vp K v (Vr K v) (an N) and x = g n and
      r = f n in Ring.ideal-ring-multiple[of Vr K v], assumption+,
      simp add:Vr-tOp-f-tOp)

apply (rule allI)
apply (subgoal-tac ∀j. (f j) ·_r (g j) =
  (f j) ·_r ((g j) ± (-_a c)) ± (f j) ·_r c, simp,
  thin-tac ∀j. (f j) ·_r (g j) =
  (f j) ·_r ((g j) ± (-_a c)) ± (f j) ·_r c,
  thin-tac ∀j. (f j) ·_r (g j ± -_a c) ± (f j) ·_r c ∈ carrier K)

```

```

apply (rotate-tac 4,
      drule-tac  $x = N + na$  ( $Abs(n-val K v c)$ ) in spec,
      drule-tac  $x = N$  in spec)
apply (erule exE)+ apply (rename-tac  $N M1 M2$ )
apply (subgoal-tac  $\forall n.$  ( $max M1 M2 < n \rightarrow (f n) \cdot_r (g n \pm -_a c) \pm$ 
                            $(f n) \cdot_r c \in vp K v (Vr K v) (an N)$ , blast))
apply (rule allI, rule impI, simp, erule conjE,
      drule-tac  $x = n$  in spec,
      drule-tac  $x = n$  in spec,
      drule-tac  $x = n$  in spec)
apply (frule n-val-valuation[of v])
apply (frule value-in-aug-inf[of n-val K v c], assumption+,
       simp add:aug-inf-def)
apply (frule val-nonzero-noninf[of n-val K v c], assumption+)
apply (cut-tac Abs-noninf[of n-val K v c])
apply (cut-tac Abs-pos[of n-val K v c]) apply (simp add:an-na)

apply (drule-tac  $x = n$  in spec, simp)
apply (frule-tac  $b = f n$  and  $n = an N$  in convergenceTr1[of
               v c], assumption+)
apply simp

apply (frule-tac  $x = f n$  and  $N = an N + Abs(n-val K v c)$  in
      mem-vp-apow-mem-Vr[of v],
      frule-tac  $n = an N$  in vp-apow-ideal[of v])
apply simp
apply (rule-tac  $x = an N$  and  $y = Abs(n-val K v c)$  in aadd-two-pos)
apply simp apply (simp add:Abs-pos) apply assumption

apply (frule-tac  $x = g n \pm (-_a c)$  and  $N = an N$  in mem-vp-apow-mem-Vr[of
               v], simp, assumption+) apply (
      frule-tac  $x = c \cdot_r (f n)$  and  $N = an N$  in mem-vp-apow-mem-Vr[of
               v], simp) apply simp
apply (simp add:Ring.ring-tOp-commute[of K c]) apply (
      frule Vr-ring[of v],
      frule-tac  $I = (vp K v) (Vr K v) (an N)$  and  $x = g n \pm (-_a c)$ 
      and  $r = f n$  in Ring.ideal-ring-multiple[of Vr K v])
apply (simp add:vp-apow-ideal) apply assumption+
apply (frule-tac  $I = vp K v (Vr K v) (an N)$  and
       $x = (f n) \cdot_r (g n \pm -_a c)$  and  $y = (f n) \cdot_r c$  in
      Ring.ideal-pOp-closed[of Vr K v])
apply (simp add:vp-apow-ideal)
apply (simp add:Vr-tOp-f-tOp, assumption)
apply (simp add:Vr-tOp-f-tOp Vr-pOp-f-pOp,
      frule-tac  $x = (f n) \cdot_r (g n \pm -_a c)$  and  $N = an N$  in mem-vp-apow-mem-Vr[of
               v], simp add:Vr-pOp-f-pOp, assumption+)
apply (frule-tac  $N = an N$  and  $x = (f n) \cdot_r c$  in mem-vp-apow-mem-Vr[of
               v]) apply simp apply assumption
apply (simp add:Vr-pOp-f-pOp) apply simp

```

```

apply (thin-tac  $\forall N. \exists M. \forall n > M. f n \in vp K v (Vr K v) (an N)$ ,
         thin-tac  $\forall N. \exists M. \forall n > M. g n \pm -_a c \in vp K v (Vr K v) (an N)$ ,
         rule allI)
apply (drule-tac  $x = j$  in spec,
         drule-tac  $x = j$  in spec,
         drule-tac  $x = j$  in spec,
         frule aGroup.ag-mOp-closed[of K c], assumption+,
         frule-tac  $x = f j$  in Ring.ring-tOp-closed[of K - c], assumption+,
         frule-tac  $x = f j$  in Ring.ring-tOp-closed[of K - -a c], assumption+)
apply (simp add:Ring.ring-distrib1, simp add:aGroup.ag-pOp-assoc,
         simp add:Ring.ring-distrib1[THEN sym],
         simp add:aGroup.ag-l-inv1, simp add:Ring.ring-times-x-0,
         simp add:aGroup.ag-r-zero)
done

lemma (in Corps) limit-minus: $\llbracket \text{valuation } K v; \forall j. f j \in \text{carrier } K;$ 
 $b \in \text{carrier } K; \lim_K v f b \rrbracket \implies \lim_K v (\lambda j. (-_a (f j))) (-_a b)$ 
apply (simp add:limit-def)
apply (rule allI,
         rotate-tac  $-1$ , frule-tac  $x = N$  in spec,
         thin-tac  $\forall N. \exists M. \forall n. M < n \xrightarrow{} f n \pm -_a b \in (vp K v) (Vr K v) (an N)$ ,
         erule exE,
         subgoal-tac  $\forall n. M < n \longrightarrow (-_a (f n)) \pm (-_a (-_a b)) \in (vp K v) (Vr K v) (an N)$ ,
         blast)
apply (rule allI, rule impI,
         frule-tac  $x = n$  in spec,
         frule-tac  $x = n$  in spec, simp)
apply (frule Vr-ring[of v],
         frule-tac  $n = an N$  in vp-apow-ideal[of v], simp)
apply (frule-tac  $x = f n \pm -_a b$  and  $N = an N$  in
         mem-vp-apow-mem-Vr[of v], simp+,
         frule-tac  $I = vp K v (Vr K v) (an N)$  and  $x = f n \pm (-_a b)$  in
         Ring.ideal-inv1-closed[of Vr K v], assumption+, simp)
apply (simp add:Vr-mOp-f-mOp)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
         frule aGroup.ag-mOp-closed[of K b], assumption+)
apply (simp add:aGroup.ag-p-inv[of K])
done

lemma (in Corps) inv-diff: $\llbracket x \in \text{carrier } K; x \neq \mathbf{0}; y \in \text{carrier } K; y \neq \mathbf{0} \rrbracket \implies$ 
 $(x^K) \pm (-_a (y^K)) = (x^K) \cdot_r (y^K) \cdot_r (-_a (x \pm (-_a y)))$ 
apply (cut-tac invf-closed1[of x], simp, erule conjE,
         cut-tac invf-closed1[of y], simp, erule conjE,
         cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
         frule Ring.ring-tOp-closed[of K x^-K y^-K], assumption+,

```

```

frule aGroup.ag-mOp-closed[of K x], assumption+,
frule aGroup.ag-mOp-closed[of K y], assumption+,
frule aGroup.ag-mOp-closed[of K x-K], assumption+,
frule aGroup.ag-mOp-closed[of K y-K], assumption+,
frule aGroup.ag-pOp-closed[of K x -a y], assumption+)

apply (simp add:Ring.ring-inv1-2[THEN sym],
      simp only:Ring.ring-distrib1[of K (x-K) ·r (y-K) x -a y],
      simp only:Ring.ring-tOp-commute[of K - x],
      simp only:Ring.ring-inv1-2[THEN sym, of K],
      simp only:Ring.ring-tOp-assoc[THEN sym],
      simp only:Ring.ring-tOp-commute[of K x],
      cut-tac linvf[of x], simp+,
      simp add:Ring.ring-l-one, simp only:Ring.ring-tOp-assoc,
      cut-tac linvf[of y], simp+,
      simp only:Ring.ring-r-one)
apply (simp add:aGroup.ag-p-inv[of K], simp add:aGroup.ag-inv-inv,
       rule aGroup.ag-pOp-commute[of K x-K -a y-K], assumption+)
apply simp+
done

lemma times2plus:(2::nat)*n = n + n
by simp

lemma (in Corps) limit-inv:[valuation K v; ∀ j. f j ∈ carrier K;
b ∈ carrier K; b ≠ 0; limK v f b] ⇒
limK v (λj. if (f j) = 0 then 0 else (f j)-K) (b-K)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (frule limit-n-val[of b v f], assumption+)
apply (subst limit-def)
apply (rule allI, erule exE)
apply (subgoal-tac ∀ m>Na. f m ≠ 0)
prefer 2
apply (rule allI, rule impI, rotate-tac -2,
drule-tac x = m in spec, simp)
apply (frule n-val-valuation[of v])
apply (frule val-nonzero-noninf[of n-val K v b], assumption+)
apply (rule contrapos-pp, simp+, simp add:value-of-zero)
apply (unfold limit-def)
apply (rotate-tac 2,
frule-tac x = N + 2*(na (Abs (n-val K v b))) in
spec)
apply (erule exE)
apply (subgoal-tac ∀ n>(max Na M).
(if f n = 0 then 0 else f n-K) ± -a b-K ∈ vp K v(Vr K v) (an N),
blast)
apply (rule allI, rule impI)
apply (cut-tac x = Na and y = max Na M and z = n
in le-less-trans)

```

```

apply simp+
apply (thin-tac  $\forall N. \exists M. \forall n > M. f n \pm -_a b \in vp K v (Vr K v) (an N)$ )
apply (drule-tac  $x = n$  in spec,
      drule-tac  $x = n$  in spec,
      drule-tac  $x = n$  in spec, simp)
apply (subst inv-diff, assumption+)
apply (cut-tac  $x = f n$  in invf-closed1, simp,
      cut-tac  $x = b$  in invf-closed1, simp, simp, (erule conjE)+)

apply (frule-tac  $n = an N + an (2 * na (Abs (n-val K v b)))$  and
        $x = f n \pm -_a b$  in n-value-x-1[of v])
apply (simp only:an-npn[THEN sym], rule an-nat-pos)
apply assumption
apply (rule-tac  $x = f n^- K \cdot_r b^- K \cdot_r (-_a (f n \pm -_a b))$  and  $v = v$  and
        $n = an N$  in n-value-x-2, assumption+)
apply (frule n-val-valuation[of v])
apply (subst val-pos-mem-Vr[THEN sym, of v], assumption+)
apply (rule Ring.ring-tOp-closed, assumption+)+
apply (rule aGroup.ag-mOp-closed, assumption)
apply (rule aGroup.ag-pOp-closed, assumption+, rule aGroup.ag-mOp-closed, assumption+)
apply (subst val-pos-n-val-pos[of v], assumption+, rule Ring.ring-tOp-closed, assumption+, rule Ring.ring-tOp-closed, assumption+, rule aGroup.ag-mOp-closed, assumption+, rule aGroup.ag-pOp-closed, assumption+, rule aGroup.ag-mOp-closed, assumption+)
apply (subst val-t2p[of n-val K v], assumption+, rule Ring.ring-tOp-closed, assumption+, rule aGroup.ag-mOp-closed, assumption+, rule aGroup.ag-pOp-closed, assumption+, rule aGroup.ag-mOp-closed, assumption+, subst val-minus-eq[of n-val K v], assumption+, (rule aGroup.ag-pOp-closed, assumption+), (rule aGroup.ag-mOp-closed, assumption+))
apply (subst val-t2p[of n-val K v], assumption+)
apply (simp add:value-of-inv)
apply (frule-tac  $x = an N + an (2 * na (Abs (n-val K v b)))$  and  $y = n-val K v (f n \pm -_a b)$  and  $z = - n-val K v b + - n-val K v b$  in aadd-le-mono)
apply (cut-tac  $z = n-val K v b$  in Abs-pos)
apply (frule val-nonzero-z[of n-val K v b], assumption+, erule exE)
apply (rotate-tac -1, drule sym, cut-tac  $z = z$  in z-neq-minf,
      cut-tac  $z = z$  in z-neq-inf, simp, cut-tac  $a = (n-val K v b)$  in Abs-noninf, simp)
apply (simp only:times2plus an-npn, simp add:an-na)
apply (rotate-tac -4, drule sym, simp)
apply (thin-tac  $f n \pm -_a b \in vp K v (Vr K v) (an N + (ant |z| + ant |z|))$ )
apply (simp add:an-def, simp add:aminus, (simp add:a-zpz)+)

```

```

apply (subst aadd-commute)
apply (rule-tac i = 0 and j = ant (int N + 2 * |z| - 2 * z) and
         k = n-val K v (f n ± -a b) + ant (-(2 * z)) in ale-trans)
apply (subst ant-0[THEN sym])
apply (subst ale-zle, simp, assumption)

apply (frule n-val-valuation[of v])
apply (subst val-t2p[of n-val K v], assumption+)
apply (rule Ring.ring-tOp-closed, assumption+)
apply (rule aGroup.ag-mOp-closed, assumption)
apply (rule aGroup.ag-pOp-closed, assumption+,
         rule aGroup.ag-mOp-closed, assumption+)
apply (subst val-t2p[of n-val K v], assumption+)
apply (subst val-minus-eq[of n-val K v], assumption+,
         rule aGroup.ag-pOp-closed, assumption+,
         rule aGroup.ag-mOp-closed, assumption+)

apply (simp add:value-of-inv)
apply (frule-tac x = an N + an (2 * na (Abs (n-val K v b))) and y = n-val K
 $v (f n \pm -a b)$  and  $z = - n\text{-val } K v b + - n\text{-val } K v b$  in aadd-le-mono)
apply (cut-tac z = n-val K v b in Abs-pos)
apply (frule val-nonzero-z[of n-val K v b], assumption+, erule exE)
apply (rotate-tac -1, drule sym, cut-tac z = z in z-neq-minf,
         cut-tac z = z in z-neq-inf, simp,
         cut-tac a = (n-val K v b) in Abs-noninf, simp)
apply (simp only:times2plus an-npn, simp add:an-na)
apply (rotate-tac -4, drule sym, simp)
apply (thin-tac f n ± -a b ∈ vp K v (Vr K v) (an N + (ant |z| + ant |z|)))
apply (simp add:an-def, simp add:aminus, (simp add:a-zpz)+)
apply (subst aadd-commute)
apply (rule-tac i = ant (int N) and j = ant (int N + 2 * |z| - 2 * z)
         and  $k = n\text{-val } K v (f n \pm -a b) + ant (-(2 * z))$  in ale-trans)
apply (subst ale-zle, simp, assumption)

apply simp
done

definition
Cauchy-seq ::  $[-, 'b \Rightarrow \text{ant}, \text{nat} \Rightarrow 'b]$ 
 $\Rightarrow \text{bool } ((3\text{Cauchy}_-)_-)_-[90,90,91]90$  where
CauchyK v f  $\longleftrightarrow$   $(\forall n. (f n) \in \text{carrier } K) \wedge$ 
 $\forall N. \exists M. (\forall n m. M < n \wedge M < m \longrightarrow$ 
 $((f n) \pm_K (-_a K (f m))) \in (vp K v)^{(Vr K v) (an N)})$ 

definition
v-complete ::  $['b \Rightarrow \text{ant}, -] \Rightarrow \text{bool}$ 
 $((2\text{Complete}_-)_-) [90,91]90$  where
Completev K f  $\longleftrightarrow$   $(\forall f. (\text{Cauchy}_{K v} f) \longrightarrow$ 
 $(\exists b. b \in (\text{carrier } K) \wedge \lim_{K v} f b))$ 

```

```

lemma (in Corps) has-limit-Cauchy: [[valuation K v;  $\forall j. f j \in \text{carrier } K;$ 
 $b \in \text{carrier } K; \lim_K v f b]] \implies \text{Cauchy}_K v f$ 
apply (simp add:Cauchy-seq-def)
apply (rule allI)
apply (simp add:limit-def)
apply (rotate-tac -1)
apply (drule-tac  $x = N$  in spec)
apply (erule exE)
apply (subgoal-tac  $\forall n m. M < n \wedge M < m \rightarrow$ 
 $f n \pm -_a (f m) \in vp K v(Vr K v) (an N))$ 
apply blast
apply ((rule allI)+, rule impI, erule conjE)
apply (frule-tac  $x = n$  in spec,
frule-tac  $x = m$  in spec,
thin-tac  $\forall j. f j \in \text{carrier } K,$ 
frule-tac  $x = n$  in spec,
frule-tac  $x = m$  in spec,
thin-tac  $\forall n. M < n \rightarrow f n \pm -_a b \in vp K v(Vr K v) (an N),$ 
simp)
apply (frule-tac  $n = an N$  in vp-apow-ideal[of v], simp)
apply (frule Vr-ring[of v])
apply (frule-tac  $x = f m \pm -_a b$  and  $I = vp K v(Vr K v) (an N)$  in
Ring.ideal-inv1-closed[of Vr K v], assumption+)
apply (frule-tac  $h = f m \pm -_a b$  and  $I = vp K v(Vr K v) (an N)$  in
Ring.ideal-subset[of Vr K v], assumption+,
frule-tac  $h = f n \pm -_a b$  and  $I = vp K v(Vr K v) (an N)$  in
Ring.ideal-subset[of Vr K v], assumption+)
apply (frule-tac  $h = -_a Vr K v (f m \pm -_a b)$  and  $I = vp K v(Vr K v) (an N)$  in
Ring.ideal-subset[of Vr K v], assumption+,
frule-tac  $h = f n \pm -_a b$  and  $I = vp K v(Vr K v) (an N)$  in
Ring.ideal-subset[of Vr K v], assumption+)
apply (frule-tac  $I = (vp K v) (Vr K v) (an N)$  and  $x = f n \pm -_a b$  and
 $y = -_a (Vr K v) (f m \pm -_a b)$  in Ring.ideal-pOp-closed[of Vr K v],
assumption+)
apply (simp add:Vr-pOp-f-pOp) apply (simp add:Vr-mOp-f-mOp)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (frule aGroup.ag-mOp-closed[of K b], assumption+)
apply (frule-tac  $x = f m \pm -_a b$  in Vr-mem-f-mem[of v], assumption+)
apply (frule-tac  $x = f m \pm -_a b$  in aGroup.ag-mOp-closed[of K],
assumption+)
apply (simp add:aGroup.ag-pOp-assoc)
apply (simp add:aGroup.ag-pOp-commute[of K -_a b])
apply (simp add:aGroup.ag-p-inv[of K])
apply (frule-tac  $x = f m$  in aGroup.ag-mOp-closed[of K], assumption+)
apply (simp add:aGroup.ag-inv-inv)
apply (simp add:aGroup.ag-pOp-assoc[of K - b -_a b])
apply (simp add:aGroup.ag-r-inv1[of K], simp add:aGroup.ag-r-zero)

```

done

```

lemma (in Corps) no-limit-zero-Cauchy:valuation K v; CauchyK v f;
   $\neg (\lim_{K v} f \mathbf{0}) \Rightarrow$ 
   $\exists N M. (\forall m. N < m \rightarrow ((n\text{-val } K v) (f M)) = ((n\text{-val } K v) (f m)))$ 
apply (frule not-limit-zero[of v f], thin-tac  $\neg \lim_{K v} f \mathbf{0}$ )
apply (simp add:Cauchy-seq-def, assumption) apply (erule exE)
apply (rename-tac L)
apply (simp add:Cauchy-seq-def, erule conjE,
  rotate-tac -1,
  frule-tac x = L in spec, thin-tac  $\forall N. \exists M. \forall n m.$ 
   $M < n \wedge M < m \rightarrow f n \pm -_a (f m) \in vp K v^{(Vr K v)} (an N)$ 
apply (erule exE)
apply (drule-tac x = M in spec)
apply (erule exE, erule conjE)
apply (rotate-tac -3,
  frule-tac x = m in spec)
apply (thin-tac  $\forall n m. M < n \wedge M < m \rightarrow$ 
   $f n \pm -_a (f m) \in (vp K v)^{(Vr K v)} (an L)$ )
apply (subgoal-tac  $M < m \wedge (\forall ma. M < ma \rightarrow$ 
   $n\text{-val } K v (f m) = n\text{-val } K v (f ma))$ )
apply blast
apply simp

apply (rule allI, rule impI)
apply (rotate-tac -2)
apply (drule-tac x = ma in spec)
apply simp

apply (frule Vr-ring[of v],
  frule-tac n = an L in vp-apow-ideal[of v], simp)
apply (frule-tac I = vp K v^{(Vr K v)} (an L) and x = f m  $\pm -_a (f ma)$ 
  in Ring.ideal-inv1-closed[of Vr K v], assumption+) apply (
  frule-tac I = vp K v^{(Vr K v)} (an L) and
  h = f m  $\pm -_a (f ma)$  in Ring.ideal-subset[of Vr K v],
  assumption+, simp add:Vr-mOp-f-mOp)
apply (frule-tac x = m in spec,
  drule-tac x = ma in spec) apply (
  cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
  frule-tac x = f ma in aGroup.ag-mOp-closed[of K], assumption+,
  frule-tac x = f m and y =  $-_a (f ma)$  in aGroup.ag-p-inv[of K],
  assumption+, simp only:aGroup.ag-inv-inv,
  frule-tac x = f m in aGroup.ag-mOp-closed[of K], assumption+,
  simp add:aGroup.ag-pOp-commute,
  thin-tac  $-_a (f m \pm -_a (f ma)) = f ma \pm -_a (f m)$ ,
  thin-tac f m  $\pm -_a (f ma) \in vp K v^{(Vr K v)} (an L)$ )

```

```

apply (frule-tac  $x = f ma \pm -_a (f m)$  and  $n = an L$  in  $n\text{-value-}x\text{-}1$ [of
 $v]$ , simp) apply assumption apply (
  frule  $n\text{-val-valuation}$ [of  $v$ ],
  frule-tac  $x = f m$  and  $y = f ma \pm -_a (f m)$  in  $\text{value-less-eq}$ [of
 $n\text{-val } K v$ ], assumption+) apply (simp add:aGroup.ag-pOp-closed)
apply (
  rule-tac  $x = n\text{-val } K v (f m)$  and  $y = an L$  and
   $z = n\text{-val } K v (f ma \pm -_a (f m))$  in
  aless-le-trans, assumption+)
apply (frule-tac  $x = f ma \pm -_a (f m)$  in  $Vr\text{-mem-}f\text{-mem}$ [of  $v$ ])
apply (simp add:Ring.ideal-subset)
apply (frule-tac  $x = f m$  and  $y = f ma \pm -_a (f m)$  in
  aGroup.ag-pOp-commute[of  $K$ ], assumption+)
apply (simp add:aGroup.ag-pOp-assoc,
  simp add:aGroup.ag-l-inv1, simp add:aGroup.ag-r-zero)
done

lemma (in Corps) no-limit-zero-Cauchy1:[valuation  $K v$ ;  $\forall j. f j \in \text{carrier } K$ ;
 $\text{Cauchy}_K v f; \neg (\lim_K v f \mathbf{0})] \implies \exists N M. (\forall m. N < m \longrightarrow v(f M) = v(f m))$ 
apply (frule no-limit-zero-Cauchy[of  $v f$ ], assumption+)
apply (erule exE)+
apply (subgoal-tac  $\forall m. N < m \longrightarrow v(f M) = v(f m)$ ) apply blast
apply (rule allI, rule impI)
apply (frule-tac  $x = M$  in spec,
  drule-tac  $x = m$  in spec,
  drule-tac  $x = m$  in spec, simp)
apply (simp add:n-val[THEN sym, of  $v$ ])
done

definition
subfield ::  $[-, ('b, 'm1) \text{ Ring-scheme}] \Rightarrow \text{bool}$  where
subfield  $K K' \longleftrightarrow \text{Corps } K' \wedge \text{carrier } K \subseteq \text{carrier } K' \wedge$ 
 $\text{idmap } (\text{carrier } K) \in rHom K K'$ 

definition
v-completion ::  $['b \Rightarrow \text{ant}, 'b \Rightarrow \text{ant}, -, ('b, 'm) \text{ Ring-scheme}] \Rightarrow \text{bool}$ 
 $((4\text{Completion\_ - - -}) [90, 90, 90, 91] 90) \text{ where}$ 
Completion $_v v' K K' \longleftrightarrow \text{subfield } K K' \wedge$ 
 $\text{Complete}_{v'} K' \wedge (\forall x \in \text{carrier } K. v x = v' x) \wedge$ 
 $(\forall x \in \text{carrier } K'. (\exists f. \text{Cauchy}_K v f \wedge \lim_{K'} v' f x))$ 

lemma (in Corps) subfield-zero:[Corps  $K'$ ; subfield  $K K'] \implies \mathbf{0}_K = \mathbf{0}_{K'}$ 
apply (simp add:subfield-def, (erule conjE)+)
apply (simp add:rHom-def, (erule conjE)+)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of  $K$ ],
  frule Corps.field-is-ring[of  $K'$ ], frule Ring.ring-is-ag[of  $K'$ ])
apply (frule aHom-0-0[of  $K K' I_K$ ], assumption+)
apply (frule aGroup.ag-inc-zero[of  $K$ ], simp add:idmap-def)
done

```

```

lemma (in Corps) subfield-pOp:[[Corps K'; subfield K K'; x ∈ carrier K;
y ∈ carrier K] ⇒ x ± y = x ±K' y
apply (cut-tac field-is-ring, frule Corps.field-is-ring[of K'],
frule Ring.ring-is-ag[of K], frule Ring.ring-is-ag[of K'])
apply (simp add:subfield-def, erule conjE, simp add:rHom-def,
frule conjunct1)
apply (thin-tac I_K ∈ aHom K K' ∧
(∀x∈carrier K. ∀y∈carrier K. I_K (x ·r y) = I_K x ·rK' I_K y) ∧
I_K 1r = 1rK')
apply (frule aHom-add[of K K' I_K x y], assumption+,
frule aGroup.ag-pOp-closed[of K x y], assumption+,
simp add:idmap-def)
done

lemma (in Corps) subfield-mOp:[[Corps K'; subfield K K'; x ∈ carrier K] ⇒
¬a x = ¬aK' x
apply (cut-tac field-is-ring, frule Corps.field-is-ring[of K'],
frule Ring.ring-is-ag[of K], frule Ring.ring-is-ag[of K'])
apply (simp add:subfield-def, erule conjE, simp add:rHom-def,
frule conjunct1)
apply (thin-tac I_K ∈ aHom K K' ∧
(∀x∈carrier K. ∀y∈carrier K. I_K (x ·r y) = I_K x ·rK' I_K y) ∧
I_K 1r = 1rK')
apply (frule aHom-inv-inv[of K K' I_K x], assumption+,
frule aGroup.ag-mOp-closed[of K x], assumption+)
apply (simp add:idmap-def)
done

lemma (in Corps) completion-val-eq:[[Corps K'; valuation K v; valuation K' v';
x ∈ carrier K; Completionv v' K K']] ⇒ v x = v' x
apply (unfold v-completion-def, (erule conjE)+)
apply simp
done

lemma (in Corps) completion-subset:[[Corps K'; valuation K v; valuation K' v';
Completionv v' K K']] ⇒ carrier K ⊆ carrier K'
apply (unfold v-completion-def, (erule conjE)+)
apply (simp add:subfield-def)
done

lemma (in Corps) completion-subfield:[[Corps K'; valuation K v;
valuation K' v'; Completionv v' K K']] ⇒ subfield K K'
apply (simp add:v-completion-def)
done

lemma (in Corps) subfield-sub:subfield K K' ⇒ carrier K ⊆ carrier K'
apply (simp add:subfield-def)
done

```

```

lemma (in Corps) completion-Vring-sub: $\llbracket \text{Corps } K'; \text{valuation } K \ v;$ 
 $\text{valuation } K' \ v'; \text{Completion}_{v \ v'} \ K \ K \rrbracket \implies$ 
 $\text{carrier} (\text{Vr } K \ v) \subseteq \text{carrier} (\text{Vr } K' \ v')$ 
apply (rule subsetI,
  frule completion-subset[of K' v v'], assumption+,
  frule-tac x = x in Vr-mem-f-mem[of v], assumption+,
  frule-tac x = x in completion-val-eq[of K' v v'],
  assumption+)
apply (frule-tac x1 = x in val-pos-mem-Vr[THEN sym, of v],
  assumption+, simp,
  frule-tac c = x in subsetD[of carrier K carrier K'], assumption+,
  simp add:Corps.val-pos-mem-Vr[of K' v'])
done

lemma (in Corps) completion-idmap-rHom: $\llbracket \text{Corps } K'; \text{valuation } K \ v;$ 
 $\text{valuation } K' \ v'; \text{Completion}_{v \ v'} \ K \ K \rrbracket \implies$ 
 $I (\text{Vr } K \ v) \in \text{rHom} (\text{Vr } K \ v) (\text{Vr } K' \ v')$ 
apply (frule completion-Vring-sub[of K' v v'],
  assumption+,
  frule completion-subfield[of K' v v'],
  assumption+,
  frule Vr-ring[of v],
  frule Ring.ring-is-ag[of Vr K v],
  frule Corps.Vr-ring[of K' v'], assumption+,
  frule Ring.ring-is-ag[of Vr K' v'])
apply (simp add:rHom-def)
apply (rule conjI)
apply (simp add:aHom-def,
  rule conjI,
  simp add:idmap-def, simp add:subsetD)
apply (rule conjI)
apply (simp add:idmap-def extensional-def)
apply ((rule ballI)+) apply (
  frule-tac x = a and y = b in aGroup.ag-pOp-closed, assumption+,
  simp add:idmap-def,
  frule-tac c = a in subsetD[of carrier (Vr K v)
    carrier (Vr K' v')], assumption+,
  frule-tac c = b in subsetD[of carrier (Vr K v)
    carrier (Vr K' v')], assumption+,
  simp add:Vr-pOp-f-pOp,
  frule-tac x = a in Vr-mem-f-mem[of v], assumption,
  frule-tac x = b in Vr-mem-f-mem[of v], assumption,
  simp add:Corps.Vr-pOp-f-pOp,
  simp add:subfield-pOp)
apply (rule conjI)
apply ((rule ballI)+,
  frule-tac x = x and y = y in Ring.ring-tOp-closed, assumption+,
  simp add:idmap-def, simp add:subfield-def, erule conjE)

```

```

apply (frule-tac c = x in subsetD[of carrier (Vr K v)
                                carrier (Vr K' v')], assumption+,
      frule-tac c = y in subsetD[of carrier (Vr K v)
                                carrier (Vr K' v')], assumption+)
apply (simp add: Vr-tOp-f-tOp Corps.Vr-tOp-f-tOp)
apply (frule-tac x = x in Vr-mem-f-mem[of v], assumption+,
      frule-tac x = y in Vr-mem-f-mem[of v], assumption+,
      frule-tac x = x in Corps.Vr-mem-f-mem[of K' v'], assumption+,
      frule-tac x = y in Corps.Vr-mem-f-mem[of K' v'], assumption+,
      cut-tac field-is-ring, frule Corps.field-is-ring[of K'],
      frule-tac x = x and y = y in Ring.ring-tOp-closed[of K], assumption+)
apply (frule-tac x = x and y = y in rHom-tOp[of K K' - - I_K],
      assumption+, simp add:idmap-def)
apply (frule Ring.ring-one[of Vr K v], simp add:idmap-def)
apply (simp add: Vr-1-f-1 Corps.Vr-1-f-1)
apply (simp add:subfield-def, (erule conjE)+)
apply (cut-tac field-is-ring, frule Corps.field-is-ring[of K'],
      frule Ring.ring-one[of K],
      frule rHom-one[of K K' I_K], assumption+, simp add:idmap-def)
done

lemma (in Corps) completion-vpr-sub:[Corps K'; valuation K v; valuation K' v';
                                         Completion_v v' K K'] ==> vp K v ⊆ vp K' v'
apply (rule subsetI,
      frule completion-subset[of K' v v'], assumption+,
      frule Vr-ring[of v], frule vp-ideal[of v],
      frule-tac h = x in Ring.ideal-subset[of Vr K v vp K v],
      assumption+,
      frule-tac x = x in Vr-mem-f-mem[of v], assumption+,
      frule-tac x = x in completion-val-eq[of K' v v'],
      assumption+)
apply (frule completion-subset[of K' v v'],
      assumption+,
      frule-tac c = x in subsetD[of carrier K carrier K'], assumption+,
      simp add:Corps.vp-mem-val-poss vp-mem-val-poss)
done

lemma (in Corps) val-v-completion:[Corps K'; valuation K v; valuation K' v';
                                         x ∈ carrier K'; x ≠ 0_K'; Completion_v v' K K'] ==>
  ∃f. (Cauchy_K v f) ∧ (∃N. (∀m. N < m → v (f m) = v' x))
apply (simp add:v-completion-def, erule conjE, (erule conjE)+)
apply (rotate-tac -1, drule-tac x = x in bspec, assumption+,
      erule exE, erule conjE,
      subgoal-tac ∃N. ∀m. N < m → v (f m) = v' x, blast)
thm Corps.limit-val
apply (frule-tac f = f and v = v' in Corps.limit-val[of K' x],
      assumption+,
      unfold Cauchy-seq-def, frule conjunct1, fold Cauchy-seq-def)
apply (rule allI, drule-tac x = j in spec,

```

```

simp add:subfield-def, erule conjE, simp add:subsetD, assumption+)
apply (simp add:Cauchy-seq-def)
done

lemma (in Corps) v-completion-v-limit:[Corps K'; valuation K v;
x ∈ carrier K; subfield K K'; Completev' K'; ∀ j. f j ∈ carrier K;
valuation K' v'; ∀ x ∈ carrier K. v x = v' x; limK' v' f x] ⇒ limK v f x
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule Corps.field-is-ring[of K'], frule Ring.ring-is-ag[of K'],
subgoal-tac ∀ j. f j ∈ carrier K',
unfold subfield-def, frule conjunct2, fold subfield-def, erule conjE)
apply (frule subsetD[of carrier K carrier K' x], assumption+,
simp add:limit-diff-val[of x f],
subgoal-tac ∀ n. f n ±K' -aK' x = f n ± -a x)
apply (rule allI)
apply (simp add:limit-def)
apply (rotate-tac 6, drule-tac x = N in spec,
erule exE)
apply (subgoal-tac ∀ n>M. an N ≤ v'(f n ± -a x),
subgoal-tac ∀ n. (v' (f n ± -a x) = v (f n ± -a x)), simp, blast)
apply (rule allI)
apply (frule-tac x = f n ± -a x in bspec,
rule aGroup.ag-pOp-closed, assumption+, simp,
rule aGroup.ag-mOp-closed, assumption+) apply simp
apply (rule allI, rule impI)
apply (frule-tac v = v' and n = an N and x = f n ± -a x in
Corps.n-value-x-1[of K'], assumption+, simp, simp)
apply (frule-tac v = v' and x = f n ± -a x in Corps.n-val-le-val[of K'],
assumption+)
apply (cut-tac x = f n and y = -a x in aGroup.ag-pOp-closed, assumption,
simp, rule aGroup.ag-mOp-closed, assumption+, simp add:subsetD)
apply (subst Corps.val-pos-n-val-pos[of K' v'], assumption+)
apply (cut-tac x = f n and y = -a x in aGroup.ag-pOp-closed, assumption,
simp, rule aGroup.ag-mOp-closed, assumption+, simp add:subsetD)
apply (rule-tac i = 0 and j = an N and k = n-val K' v' (f n ± -a x) in
ale-trans, simp+, rule allI)
apply (subst subfield-pOp[of K'], assumption+, simp+,
rule aGroup.ag-mOp-closed, assumption+)
apply (simp add:subfield-mOp[of K'])
apply (cut-tac subfield-sub[of K'], simp add:subsetD, assumption+)
done

lemma (in Corps) Vr-idmap-aHom:[Corps K'; valuation K v; valuation K' v';
subfield K K'; ∀ x ∈ carrier K. v x = v' x] ⇒
I(Vr K v) ∈ aHom (Vr K v) (Vr K' v')
apply (simp add:aHom-def)
apply (subgoal-tac I(Vr K v) ∈ carrier (Vr K v) → carrier (Vr K' v'))
apply simp
apply (rule conjI)

```

```

apply (simp add:idmap-def)
apply (rule ballI)+
apply (frule Vr-ring[of v],
       frule Ring.ring-is-ag[of Vr K v],
       frule Corps.Vr-ring[of K' v'], assumption+,
       frule Ring.ring-is-ag[of Vr K' v'])
apply (frule-tac x = a and y = b in aGroup.ag-pOp-closed[of Vr K v],
       assumption+,
       frule-tac x = a in funcset-mem[of I(Vr K v)
                                         carrier (Vr K v) carrier (Vr K' v')], assumption+,
       frule-tac x = b in funcset-mem[of I(Vr K v)
                                         carrier (Vr K v) carrier (Vr K' v')], assumption+,
       frule-tac x = (I(Vr K v)) a and y = (I(Vr K v)) b in
       aGroup.ag-pOp-closed[of Vr K' v'], assumption+,
       simp add: Vr-pOp-f-pOp)
apply (simp add:idmap-def, simp add:subfield-def, erule conjE,
       simp add:rHom-def, frule conjunct1,
       thin-tac  $I_K \in aHom K K'$   $\wedge$ 
               ( $\forall x \in carrier K. \forall y \in carrier K. I_K(x \cdot_r y) = I_K x \cdot_{rK'} I_K y$ )  $\wedge$ 
                $I_K 1_r = 1_{rK'}$ )
apply (simp add:Corps.Vr-pOp-f-pOp[of K' v'])
apply (frule-tac x = a in Vr-mem-f-mem[of v], assumption+,
       frule-tac x = b in Vr-mem-f-mem[of v], assumption+)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
       frule Corps.field-is-ring[of K'], frule Ring.ring-is-ag[of K'])
apply (frule-tac a = a and b = b in aHom-add[of K K' I_K], assumption+,
       frule-tac x = a and y = b in aGroup.ag-pOp-closed[of K], assumption+,
       simp add:idmap-def)
apply (rule Pi-I,
       drule-tac x = x in bspec, simp add: Vr-mem-f-mem)
apply (simp add:idmap-def)
apply (frule-tac x1 = x in val-pos-mem-Vr[THEN sym, of v],
       simp add: Vr-mem-f-mem, simp)
apply (subst Corps.val-pos-mem-Vr[THEN sym, of K' v'], assumption+,
       frule-tac x = x in Vr-mem-f-mem[of v], assumption+,
       frule subfield-sub[of K'], simp add:subsetD)
apply assumption
done

lemma amult-pos-pos:  $0 \leq a \implies 0 \leq a * an N$ 
apply (case-tac N = 0, simp add:an-0)
apply (case-tac a =  $\infty$ , simp)
apply (frule apos-neq-minf[of a])
apply (subst ant-tna[THEN sym, of a], simp)
apply (subst amult-0-r, simp)
apply (case-tac a =  $\infty$ , simp add:an-def)
apply (frule apos-neq-minf[of a])
apply (subst ant-tna[THEN sym, of a], simp)
apply (case-tac a = 0, simp)

```

```

apply (simp add:ant-0 an-def amult-0-l)
apply (cut-tac amult-pos1[of tna a an N])
apply (simp add:ant-tna)
apply (rule-tac ale-trans[of 0 an N a * an N], simp+)
apply (frule ale-neq-less[of 0 a], rule not-sym, assumption)
apply (subst aless-zless[THEN sym, of 0 tna a], simp add:ant-tna ant-0)
apply simp
done

lemma (in Corps) Cauchy-down:[Corps K'; valuation K v; valuation K' v'];
  subfield K K'; ∀ x∈carrier K. v x = v' x; ∀ j. f j ∈ carrier K; CauchyK' v' f]
  ==> CauchyK v f
apply (simp add:Cauchy-seq-def, rule allI, erule conjE)
apply (rotate-tac -1, drule-tac
  x = na (Lv K v) * N in spec,
  erule exE,
  subgoal-tac ∀ n m. M < n ∧ M < m →
    f n ± (-a (f m)) ∈ vp K v (Vr K v) (an N), blast)
apply (simp add:amult-an-an) apply (simp add:an-na-Lv)
apply ((rule allI)+, rule impI, erule conjE) apply (
  rotate-tac -3, drule-tac x = n in spec,
  rotate-tac -1, drule-tac x = m in spec,
  simp)
apply (rotate-tac 7,
  frule-tac x = n in spec,
  drule-tac x = m in spec)
apply (simp add:subfield-mOp[THEN sym],
  cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
  frule-tac x = f m in aGroup.ag-mOp-closed[of K], assumption+)
apply (simp add:subfield-pOp[THEN sym])
apply (frule-tac x = f n and y = -a f m in aGroup.ag-pOp-closed,
  assumption+,
  frule subfield-sub[of K'],
  frule-tac c = f n ± -a f m in subsetD[of carrier K carrier K'],
  assumption+)
apply (frule Lv-pos[of v],
  frule aless-imp-le[of 0 Lv K v])
apply (frule-tac N = N in amult-pos-pos[of Lv K v])
apply (frule-tac n = (Lv K v) * an N and x = f n ± -a f m in
  Corps.n-value-x-1[of K' v'], assumption+)
apply (frule-tac x = f n ± -a f m in Corps.n-val-le-val[of K' v'],
  assumption+,
  frule-tac j = Lv K v * an N and k = n-val K' v' (f n ± -a f m) in
  ale-trans[of 0], assumption+, simp add:Corps.val-pos-n-val-pos)
apply (frule-tac i = Lv K v * an N and j = n-val K' v' (f n ± -a f m)
  and k = v' (f n ± -a f m) in ale-trans, assumption+,
  thin-tac n-val K' v' (f n ± -a f m) ≤ v' (f n ± -a f m),
  thin-tac Lv K v * an N ≤ n-val K' v' (f n ± -a f m))
apply (rotate-tac 1,

```

```

drule-tac  $x = f n \pm -_a f m$  in bspec, assumption,
rotate-tac -1, drule sym, simp)
apply (frule-tac  $v1 = v$  and  $x1 = f n \pm -_a f m$  in n-val[THEN sym],
assumption)
apply simp
apply (simp only:amult-commute[of - Lv K v])
apply (frule Lv-z[of v], erule exE)

apply (cut-tac  $w = z$  and  $x = an N$  and  $y = n\text{-val } K v (f n \pm -_a f m)$  in
amult-pos-mono-l,
cut-tac  $m = 0$  and  $n = z$  in aless-zless, simp add:ant-0)
apply simp
apply (rule-tac  $x = f n \pm -_a (f m)$  and  $n = an N$  in n-value-x-2[of v],
assumption+)
apply (subst val-pos-mem-Vr[THEN sym, of v], assumption+)
apply (subst val-pos-n-val-pos[of v], assumption+)
apply (rule-tac  $j = an N$  and  $k = n\text{-val } K v (f n \pm -_a f m)$  in
ale-trans[of 0], simp, assumption+)
apply simp
done

lemma (in Corps) Cauchy-up:[Corps K'; valuation K v; valuation K' v';
Completionv v' K K'; Cauchy K v f]  $\implies$  Cauchy K' v' f
apply (simp add:Cauchy-seq-def,
erule conjE,
rule conjI, unfold v-completion-def, frule conjunct1,
fold v-completion-def, rule allI, frule subfield-sub[of K'])
apply (simp add:subsetD)

apply (rule allI)
apply (rotate-tac -1, drule-tac  $x = na (Lv K' v') * N$ 
in spec, erule exE)
apply (subgoal-tac  $\forall n m. M < n \wedge M < m \longrightarrow$ 
 $f n \pm_{K'} (-_a K' (f m)) \in vp K' v' (Vr K' v') (an N)$ , blast,
(rule allI)+, rule impI, erule conjE)
apply (rotate-tac -3, drule-tac  $x = n$  in spec,
rotate-tac -1,
drule-tac  $x = m$  in spec, simp,
frule-tac  $x = n$  in spec,
drule-tac  $x = m$  in spec)
apply (unfold v-completion-def, frule conjunct1, fold v-completion-def,
cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule-tac  $x = f m$  in aGroup.ag-mOp-closed[of K], assumption+,
frule-tac  $x = f n$  and  $y = -_a (f m)$  in aGroup.ag-pOp-closed[of K],
assumption+)
apply (simp add:amult-an-an) apply (simp add:Corps.an-na-Lv)
apply (simp add:subfield-mOp, simp add:subfield-pOp) apply (
frule-tac  $x = f n \pm_{K'} -_a K' f m$  and  $n = (Lv K' v') * (an N)$ 
in n-value-x-1[of v])

```

```

apply (frule Corps.Lv-pos[of  $K' v'$ ], assumption+,
      frule Corps.Lv-z[of  $K' v'$ ],
      assumption, erule exE, simp,
      cut-tac  $n = N$  in an-nat-pos,
      frule-tac  $w1 = z$  and  $x1 = 0$  and  $y1 = an N$  in
                  amult-pos-mono-l[THEN sym], simp, simp add:amult-0-r)
apply assumption
apply (frule-tac  $x = f n \pm_{K'} -a_{K'} f m$  in n-val-le-val[of  $v$ ],
       assumption+)
apply (subst n-val[THEN sym, of  $v$ ], assumption+)
apply (frule Lv-pos[of  $v$ ], frule Lv-z[of  $v$ ], erule exE, simp)
apply (frule Corps.Lv-pos[of  $K' v'$ ], assumption+,
       frule Corps.Lv-z[of  $K' v'$ ], assumption, erule exE, simp,
       cut-tac  $n = N$  in an-nat-pos,
       frule-tac  $w1 = za$  and  $x1 = 0$  and  $y1 = an N$  in
                  amult-pos-mono-l[THEN sym], simp, simp add:amult-0-r)
apply (frule-tac  $j = ant za * an N$  and  $k = n\text{-val } K v (f n \pm_{K'} -a_{K'} (f m))$ 
       in ale-trans[of 0], assumption+)
apply (frule-tac  $w1 = z$  and  $x1 = 0$  and  $y1 = n\text{-val } K v (f n \pm_{K'} -a_{K'} (f m))$ 
       in amult-pos-mono-r[THEN sym], simp, simp add:amult-0-l)
apply (frule-tac  $i = Lv K' v' * an N$  and  $j = n\text{-val } K v (f n \pm_{K'} -a_{K'} (f m))$ 
       and  $k = v (f n \pm_{K'} -a_{K'} (f m))$  in ale-trans, assumption+)
apply (thin-tac  $f n \pm_{K'} -a_{K'} (f m) \in vp K v (Vr K v) (Lv K' v') * (an N)$ ,
       thin-tac  $Lv K' v' * an N \leq n\text{-val } K v (f n \pm_{K'} -a_{K'} (f m))$ ,
       thin-tac  $n\text{-val } K v (f n \pm_{K'} -a_{K'} (f m)) \leq v (f n \pm_{K'} -a_{K'} (f m))$ )
apply (simp add:v-completion-def, (erule conjE)+)
apply (thin-tac  $\forall x \in carrier K. v x = v' x$ ,
       thin-tac  $\forall x \in carrier K'. \exists f. Cauchy K v f \wedge \lim_{K'} v' f x$ )
apply (frule subfield-sub[of  $K$ ],
       frule-tac  $c = f n \pm_{K'} -a_{K'} (f m)$  in
                  subsetD[of carrier  $K$  carrier  $K'$ ], assumption+)
apply (simp add:Corps.n-val[THEN sym, of  $K' v'$ ])
apply (simp add:amult-commute[- Lv  $K' v'$ ])
apply (frule Corps.Lv-pos[of  $K' v'$ ], assumption,
       frule Corps.Lv-z[of  $K' v'$ ], assumption+, erule exE, simp)
apply (simp add:amult-pos-mono-l)

apply (rule-tac  $x = f n \pm_{K'} -a_{K'} (f m)$  and  $n = an N$  in
               Corps.n-value-x-2[of  $K' v'$ ], assumption+)
apply (cut-tac  $n = N$  in an-nat-pos)
apply (frule-tac  $j = an N$  and  $k = n\text{-val } K' v' (f n \pm_{K'} -a_{K'} (f m))$  in
               ale-trans[of 0], assumption+)
apply (simp add:Corps.val-pos-n-val-pos[THEN sym, of  $K' v'$ ])
apply (simp add:Corps.val-pos-mem-Vr) apply assumption apply simp
done

lemma max-gtTr:( $n :: nat$ ) < max (Suc  $n$ ) (Suc  $m$ )  $\wedge$   $m < max (Suc n) (Suc m)$ 
by (simp add:max-def)

```

lemma (in Corps) completion-approx:[Corps K'; valuation K v; valuation K' v';
 $\text{Completion}_{v v'} K K'$; $x \in \text{carrier} (\text{Vr } K' v')$] \implies
 $\exists y \in \text{carrier} (\text{Vr } K v). (y \pm_{K'} -a_{K'} x) \in (\text{vp } K' v')$

apply (frule Corps.Vr-mem-f-mem [of K' v' x], assumption+,
frule Corps.val-pos-mem-Vr[THEN sym, of K' v' x], assumption+,
simp add:v-completion-def, (erule conjE)+,
rotate-tac -1, drule-tac x = x in bspec, assumption+,
erule exE, erule conjE)

apply (unfold Cauchy-seq-def, frule conjunct1, fold Cauchy-seq-def)

apply (case-tac x = $\mathbf{0}_{K'}$,
simp, frule Corps.field-is-ring[of K'],
frule Ring.ring-is-ag[of K'],
subgoal-tac $\mathbf{0}_{K'} \in \text{carrier} (\text{Vr } K v)$,
subgoal-tac $(\mathbf{0}_{K'} \pm_{K'} -a_{K'} \mathbf{0}_{K'}) \in \text{vp } K' v'$, blast,
frule aGroup.ag-inc-zero[of K'], simp add:aGroup.ag-r-inv1,
simp add:Corps.Vr-0-f-0[THEN sym, of K' v'],
frule Corps.Vr-ring[of K' v'], assumption+,
frule Corps.vp-ideal[of K' v'], assumption+,
simp add:Ring.ideal-zero,
simp add:subfield-zero[THEN sym, of K'],
cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
frule aGroup.ag-inc-zero[of K],
simp add:Vr-0-f-0[THEN sym, of v],
frule Vr-ring[of v], simp add:Ring.ring-zero)

apply (frule-tac f = f in Corps.limit-val[of K' x - v'],
assumption+)

apply (rule allI, rotate-tac -2, frule-tac x = j in spec,
frule subfield-sub[of K'], simp add:subsetD, assumption+)

apply (erule exE)

apply (simp add:limit-def,
frule Corps.Vr-ring[of K' v'], assumption+,
rotate-tac 10,
drule-tac x = Suc 0 in spec, erule exE,
rotate-tac 1,
frule-tac x = N and y = M in two-inequalities, assumption+,
thin-tac $\forall n > N. v'(f n) = v' x$,
thin-tac $\forall n > M. f n \pm_{K'} -a_{K'} x \in \text{vp } K' v' (\text{Vr } K' v') (\text{an } (\text{Suc } 0))$)

apply (frule Corps.vp-ideal[of K' v'], assumption+,
simp add:Ring.r-apow-Suc[of Vr K' v' vp K' v'])

apply (drule-tac x = N + M + 1 in spec, simp,
drule-tac x = N + M + 1 in spec, simp,
erule conjE)

apply (drule-tac x = f (Suc (N + M)) in bspec, assumption+)

apply simp

apply (cut-tac x = f (Suc (N + M)) in val-pos-mem-Vr[of v], assumption+)

apply simp apply blast

done

```

lemma (in Corps) res-v-completion-surj:[ Corps K'; valuation K v;
valuation K' v'; Completionv v' K K'] ==>
surjec( Vr K v), (qring (Vr K' v') (vp K' v')) 
(compos (Vr K v) (pj (Vr K' v') (vp K' v')) (I(Vr K v)))
apply (frule Vr-ring[of v],
frule Ring.ring-is-ag[of Vr K v],
frule Corps.Vr-ring[of K' v'], assumption+,
frule Ring.ring-is-ag[of Vr K' v'],
frule Ring.ring-is-ag[of Vr K v])
apply (frule Corps.vp-ideal[of K' v'], assumption+,
frule Ring.qring-ring[of Vr K' v' vp K' v'], assumption+)
apply (simp add:surjec-def)
apply (frule aHom-compos[of Vr K v Vr K' v'
qring (Vr K' v') (vp K' v') I(Vr K v)
pj (Vr K' v') (vp K' v')], assumption+, simp add:Ring.ring-is-ag)
apply (rule Vr-idmap-aHom, assumption+) apply (simp add:completion-subfield,
simp add:v-completion-def) apply (
frule pj-Hom[of Vr K' v' vp K' v'], assumption+) apply (
simp add:rHom-def, simp)
apply (rule surj-to-test)
apply (simp add:aHom-def)
apply (rule ballI)
apply (thin-tac Ring (Vr K' v' /r vp K' v'),
thin-tac compos (Vr K v) (pj (Vr K' v') (vp K' v')) (I(Vr K v)) ∈
aHom (Vr K v) (Vr K' v' /r vp K' v'))
apply (simp add:Ring.qring-carrier)
apply (erule bxE)
apply (frule-tac x = a in completion-approx[of K' v v'],
assumption+, erule bxE)
apply (subgoal-tac compos (Vr K v) (pj (Vr K' v')
(vp K' v')) ((I(Vr K v))) y = b, blast)
apply (simp add:compos-def compose-def idmap-def)
apply (frule completion-Vring-sub[of K' v v'], assumption+)
apply (frule-tac c = y in subsetD[of carrier (Vr K v) carrier (Vr K' v')],
assumption+)
apply (frule-tac x = y in pj-mem[of Vr K' v' vp K' v'], assumption+, simp,
thin-tac pj (Vr K' v') (vp K' v') y = y ⊕(Vr K' v') (vp K' v'))
apply (rotate-tac -5, frule sym, thin-tac a ⊕(Vr K' v') (vp K' v') = b,
simp)
apply (rule-tac b1 = y and a1 = a in Ring.ar-coset-same1[THEN sym,
of Vr K' v' vp K' v'], assumption+)
apply (frule Ring.ring-is-ag[of Vr K' v'],
frule-tac x = a in aGroup.ag-mOp-closed[of Vr K' v'],
assumption+)
apply (simp add:Corps.Vr-mOp-f-mOp, simp add:Corps.Vr-pOp-f-pOp)
done

```

```

lemma (in Corps) res-v-completion-ker: $\llbracket \text{Corps } K'; \text{valuation } K v;$ 
 $\text{valuation } K' v'; \text{Completion}_{v v'} K K' \rrbracket \implies$ 
 $\text{ker}(\text{Vr } K v), (\text{qring } (\text{Vr } K' v') (\text{vp } K' v'))$ 
 $(\text{compos } (\text{Vr } K v) (\text{pj } (\text{Vr } K' v') (\text{vp } K' v')) (\text{I}_{(\text{Vr } K v)})) = \text{vp } K v$ 
apply (rule equalityI)
apply (rule subsetI)
apply (simp add:ker-def, (erule conjE)+)
apply (frule Corps.Vr-ring[of K' v'], assumption+,
       frule Corps.vp-ideal[of K' v'], assumption+,
       frule Ring.qring-ring[of Vr K' v' vp K' v'], assumption+,
       simp add:Ring.qring-zero)
apply (simp add:compos-def, simp add:compose-def, simp add:idmap-def)
apply (frule completion-Vring-sub[of K' v v'], assumption+)
apply (frule-tac c = x in subsetD[of carrier (Vr K v) carrier (Vr K' v')],
       assumption+)
apply (simp add:pj-mem)
apply (frule-tac a = x in Ring.qring-zero-1[of Vr K' v' - vp K' v'],
       assumption+)
apply (subst vp-mem-val-poss[of v], assumption+)
apply (simp add:Vr-mem-f-mem)
apply (frule-tac x = x in Corps.vp-mem-val-poss[of K' v'],
       assumption+, simp add:Corps.Vr-mem-f-mem, simp)
apply (frule-tac x = x in Vr-mem-f-mem[of v], assumption+)
apply (frule-tac x = x in completion-val-eq[of K' v v'],
       assumption+, simp)
apply (rule subsetI)
apply (simp add:ker-def)
apply (frule Vr-ring[of v])
apply (frule vp-ideal[of v])
apply (frule-tac h = x in Ring.ideal-subset[of Vr K v vp K v],
       assumption+, simp)
apply (frule Corps.Vr-ring[of K' v'], assumption+,
       frule Corps.vp-ideal[of K' v'], assumption+,
       simp add:Ring.qring-zero)
apply (simp add:compos-def compose-def idmap-def)
apply (frule completion-Vring-sub[of K' v v'],
       assumption+, frule-tac c = x in subsetD[of carrier (Vr K v)
       carrier (Vr K' v')], assumption+)
apply (simp add:pj-mem)
apply (frule completion-vpr-sub[of K' v v'], assumption+,
       frule-tac c = x in subsetD[of vp K v vp K' v'], assumption+)
apply (simp add:Ring.ar-coset-same4[of Vr K' v' vp K' v'])
done

lemma (in Corps) completion-res-qring-isom: $\llbracket \text{Corps } K'; \text{valuation } K v;$ 
 $\text{valuation } K' v'; \text{Completion}_{v v'} K K' \rrbracket \implies$ 
 $r\text{-isom } ((\text{Vr } K v) /_r (\text{vp } K v)) ((\text{Vr } K' v') /_r (\text{vp } K' v'))$ 
apply (subst r-isom-def)
apply (frule res-v-completion-surj[of K' v v'], assumption+)

```

```

apply (frule Vr-ring[of v],
         frule Corps.Vr-ring[of K' v], assumption+, 
         frule Corps.vp-ideal[of K' v], assumption+, 
         frule Ring.qring-ring[of Vr K' v' vp K' v'], assumption+) 
apply (frule rHom-compos[of Vr K v Vr K' v' (Vr K' v' /r vp K' v')
          (I(Vr K v) pj (Vr K' v') (vp K' v')], assumption+) 
apply (simp add:completion-idmap-rHom)
apply (simp add:pj-Hom)
apply (frule surjec-ind-bijec [of Vr K v (Vr K' v' /r vp K' v')
         compos (Vr K v) (pj (Vr K' v') (vp K' v')) (I(Vr K v))], assumption+) 
apply (frule ind-hom-rhom[of Vr K v (Vr K' v' /r vp K' v')
         compos (Vr K v) (pj (Vr K' v') (vp K' v')) (I(Vr K v))], assumption+) 
apply (simp add:res-v-completion-ker) apply blast
done

```

expansion of x in a complete field K , with normal valuation v . Here we suppose t is an element of K satisfying the equation $v t = 1$.

definition

```

Kxa :: [-, 'b  $\Rightarrow$  ant, 'b]  $\Rightarrow$  'b set where
Kxa K v x = {y.  $\exists k \in carrier (Vr K v)$ . y = x ·R k}

```

primrec

```

partial-sum :: [('b, 'm) Ring-scheme, 'b, 'b  $\Rightarrow$  ant, 'b]
                   $\Rightarrow$  nat  $\Rightarrow$  'b
((5psum - - - -) [96,96,96,96,97]96)

```

where

```

psum-0: psum K x v t 0 = (csrp-fn (Vr K v) (vp K v)
  (pj (Vr K v) (vp K v) (x ·R tK-(tna (v x)))) ·R K (tK(tna (v x)))
| psum-Suc: psum K x v t (Suc n) = (psum K x v t n) ±K
  ((csrp-fn (Vr K v) (vp K v) (pj (Vr K v) (vp K v)
    ((x ±K −aK (psum K x v t n)) ·R K (tK−(tna (v x) + int (Suc n))))) ·R K
    (tK(tna (v x) + int (Suc n))))))

```

definition

```

expand-coeff :: [-, 'b  $\Rightarrow$  ant, 'b, 'b]
                   $\Rightarrow$  nat  $\Rightarrow$  'b
((5ecf- - - -) [96,96,96,96,97]96) where
ecf K v t x n = (if n = 0 then csrp-fn (Vr K v) (vp K v)
  (pj (Vr K v) (vp K v) (x ·R tK−(tna (v x)))))
  else csrp-fn (Vr K v) (vp K v) (pj (Vr K v)
  (vp K v) ((x ±K −aK (psum K x v t (n − 1))) ·R K (tK−(tna (v x) + int n)))))

```

definition

```

expand-term :: [-, 'b  $\Rightarrow$  ant, 'b, 'b]
                   $\Rightarrow$  nat  $\Rightarrow$  'b
((5etm- - - -) [96,96,96,96,97]96) where

```

$$etm_K v t x n = (ecf_K v t x n) \cdot_{rK} (t_K^{(tna(v x) + int n)})$$

```

lemma (in Corps) Kxa-val-ge: ``valuation K v; t ∈ carrier K; v t = 1``
  ==> Kxa K v (t_K^j) = {x. x ∈ carrier K ∧ (ant j) ≤ (v x)}
apply (frule val1-neq-0[of v t], assumption+)
apply (cut-tac field-is-ring)
apply (rule equalityI)
apply (rule subsetI,
      simp add:Kxa-def, erule bexE,
      frule-tac x = k in Vr-mem-f-mem[of v], assumption+,
      frule npowf-mem[of t j], simp,
      simp add:Ring.ring-tOp-closed)
apply (simp add:val-t2p) apply (simp add:val-exp[THEN sym])
apply (simp add:val-pos-mem-Vr[THEN sym, of v])
apply (frule-tac x = 0 and y = v k in aadd-le-mono[of - - ant j])
apply (simp add:aadd-0-l aadd-commute)
apply (rule subsetI, simp, erule conjE)
apply (simp add:Kxa-def)
apply (case-tac x = 0_K)
apply (frule Vr-ring[of v],
      frule Ring.ring-zero[of Vr K v],
      simp add:Vr-0-f-0,
      frule-tac x1 = t_K^j in Ring.ring-times-x-0[THEN sym, of K],
      simp add:npowf-mem, blast)
apply (frule val-exp[of v t j], assumption+, simp)
apply (frule field-potent-nonzero1[of t j],
      frule npowf-mem[of t j], assumption+)
apply (frule-tac y = v x in ale-diff-pos[of v (t_K^j)],
      simp add:diff-ant-def)
apply (cut-tac npowf-mem[of t j])
defer
apply assumption apply simp
apply (frule value-of-inv[THEN sym, of v t_K^j], assumption+)

apply (cut-tac invf-closed1[of t_K^j], simp, erule conjE)
apply (frule-tac x1 = x in val-t2p[THEN sym, of v - (t_K^j)^-K],
      assumption+, simp)
apply (frule-tac x = (t_K^j)^-K and y = x in Ring.ring-tOp-closed[of K],
      assumption+)
apply (simp add:Ring.ring-tOp-commute[of K - (t_K^j)^-K])
apply (frule-tac x = ((t_K^j)^-K) ·_r x in val-pos-mem-Vr[of v], assumption+,
      simp)
apply (frule-tac z = x in Ring.ring-tOp-assoc[of K t_K^j (t_K^j)^-K],
      assumption+)

```

```

apply (simp add:Ring.ring-tOp-commute[of K tKj (tKj)-K] linvf,
      simp add:Ring.ring-l-one, blast)
apply simp
done

lemma (in Corps) Kxa-pow-vpr: [| valuation K v; t ∈ carrier K; v t = 1;
(0::int) ≤ j] ==> Kxa K v (tKj) = (vp K v)(Vr K v) (ant j)
apply (frule val-surj-n-val[of v], blast)
apply (simp add:Kxa-val-ge)
apply (rule equalityI)
apply (rule subsetI, simp, erule conjE)
apply (rule-tac x = x in n-value-x-2[of v - (ant j)],
       assumption+)
apply (cut-tac ale-zle[THEN sym, of 0 j])
apply (frule-tac a = 0 ≤ j and b = ant 0 ≤ ant j in a-b-exchange,
       assumption+)
apply (thin-tac (0 ≤ j) = (ant 0 ≤ ant j), simp add:ant-0)
apply (frule-tac k = v x in ale-trans[of 0 ant j], assumption+)
apply (simp add:val-pos-mem-Vr) apply simp
apply (simp only:ale-zle[THEN sym, of 0 j], simp add:ant-0)
apply (rule subsetI)
apply simp
apply (frule-tac x = x in mem-vp-apow-mem-Vr[of v ant j])
apply (simp only:ale-zle[THEN sym, of 0 j], simp add:ant-0)
apply assumption
apply (simp add:Vr-mem-f-mem[of v])
apply (frule-tac x = x in n-value-x-1[of v ant j - ])
apply (simp only:ale-zle[THEN sym, of 0 j], simp add:ant-0)
apply assumption apply simp
done

lemma (in Corps) field-distribTr: [| a ∈ carrier K; b ∈ carrier K;
x ∈ carrier K; x ≠ 0] ==> a ± (-a (b ·r x)) = (a ·r (x-K) ± (-a b)) ·r x
apply (cut-tac field-is-ring,
       cut-tac invf-closed1[of x], simp, erule conjE) apply (
       simp add:Ring.ring-inv1-1[of K b x],
       frule Ring.ring-is-ag[of K],
       frule aGroup.ag-mOp-closed[of K b], assumption+)
apply (frule Ring.ring-tOp-closed[of K a x-K], assumption+,
       simp add:Ring.ring-distrib2, simp add:Ring.ring-tOp-assoc,
       simp add:linvf,
       simp add:Ring.ring-r-one)
apply simp
done

lemma a0-le-1 [simp]: (0::ant) ≤ 1
by (cut-tac a0-less-1, simp add:aless-imp-le)

lemma (in Corps) vp-mem-times-t: [| valuation K v; t ∈ carrier K; t ≠ 0;

```

```

 $v t = 1; x \in vp K v] \implies \exists a \in carrier (Vr K v). x = a \cdot_r t$ 
apply (frule Vr-ring[of v],
         frule vp-ideal[of v])
apply (frule val-surj-n-val[of v], blast)
apply (frule vp-mem-val-poss[of v x],
         frule-tac h = x in Ring.ideal-subset[of Vr K v vp K v],
         assumption+, simp add: Vr-mem-f-mem, simp)
apply (frule gt-a0-ge-1[of v x])
apply (subgoal-tac v t \leq v x)
apply (thin-tac 1 \leq v x)
apply (frule val-Rxa-gt-a-1[of v t x])
apply (subst val-pos-mem-Vr[THEN sym, of v t], assumption+)
apply simp
apply (simp add: vp-mem-Vr-mem) apply assumption+
apply (simp add: Rxa-def, erule bexE)
apply (cut-tac a0-less-1)
apply (subgoal-tac 0 \leq v t)
apply (frule val-pos-mem-Vr[of v t], assumption+)
apply (simp, simp add: Vr-tOp-f-tOp, blast, simp+)
done

lemma (in Corps) psum-diff-mem-Kxa:  $\llbracket \text{valuation } K v; t \in \text{carrier } K;$ 
 $v t = 1; x \in \text{carrier } K; x \neq 0 \rrbracket \implies$ 
 $(\text{psum } K x v t n) \in \text{carrier } K \wedge$ 
 $(x \pm (-_a (\text{psum } K x v t n))) \in Kxa K v (t_K((tna (v x)) + (1 + \text{int } n)))$ 
apply (frule val1-neq-0[of v t], assumption+)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K],
         frule Vr-ring[of v], frule vp-ideal[of v])
apply (induct-tac n)
apply (subgoal-tac x \cdot_r (t_K^- tna (v x)) \in \text{carrier } (Vr K v),
         rule conjI, simp, rule Ring.ring-tOp-closed[of K], assumption+,
         frule Ring.csrp-fn-mem[of Vr K v vp K v
         pj (Vr K v) (vp K v) (x \cdot_r (t_K^- tna (v x))), assumption+,
         simp add: pj-mem Ring.qring-carrier, blast,
         simp add: Vr-mem-f-mem, simp add: npowf-mem)
apply (simp,
         frule npowf-mem[of t tna (v x)], assumption+,
         frule field-potent-nonzero1[of t tna (v x)], assumption+,
         subst field-distribTr[of x - t_K^(tna (v x))], assumption+,
         frule Ring.csrp-fn-mem[of Vr K v vp K v
         pj (Vr K v) (vp K v) (x \cdot_r (t_K^- tna (v x))), assumption+,
         simp add: pj-mem Ring.qring-carrier, blast, simp add: Vr-mem-f-mem,
         simp add: npowf-mem, assumption)
apply (frule Ring.csrp-diff-in-vpr[of Vr K v vp K v
         x \cdot_r ((t_K^(tna (v x)))^-K)], assumption+)
apply (simp add: npowf-minus)

```

```

apply (simp add:npowf-minus)
apply (frule pj-Hom[of Vr K v vp K v], assumption+)
apply (frule rHom-mem[of pj (Vr K v) (vp K v) Vr K v Vr K v /r vp K v
         x ·r (tK- tna (v x))], assumption+)
apply (frule Ring.csrf-fn-mem[of Vr K v vp K v
         pj (Vr K v) (vp K v) (x ·r (tK- tna (v x)))], assumption+)
apply (frule Ring.ring-is-ag[of Vr K v],
         frule-tac x = csrf-fn (Vr K v) (vp K v) (pj (Vr K v) (vp K v)
         (x ·r (tK- tna (v x)))) in aGroup.ag-mOp-closed[of Vr K v], assumption+)
apply (simp add:Vr-pOp-f-pOp) apply (simp add:Vr-mOp-f-mOp)
apply (frule-tac x = x ·r (tK- tna (v x)) ± -a (csrf-fn (Vr K v) (vp K v)
         (pj (Vr K v) (vp K v) (x ·r (tK- tna (v x)))) in
         vp-mem-times-t[of v t], assumption+, erule bxE, simp)
apply (frule-tac x = a in Vr-mem-f-mem[of v], assumption+)
apply (simp add:Ring.ring-tOp-assoc[of K])
apply (simp add:npowf-exp-1-add[THEN sym, of t tna (v x)])
apply (simp add:add.commute)
apply (simp add:Kxa-def)
apply (frule npowf-mem[of t 1 + tna (v x)], assumption+)
apply (simp add:Ring.ring-tOp-commute[of K - tK(1 + tna (v x))])
apply blast
apply (frule npowf-mem[of t - tna (v x)], assumption+)
apply (frule Ring.ring-tOp-closed[of K x tK- tna (v x)], assumption+)
apply (subst val-pos-mem-Vr[THEN sym, of v], assumption+)
apply (simp add:val-t2p) apply (simp add:val-exp[THEN sym, of v t])
apply (simp add:aminus[THEN sym])
apply (frule value-in-aug-inf[of v x], assumption+,
         simp add:aug-inf-def)
apply (frule val-nonzero-noninf[of v x], assumption+,
         simp add:ant-tna)
apply (simp add:aadd-minus-r)

apply (erule conjE)
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (rule conjI)
apply simp
apply (rule aGroup.ag-pOp-closed, assumption+)
apply (rule Ring.ring-tOp-closed[of K], assumption+)
apply (simp add:Kxa-def, erule bxE, simp)
apply (subst Ring.ring-tOp-commute, assumption+)
apply (rule npowf-mem, assumption+) apply (simp add:Vr-mem-f-mem)
apply (frule-tac n = tna (v x) + (1 + int n) in npowf-mem[of t ],
         assumption,
         frule-tac n = - tna (v x) + (-1 - int n) in npowf-mem[of t ],
         assumption,
         frule-tac x = k in Vr-mem-f-mem[of v], assumption+)
apply (simp add:Ring.ring-tOp-assoc npowf-exp-add[THEN sym, of t])

```

```

apply (simp add:field-npowf-exp-zero)
apply (simp add:Ring.ring-r-one)

apply (frule pj-Hom[of Vr K v vp K v], assumption+)
apply (frule-tac a = k in rHom-mem[of pj (Vr K v) (vp K v) Vr K v
    Vr K v /r vp K v], assumption+)
apply (frule-tac x = pj (Vr K v) (vp K v) k in Ring.csrf-fn-mem[of Vr K v
    vp K v], assumption+)
apply (simp add:Vr-mem-f-mem)
apply (rule npowf-mem, assumption+)

apply (simp add:Kxa-def) apply (erule bxE, simp)
apply (frule-tac x = k in Vr-mem-f-mem[of v], assumption+)
apply (frule-tac n = tna (v x) + (1 + int n) in npowf-mem[of t],
    assumption+)
apply (frule-tac n = - tna (v x) + (- 1 - int n) in npowf-mem[of t],
    assumption+)
apply (frule-tac x = tK(tna (v x) + (1 + int n)) and y = k in
    Ring.ring-tOp-commute[of K], assumption+) apply simp
apply (simp add:Ring.ring-tOp-assoc, simp add:npowf-exp-add[THEN sym])
apply (simp add:field-npowf-exp-zero)
apply (simp add:Ring.ring-r-one)
apply (thin-tac (tK(tna (v x) + (1 + int n))) ·r k =
    k ·r (tK(tna (v x) + (1 + int n))))
apply (frule pj-Hom[of Vr K v vp K v], assumption+)
apply (frule-tac a = k in rHom-mem[of pj (Vr K v) (vp K v) Vr K v
    Vr K v /r vp K v], assumption+)
apply (frule-tac x = pj (Vr K v) (vp K v) k in Ring.csrf-fn-mem[of Vr K v
    vp K v], assumption+)
apply (frule-tac x = csrf-fn (Vr K v) (vp K v) (pj (Vr K v) (vp K v) k) in
    Vr-mem-f-mem[of v], assumption+)
apply (frule-tac x = csrf-fn (Vr K v) (vp K v) (pj (Vr K v) (vp K v) k) and
    y = tK(tna (v x) + (1 + int n)) in Ring.ring-tOp-closed[of K], assumption+)
apply (simp add:aGroup.ag-p-inv[of K])
apply (frule-tac x = psum K x v t n in aGroup.ag-mOp-closed[of K],
    assumption+)
apply (frule-tac x = (csrf-fn (Vr K v) (vp K v)(pj (Vr K v) (vp K v) k)) ·r
    (tK(tna (v x) + (1 + int n))) in aGroup.ag-mOp-closed[of K], assumption+)
apply (subst aGroup.ag-pOp-assoc[THEN sym], assumption+)
apply simp
apply (simp add:Ring.ring-inv1-1)
apply (subst Ring.ring-distrib2[THEN sym, of K], assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+)
apply (frule-tac x = k in Ring.csrf-diff-in-vpr[of Vr K v vp K v]
    , assumption+)
apply (frule Ring.ring-is-ag[of Vr K v])
apply (frule-tac x = csrf-fn (Vr K v) (vp K v) (pj (Vr K v) (vp K v) k) in
    aGroup.ag-mOp-closed[of Vr K v], assumption+)

```

```

apply (simp add: Vr-pOp-f-pOp) apply (simp add: Vr-mOp-f-mOp)
apply (frule-tac  $x = k \pm -_a$  (csrp-fn (Vr K v) (vp K v) (pj (Vr K v) (vp K v) k)) in vp-mem-times-t[of v t], assumption+, erule bxE, simp)
apply (frule-tac  $x = a$  in Vr-mem-f-mem[of v], assumption+)
apply (simp add:Ring.ring-tOp-assoc[of K])
apply (simp add:npowf-exp-1-add[THEN sym, of t])
apply (simp add:add.commute[of 2])
apply (simp add:add.assoc)
apply (subst Ring.ring-tOp-commute, assumption+)
apply (rule npowf-mem, assumption+) apply blast
done

lemma Suc-diff-int:  $0 < n \implies \text{int}(n - \text{Suc } 0) = \text{int } n - 1$ 
by (cut-tac of-nat-Suc[of n - Suc 0], simp)

lemma (in Corps) ecf-mem:[valuation K v;  $t \in \text{carrier } K$ ;  $v t = 1$ ;
 $x \in \text{carrier } K$ ;  $x \neq \mathbf{0}$ ]  $\implies \text{ecf}_K v t x n \in \text{carrier } K$ 
apply (frule val1-neq-0[of v t], assumption+)
apply (cut-tac field-is-ring,
       frule Vr-ring[of v], frule vp-ideal[of v])
apply (case-tac  $n = 0$ )
apply (simp add:expand-coeff-def)
apply (rule Vr-mem-f-mem[of v], assumption+)
apply (rule Ring.csrp-fn-mem, assumption+)
apply (subgoal-tac  $x \cdot_r (t_K^{-} \text{tna}(v x)) \in \text{carrier } (\text{Vr } K v)$ )
apply (simp add:pj-mem Ring.qring-carrier, blast)
apply (frule npowf-mem[of t - tna(v x)], assumption+,
       subst val-pos-mem-Vr[THEN sym, of v x  $\cdot_r (t_K^{(-} \text{tna}(v x)))$ ],
       assumption+,
       rule Ring.ring-tOp-closed, assumption+)
apply (simp add:val-t2p,
       simp add:val-exp[THEN sym, of v t - tna(v x)])
apply (frule value-in-aug-inf[of v x], assumption+,
       simp add:aug-inf-def)
apply (frule val-nonzero-noninf[of v x], assumption+)
apply (simp add:aminus[THEN sym], simp add:ant-tna)
apply (simp add:aadd-minus-r)

apply (simp add:expand-coeff-def)
apply (frule psum-diff-mem-Kxa[of v t x n - 1],
       assumption+, erule conjE)
apply (simp add:Kxa-def, erule bxE,
       frule-tac  $x = k$  in Vr-mem-f-mem[of v], assumption+,
       frule npowf-mem[of t tna(v x) + (1 + int(n - Suc 0))],
       assumption+,
       frule npowf-mem[of t - tna(v x) - int n], assumption+)
apply simp
apply (thin-tac  $x \pm -_a$  psum K x v t (n - Suc 0) =
           (t_K(tna(v x) + (1 + int(n - Suc 0))))  $\cdot_r k$ )

```

```

apply(simp add:Ring.ring-tOp-commute[of K t_K^(tna (v x) + (1 + int (n - Suc 0)))])
apply (simp add:Ring.ring-tOp-assoc, simp add:npowf-exp-add[THEN sym])
apply (thin-tac t_K^(tna (v x) + (1 + int (n - Suc 0))) ∈ carrier K,
       thin-tac t_K^(- tna (v x) - int n) ∈ carrier K)
apply (simp add:Suc-diff-int[of n])
apply (simp add:npowf-def, simp add:Ring.ring-r-one)
apply (rule Vr-mem-f-mem, assumption+)
apply (rule Ring.csrp-fn-mem, assumption+)
apply (simp add:pj-mem Ring.qring-carrier, blast)
done

lemma (in Corps) etm-mem:[valuation K v; t ∈ carrier K; v t = 1;
  x ∈ carrier K; x ≠ 0] ⇒ etm_K v t x n ∈ carrier K
apply (frule val1-neq-0[of v t], assumption+)
apply (simp add:expand-term-def)
apply (cut-tac field-is-ring,
       rule Ring.ring-tOp-closed[of K], assumption)
apply (simp add:ecf-mem)
apply (simp add:npowf-mem)
done

lemma (in Corps) psum-sum-etm:[valuation K v; t ∈ carrier K; v t = 1;
  x ∈ carrier K; x ≠ 0] ⇒
  psum_K x v t n = nsum K (λj. (ecf_K v t x j) · r (t_K^(tna (v x) + (int j)))) n
apply (frule val1-neq-0[of v t], assumption+)
apply (induct-tac n)
apply (simp add:expand-coeff-def)
apply (rotate-tac -1, drule sym)
apply simp
apply (thin-tac Σ_e K (λj. ecf_K v t x j) · r t_K^(tna (v x) + int j)) n =
  psum_K x v t n
apply (simp add:expand-coeff-def)
done

lemma zabs-pos:0 ≤ (abs (z::int))
by (simp add:zabs-def)

lemma abs-p-self-pos:0 ≤ z + (abs (z::int))
by (simp add:zabs-def)

lemma zadd-right-mono:(i::int) ≤ j ⇒ i + k ≤ j + k
by simp

theorem (in Corps) expansion-thm:[valuation K v; t ∈ carrier K;
  v t = 1; x ∈ carrier K; x ≠ 0] ⇒ lim_K v (partial-sum K x v t) x
apply (cut-tac field-is-ring, frule Ring.ring-is-ag[of K])
apply (simp add:limit-def)
apply (rule allI)
apply (subgoal-tac ∀ n. (N + na (Abs (v x))) < n →

```

```

 $psum_K x v t n \pm -_a x \in vp K v (Vr K v) (an N)$ 
apply blast
apply (rule allI, rule impI)
apply (frule-tac n = n in psum-diff-mem-Kxa[of v t x], assumption+, erule conjE)
apply (frule-tac j = tna (v x) + (1 + int n) in Kxa-val-ge[of v t], assumption+)
apply simp
apply (thin-tac Kxa K v (t_K(tna (v x) + (1 + int n))) = {xa ∈ carrier K. ant (tna (v x) + (1 + int n)) ≤ v xa})
apply (erule conjE)
apply (simp add:a-zpz[THEN sym])

apply (subgoal-tac (an N) ≤ v (psum_K x v t n ± -_a x))

apply (frule value-in-aug-inf[of v x], assumption+, simp add:aug-inf-def)
apply (frule val-nonzero-noninf[of v x], assumption+)
apply (simp add:ant-tna)
apply (frule val-surj-n-val[of v], blast)
apply (rule-tac x = psum_K x v t n ± -_a x and n = an N in n-value-x-2[of v], assumption+)
apply (subst val-pos-mem-Vr[THEN sym, of v], assumption+)
apply (rule aGroup.ag-pOp-closed[of K], assumption+)
apply (simp add:aGroup.ag-mOp-closed)

apply (cut-tac n = N in an-nat-pos)
apply (rule-tac i = 0 and j = an N and k = v (psum_K x v t n ± -_a x) in ale-trans, assumption+)
apply simp
apply simp

apply (frule-tac x1 = x ± (-_a psum_K x v t n) in val-minus-eq[THEN sym, of v], assumption+, simp, thin-tac v (x ± -_a psum_K x v t n) = v (-_a (x ± -_a psum_K x v t n)))
apply (frule-tac x = psum_K x v t n in aGroup.ag-mOp-closed[of K], assumption+, simp add:aGroup.ag-p-inv, simp add:aGroup.ag-inv-inv)
apply (frule aGroup.ag-mOp-closed[of K x], assumption+)
apply (simp add:aGroup.ag-pOp-commute[of K -_a x])

apply (cut-tac Abs-pos[of v x])
apply (frule val-nonzero-z[of v x], assumption+, erule exE, simp)
apply (simp add:na-def)
apply (cut-tac aneg-less[THEN sym, of 0 Abs (v x)], simp)
apply (frule val-nonzero-noninf[of v x], assumption+)
apply (simp add:tna-ant)
apply (simp only:ant-1[THEN sym], simp del:ant-1 add:a-zpz)
apply (simp add:add.assoc[THEN sym])
apply (cut-tac m1 = N + nat |z| and n1 = n in of-nat-less-iff [where ?'a =
```

```

int, THEN sym], simp)
apply (frule-tac a = int N + abs z and b = int n and c = z + 1 in
       add-strict-right-mono, simp only:add.commute)
apply (simp only:add.assoc[of - 1])
apply (simp only:add.assoc[THEN sym, of 1])
apply (simp only:add.commute[of 1])
apply (simp only:add.assoc[of - 1])
apply (cut-tac ?a1 = z and ?b1 = abs z and ?c1 = 1 + int N in
       add.assoc[THEN sym])
apply (thin-tac |z| + int N < int n)
apply (frule-tac a = z + (|z| + (1 + int N)) and b = z + |z| + (1 + int N) and
       c = int n + (z + 1) in ineq-conv1, assumption+)
apply (thin-tac z + (|z| + (1 + int N)) < int n + (z + 1),
       thin-tac z + (|z| + (1 + int N)) = z + |z| + (1 + int N),
       thin-tac N + nat |z| < n)
apply (cut-tac z = z in abs-p-self-pos)
apply (frule-tac i = 0 and j = z + abs z and k = 1 + int N in
       zadd-right-mono)
apply (simp only:add-0)
apply (frule-tac i = 1 + int N and j = z + |z| + (1 + int N) and
       k = int n + (z + 1) in zle-zless-trans, assumption+)
apply (thin-tac z + |z| + (1 + int N) < int n + (z + 1),
       thin-tac 0 ≤ z + |z|,
       thin-tac 1 + int N ≤ z + |z| + (1 + int N))
apply (cut-tac m1 = 1 + int N and n1 = int n + (z + 1) in
       aless-zless[THEN sym], simp)

apply (frule-tac x = ant (1 + int N) and y = ant (int n + (z + 1))
       and z = v ( psum K x v t n ± -a x) in aless-le-trans, assumption+)
apply (thin-tac ant (1 + int N) < ant (int n + (z + 1)))

apply (simp add:a-zpz[THEN sym])
apply (frule-tac x = 1 + ant (int N) and y = v ( psum K x v t n ± -a x) in
       aless-imp-le, thin-tac 1 + ant (int N) < v ( psum K x v t n ± -a x))
apply (cut-tac a0-less-1, frule aless-imp-le[of 0 1])
apply (frule-tac b = ant (int N) in aadd-pos-le[of 1])
apply (subst an-def)
apply (rule-tac i = ant (int N) and j = 1 + ant (int N) and
       k = v ( psum K x v t n ± -a x) in ale-trans, assumption+)
done

```

2.9.1 Hensel's theorem

definition

```

pol-Cauchy-seq :: [('b, 'm) Ring-scheme, 'b, -, 'b ⇒ ant,
                    nat ⇒ 'b] ⇒ bool ((5PCauchy - - - -) [90,90,90,90,91]90) where
  PCauchy R X K v F ←→ (∀ n. (F n) ∈ carrier R) ∧
    (∃ d. (∀ n. deg R (Vr K v) X (F n) ≤ (an d))) ∧

```

$$(\forall N. \exists M. (\forall n m. M < n \wedge M < m \longrightarrow P\text{-mod } R (Vr K v) X ((vp K v)^{(Vr K v)} (an N)) (F n \pm_R -_a R (F m))))$$

definition

$$\begin{aligned} pol-limit :: [(&'b, 'm) Ring-scheme, 'b, -, 'b \Rightarrow ant, \\ nat \Rightarrow 'b, 'b] \Rightarrow bool \\ ((6Plimit \dots \dots \dots) [90, 90, 90, 90, 90, 91] 90) \text{ where} \\ Plimit_R X K v F p \longleftrightarrow (\forall n. (F n) \in carrier R) \wedge \\ (\forall N. \exists M. (\forall m. M < m \longrightarrow \\ P\text{-mod } R (Vr K v) X ((vp K v)^{(Vr K v)} (an N)) ((F m) \pm_R -_a R p))) \end{aligned}$$

definition

$$\begin{aligned} Pseql :: [(&'b, 'm) Ring-scheme, 'b, -, 'b \Rightarrow ant, nat, \\ nat \Rightarrow 'b] \Rightarrow nat \Rightarrow 'b \\ ((6Pseql \dots \dots \dots) [90, 90, 90, 90, 90, 91] 90) \text{ where} \\ Pseql_R X K v d F = (\lambda n. (ldeg-p R (Vr K v) X d (F n))) \end{aligned}$$

definition

$$\begin{aligned} Pseqh :: [(&'b, 'm) Ring-scheme, 'b, -, 'b \Rightarrow ant, nat, nat \Rightarrow 'b] \Rightarrow \\ nat \Rightarrow 'b \\ ((6Pseqh \dots \dots \dots) [90, 90, 90, 90, 90, 91] 90) \text{ where} \\ Pseqh_R X K v d F = (\lambda n. (hdeg-p R (Vr K v) X (Suc d) (F n))) \end{aligned}$$

lemma *an-neq-minf*[simp]: $\forall n. -\infty \neq an n$

apply (*rule allI*)
apply (*simp add:an-def*) **apply** (*rule not-sym*) **apply** *simp*
done

lemma *an-neq-minf1*[simp]: $\forall n. an n \neq -\infty$

apply (*rule allI*) **apply** (*simp add:an-def*)
done

lemma (in Corps) *Pseql-mem*: $\llbracket \text{valuation } K v; \text{PolynRg } R (Vr K v) X;$

$F n \in \text{carrier } R; \forall n. \text{deg } R (Vr K v) X (F n) \leq an (\text{Suc } d) \rrbracket \implies$
 $(Pseql_R X K v d F) n \in \text{carrier } R$

apply (*frule PolynRg.is-Ring*)
apply (*simp add:Pseql-def*)
apply (*frule Vr-ring[of v]*,
rule PolynRg.ldeg-p-mem, assumption+, simp)
done

lemma (in Corps) *Pseqh-mem*: $\llbracket \text{valuation } K v; \text{PolynRg } R (Vr K v) X;$

$F n \in \text{carrier } R; \forall n. \text{deg } R (Vr K v) X (F n) \leq an (\text{Suc } d) \rrbracket \implies$
 $(Pseqh_R X K v d F) n \in \text{carrier } R$

apply (*frule PolynRg.is-Ring*)

```

apply (frule Vr-ring[of v])
apply (frule PolynRg.subring[of R Vr K v X])
apply (frule PolynRg.X-mem-R[of R Vr K v X])
apply (simp del:npow-suc add:Pseqh-def)
apply (rule PolynRg.hdeg-p-mem, assumption+, simp)
done

lemma (in Corps) PCauchy-lTr:[valuation K v; PolynRg R (Vr K v) X;
  p ∈ carrier R; deg R (Vr K v) X p ≤ an (Suc d);
  P-mod R (Vr K v) X (vp K v(Vr K v) (an N)) p] ==>
  P-mod R (Vr K v) X (vp K v(Vr K v) (an N)) (ldeg-p R (Vr K v) X d p)
apply (frule PolynRg.is-Ring)
apply (simp add:ldeg-p-def)
apply (frule Vr-ring[of v])
apply (frule PolynRg.scf-d-pol[of R Vr K v X p Suc d], assumption+,
  (erule conjE)+)
apply (frule-tac n = an N in vp-apow-ideal[of v], simp)
apply (frule PolynRg.P-mod-mod[THEN sym, of R Vr K v X vp K v (Vr K v) (an N)
  p scf-d R (Vr K v) X p (Suc d)], assumption+, simp)
apply (subst PolynRg.polyn-expr-short[of R Vr K v X
  scf-d R (Vr K v) X p (Suc d) d], assumption+, simp)
apply (subst PolynRg.P-mod-mod[THEN sym, of R Vr K v X vp K v (Vr K v) (an N)
  polyn-expr R X d (d, snd (scf-d R (Vr K v) X p (Suc d)))
  (d, snd (scf-d R (Vr K v) X p (Suc d)))], assumption+)
apply (subst PolynRg.polyn-expr-short[THEN sym], simp+,
  simp add:PolynRg.polyn-mem)
apply (subst pol-coeff-def, rule allI, rule impI,
  simp add:PolynRg.pol-coeff-mem)
apply simp+
done

lemma (in Corps) PCauchy-hTr:[valuation K v; PolynRg R (Vr K v) X;
  p ∈ carrier R; deg R (Vr K v) X p ≤ an (Suc d);
  P-mod R (Vr K v) X (vp K v(Vr K v) (an N)) p]
  ==> P-mod R (Vr K v) X (vp K v(Vr K v) (an N)) (hdeg-p R (Vr K v) X (Suc d) p)
apply (frule PolynRg.is-Ring)
apply (cut-tac Vr-ring[of v])
apply (frule PolynRg.scf-d-pol[of R Vr K v X p Suc d], assumption+)
apply (frule-tac n = an N in vp-apow-ideal[of v], simp)
apply (frule PolynRg.P-mod-mod[THEN sym, of R Vr K v X
  vp K v (Vr K v) (an N) p scf-d R (Vr K v) X p (Suc d)], assumption+, simp+)
apply (subst hdeg-p-def)
apply (subst PolynRg.monomial-P-mod-mod[THEN sym, of R Vr K v X
  vp K v (Vr K v) (an N) snd (scf-d R (Vr K v) X p (Suc d)) (Suc d)
  (snd (scf-d R (Vr K v) X p (Suc d)) (Suc d)) ·_R (X^R (Suc d))
  Suc d],

```

```

assumption+)
apply (rule PolynRg.pol-coeff-mem[of R Vr K v X
scf-d R (Vr K v) X p (Suc d) Suc d], assumption+, simp+)
done

lemma (in Corps) v-ldeg-p-pOp:[valuation K v; PolynRg R (Vr K v) X;
p ∈ carrier R; q ∈ carrier R; deg R (Vr K v) X p ≤ an (Suc d);
deg R (Vr K v) X q ≤ an (Suc d)] ==>
(ldeg-p R (Vr K v) X d p) ±R (ldeg-p R (Vr K v) X d q) =
ldeg-p R (Vr K v) X d (p ±R q)
by (simp add:PolynRg.ldeg-p-pOp[of R Vr K v X p q d])

lemma (in Corps) v-hdeg-p-pOp:[valuation K v; PolynRg R (Vr K v) X;
p ∈ carrier R; q ∈ carrier R; deg R (Vr K v) X p ≤ an (Suc d);
deg R (Vr K v) X q ≤ an (Suc d)] ==> (hdeg-p R (Vr K v) X (Suc d) p) ±R
(hdeg-p R (Vr K v) X (Suc d) q) = hdeg-p R (Vr K v) X (Suc d) (p ±R q)
by (simp add:PolynRg.hdeg-p-pOp[of R Vr K v X p q d])

lemma (in Corps) v-ldeg-p-mOp:[valuation K v; PolynRg R (Vr K v) X;
p ∈ carrier R; deg R (Vr K v) X p ≤ an (Suc d)] ==>
¬aR (ldeg-p R (Vr K v) X d p) = ldeg-p R (Vr K v) X d (¬aR p)
by (simp add:PolynRg.ldeg-p-mOp)

lemma (in Corps) v-hdeg-p-mOp:[valuation K v; PolynRg R (Vr K v) X;
p ∈ carrier R; deg R (Vr K v) X p ≤ an (Suc d)] ==>
¬aR (hdeg-p R (Vr K v) X (Suc d) p) = hdeg-p R (Vr K v) X (Suc d) (¬aR p)
by (simp add:PolynRg.hdeg-p-mOp)

lemma (in Corps) PCauchy-lPCauchy:[valuation K v; PolynRg R (Vr K v) X;
∀ n. F n ∈ carrier R; ∀ n. deg R (Vr K v) X (F n) ≤ an (Suc d);
P-mod R (Vr K v) X (vp K v (Vr K v) (an N)) (F n ±R ¬aR (F m))] ==>
P-mod R (Vr K v) X (vp K v (Vr K v) (an N))
(((Pseq R X K v d F) n) ±R ¬aR ((Pseq R X K v d F) m))
apply (frule PolynRg.is-Ring)
apply (cut-tac Vr-ring[of v],
frule Ring.ring-is-ag[of R],
frule PolynRg.subring[of R Vr K v X])
apply (simp add:Pseq-def)
apply (subst v-ldeg-p-mOp[of v R X - d], assumption+,
simp, simp)
apply (subst v-ldeg-p-pOp[of v R X F n -aR (F m)], assumption+,
simp, rule aGroup.ag-mOp-closed, assumption, simp, simp,
simp add:PolynRg.deg-minus-eq1)
apply (rule PCauchy-lTr[of v R X F n ±R ¬aR (F m) d N],
assumption+,
rule aGroup.ag-pOp-closed[of R F n -aR (F m)], assumption+,
simp, rule aGroup.ag-mOp-closed, assumption+, simp)
apply (frule PolynRg.deg-minus-eq1 [of R Vr K v X F m], simp)

```

```

apply (rule PolynRg.polyn-deg-add4[of R Vr K v X F n
   $\neg_{aR} (F m) \text{ Suc } d], assumption+, simp,$ 
  rule aGroup.ag-mOp-closed, assumption, simp+)
done

lemma (in Corps) PCauchy-hPCauchy:[valuation K v; PolynRg R (Vr K v) X;
   $\forall n. F n \in \text{carrier } R; \forall n. \deg R (Vr K v) X (F n) \leq an (\text{Suc } d);$ 
  P-mod R (Vr K v) X (vp K v (Vr K v) (an N)) (F n  $\pm_R \neg_{aR} (F m)) \Rightarrow$ 
   $\Rightarrow P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
   $((Pseqh_R X K v d F) n) \pm_R \neg_{aR} ((Pseqh_R X K v d F) m))$ 
apply (frule PolynRg.is-Ring)
apply (frule Vr-ring[of v], frule Ring.ring-is-ag[of R],
  frule PolynRg.subring[of R Vr K v X],
  frule vp-apow-ideal[of v an N], simp)

apply (simp add:Pseqh-def,
  subst v-hdeg-p-mOp[of v R X F m d], assumption+,
  simp+)

apply (subst v-hdeg-p-pOp[of v R X F n  $\neg_{aR} (F m)], assumption+,$ 
  simp, rule aGroup.ag-mOp-closed, assumption, simp, simp,
  frule PolynRg.deg-minus-eq1 [of R Vr K v X F m],
  simp+ )
apply (frule PCauchy-hTr[of v R X F n  $\pm_R \neg_{aR} (F m) d N],$ 
  assumption+,
  rule aGroup.ag-pOp-closed[of R F n  $\neg_{aR} (F m)], assumption+,$ 
  simp, rule aGroup.ag-mOp-closed, assumption+, simp)
apply (frule PolynRg.deg-minus-eq1 [of R Vr K v X F m], simp+)
apply (rule PolynRg.polyn-deg-add4[of R Vr K v X F n  $\neg_{aR} (F m)$ 
  Suc d], assumption+,
  simp, rule aGroup.ag-mOp-closed, assumption, simp+)
done

```

```

lemma (in Corps) Pseq-decompos:[valuation K v; PolynRg R (Vr K v) X;
   $F n \in \text{carrier } R; \deg R (Vr K v) X (F n) \leq an (\text{Suc } d) \Rightarrow$ 
   $\Rightarrow F n = ((Pseql_R X K v d F) n) \pm_R ((Pseqh_R X K v d F) n)$ 
apply (frule PolynRg.is-Ring)
apply (simp del:npow-suc add:Pseql-def Pseqh-def)
apply (frule Vr-ring[of v])
apply (frule PolynRg.subring[of R Vr K v X])
apply (rule PolynRg.decompos-p[of R Vr K v X F n d], assumption+)
done

```

```

lemma (in Corps) deg-0-const:[valuation K v; PolynRg R (Vr K v) X;
   $p \in \text{carrier } R; \deg R (Vr K v) X p \leq 0] \Rightarrow p \in \text{carrier } (Vr K v)$ 
apply (frule Vr-ring[of v])
apply (frule PolynRg.subring)

```

```

apply (frule PolynRg.is-Ring)
apply (case-tac p = 0R, simp,
      simp add:Ring.Subring-zero-ring-zero[THEN sym],
      simp add:Ring.ring-zero)
apply (subst PolynRg.pol-of-deg0[THEN sym, of R Vr K v X p],
      assumption+)
apply (simp add:PolynRg.deg-an, simp only:an-0[THEN sym])
apply (simp only:ale-nat-le[of deg-n R (Vr K v) X p 0])
done

lemma (in Corps) monomial-P-limt:[valuation K v; Completev K;
  PolynRg R (Vr K v) X; ∀ n. f n ∈ carrier (Vr K v);
  ∀ n. F n = (f n) ·rR (X^R d); ∀ N. ∃ M. ∀ n m. M < n ∧ M < m →
  P-mod R (Vr K v) X (vp K v (Vr K v) (an N)) (F n ±R −aR (F m))] ==>
  ∃ b ∈ carrier (Vr K v). Plimit R X K v F (b ·rR (X^R d))
apply (frule PolynRg.is-Ring)
apply (frule Vr-ring[of v])
apply (frule PolynRg.subring[of R Vr K v X])
apply simp

apply (subgoal-tac Cauchy K v f)
apply (simp add:v-complete-def)
apply (drule-tac a = f in forall-spec)
apply (thin-tac ∀ N. ∃ M. ∀ n m. M < n ∧ M < m →
  P-mod R (Vr K v) X (vp K v (Vr K v) (an N))
  ((f n) ·rR (X^R d) ±R −aR (f m) ·rR (X^R d)), assumption)
apply (erule exE, erule conjE)
apply (subgoal-tac b ∈ carrier (Vr K v))
apply (subgoal-tac Plimit R X K v F (b ·rR (X^R d)), blast)

apply (simp add:pol-limit-def)
apply (rule conjI)
apply (rule allI)
apply (rule Ring.ring-tOp-closed[of R], assumption)
apply (frule PolynRg.subring[of R Vr K v X])
apply (rule Ring.mem-subring-mem-ring[of R Vr K v], assumption+)
apply simp

apply (frule PolynRg.X-mem-R[of R Vr K v X])
apply (simp add:Ring.npClose)
apply (thin-tac ∀ n. F n = f n ·rR X^R d)
apply (simp add:limit-def)
apply (rule allI)

apply (rotate-tac −2, drule-tac x = N in spec)
apply (erule exE)
apply (subgoal-tac ∀ n > M. P-mod R (Vr K v) X (vp K v (Vr K v) (an N)))

```

```

 $((f n) \cdot_r R (X^{\sim R} d) \pm_R -_a R (b \cdot_r R (X^{\sim R} d))), \text{blast}$ 
apply (rule allI, rule impI)
apply (rotate-tac -2, drule-tac x = n in spec, simp)
apply (drule-tac x = n in spec)

apply (frule-tac x = f n in Ring.mem-subring-mem-ring[of R Vr K v], assumption+, frule-tac x = b in Ring.mem-subring-mem-ring[of R Vr K v], assumption+)
apply (frule PolynRg.X-mem-R[of R Vr K v X])
apply (frule Ring.npClose[of R X d], assumption+)
apply (simp add:Ring.ring-inv1-1)
apply (frule Ring.ring-is-ag[of R], frule-tac x = b in aGroup.ag-mOp-closed[of R], assumption+)
apply (subst Ring.ring-distrib2[THEN sym, of R X^{\sim R} d], assumption+)

apply (frule-tac n = an N in vp-apow-ideal[of v], simp)
apply (frule-tac I = vp K v (Vr K v) (an N) and c = f n \pm_R -_a R b and p = (f n \pm_R -_a R b) \cdot_r R (X^{\sim R} d) in PolynRg.monomial-P-mod-mod[of R Vr K v X - - - d], assumption+)
apply (simp add:Ring.Subring-minus-ring-minus[THEN sym])
apply (frule Ring.ring-is-ag[of Vr K v])
apply (frule-tac x = b in aGroup.ag-mOp-closed[of Vr K v], assumption+)
apply (simp only:Ring.Subring-pOp-ring-pOp[THEN sym])
apply (rule aGroup.ag-pOp-closed, assumption+ apply simp)
apply (frule-tac I1 = vp K v (Vr K v) (an N) and c1 = f n \pm_R -_a R b and p1 = (f n \pm_R -_a R b) \cdot_r R (X^{\sim R} d) in PolynRg.monomial-P-mod-mod[THEN sym, of R Vr K v X - - - d], assumption+)
apply (frule Ring.ring-is-ag[of Vr K v])
apply (frule-tac x = b in aGroup.ag-mOp-closed[of Vr K v], assumption+)
apply (simp only:Ring.Subring-minus-ring-minus[THEN sym])
apply (simp only:Ring.Subring-pOp-ring-pOp[THEN sym])
apply (rule aGroup.ag-pOp-closed, assumption+ apply simp apply simp)
apply (simp only:Vr-mOp-f-mOp[THEN sym])
apply (frule Ring.ring-is-ag[of Vr K v])
apply (frule-tac x = b in aGroup.ag-mOp-closed[of Vr K v], assumption+)
apply (simp add:Vr-pOp-f-pOp[THEN sym])
apply (simp add:Ring.Subring-pOp-ring-pOp)
apply (simp add:Ring.Subring-minus-ring-minus)

apply (case-tac b = 0_K, simp add:Vr-0-f-0[THEN sym], simp add:Ring.ring-zero)
apply (frule-tac b = b in limit-val[of - f v], assumption+, rule allI, frule-tac x = j in spec, simp add:Vr-mem-f-mem, assumption+, erule exE)
apply (thin-tac \forall n. F n = f n \cdot_r R X^{\sim R} d,

```

$\text{thin-tac } \forall N. \exists M. \forall n m. M < n \wedge M < m \rightarrow$
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$
 $(f n \cdot_r R X^{\sim R d} \pm_R -_a R f m \cdot_r R X^{\sim R d}))$
apply (*rotate-tac* –1, *drule-tac* $x = Suc N$ **in** *spec*, *simp*)
apply (*drule-tac* $x = Suc N$ **in** *spec*)
apply (*frule-tac* $x1 = f (Suc N)$ **in** *val-pos-mem*-*Vr*[*THEN sym, of v*],
simp add:*Vr-mem-f-mem*, *simp*, *simp add*:*val-pos-mem*-*Vr*[*of v*])

apply (*simp add*:*Cauchy-seq-def*)
apply (*simp add*:*Vr-mem-f-mem*)
apply (*rule allI*)
apply (*rotate-tac* –3, *frule-tac* $x = N$ **in** *spec*)

apply (*thin-tac* $\forall n. F n = f n \cdot_r R X^{\sim R d}$)

apply (*frule-tac* $n = an N$ **in** *vp-apow-ideal*[*of v*], *simp*)
apply (*drule-tac* $x = N$ **in** *spec*, *erule exE*)
apply (*subgoal-tac* $\forall n m. M < n \wedge M < m \rightarrow$
 $f n \pm -_a (f m) \in vp K v (Vr K v) (an N)$, *blast*)
apply (*rule allI*)+
apply (*rule impI*, *erule conjE*)
apply (*frule-tac* $I = vp K v (Vr K v) (an N)$ **and** $c = f n \pm -_a (f m)$ **and**
 $p = (f n \pm -_a (f m)) \cdot_r R (X^{\sim R d})$ **in**
PolynRg.monomial-P-mod-mod[*of R Vr K v X --- d*], *assumption+*)

apply (*frule-tac* $x = n$ **in** *spec*,
drule-tac $x = m$ **in** *spec*)
apply (*frule Ring.ring-is-ag*[*of Vr K v*],
simp add:*Vr-mOp-f-mOp*[*THEN sym*],
frule-tac $x = f m$ **in** *aGroup.ag-mOp-closed*[*of Vr K v*], *assumption+*,
simp add:*Vr-pOp-f-pOp*[*THEN sym*])
apply (*rule aGroup.ag-pOp-closed*, *assumption+*, *simp*)
apply *simp*
apply (*thin-tac* $(f n \pm -_a f m \in vp K v (Vr K v) (an N)) =$
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N)) ((f n \pm -_a f m) \cdot_r R X^{\sim R d}))$
apply (*rotate-tac* –3, *drule-tac* $x = n$ **in** *spec*,
rotate-tac –1, *drule-tac* $x = m$ **in** *spec*, *simp*)
apply (*frule-tac* $x = n$ **in** *spec*,
drule-tac $x = m$ **in** *spec*)
apply (*frule-tac* $x = f n$ **in** *Ring.mem-subring-mem-ring*[*of R Vr K v*],
assumption+,
frule-tac $x = f m$ **in** *Ring.mem-subring-mem-ring*[*of R Vr K v*],
assumption+,
frule Ring.ring-is-ag[*of R*],
frule-tac $x = f m$ **in** *aGroup.ag-mOp-closed*[*of R*], *assumption+*,
frule PolynRg.X-mem-R[*of R Vr K v X*],
frule Ring.npClose[*of R X d*], *assumption*)
apply (*simp add*:*Ring.ring-inv1-1*[*of R*],

```

frule-tac  $y1 = f n \text{ and } z1 = -_a R f m$  in Ring.ring-distrib2[
  THEN sym, of  $R X^{\sim R} d$ ], assumption+, simp,
thin-tac  $f n \cdot_r R X^{\sim R} d \pm_R (-_a R f m) \cdot_r R X^{\sim R} d =$ 
           $(f n \pm_R -_a R f m) \cdot_r R X^{\sim R} d$ )
apply (simp only:Ring.Subring-minus-ring-minus[THEN sym,of  $R Vr K v$ ])
apply (frule Ring.subring-Ring[of  $R Vr K v$ ], assumption,
       frule Ring.ring-is-ag[of  $Vr K v$ ],
       frule-tac  $x = f m$  in aGroup.ag-mOp-closed[of  $Vr K v$ ], assumption+)
apply (simp add:Ring.Subring-pOp-ring-pOp[THEN sym, of  $R Vr K v$ ],
       simp add:Vr-pOp-f-pOp, simp add:Vr-mOp-f-mOp)
done

lemma (in Corps) mPlimit-uniqueTr:[valuation  $K v$ ;
  PolynRg  $R (Vr K v) X; \forall n. f n \in carrier (Vr K v)$ ;
   $\forall n. F n = (f n) \cdot_r R (X^{\sim R} d); c \in carrier (Vr K v)$ ;
   $Plimit R X K v F (c \cdot_r R (X^{\sim R} d)) \Rightarrow lim K v f c$ 
apply (frule PolynRg.is-Ring,
       simp add:pol-limit-def limit-def,
       rule allI,
       erule conjE,
       rotate-tac -1, drule-tac  $x = N$  in spec,
       erule exE)
apply (subgoal-tac  $\forall n. M < n \rightarrow f n \pm -_a c \in vp K v (Vr K v) (an N)$ , blast)
apply (rule allI, rule impI,
       rotate-tac -2, drule-tac  $x = n$  in spec, simp,
       drule-tac  $x = n$  in spec,
       drule-tac  $x = n$  in spec,
       thin-tac  $\forall n. f n \cdot_r R X^{\sim R} d \in carrier R$ )
apply (frule Vr-ring[of  $v$ ],
       frule PolynRg.X-mem-R[of  $R Vr K v X$ ],
       frule Ring,npClose[of  $R X d$ ], assumption+,
       frule PolynRg.subring[of  $R Vr K v X$ ])
apply (frule-tac  $x = c$  in Ring.mem-subring-mem-ring[of  $R Vr K v$ ],
       assumption+,
       frule-tac  $x = f n$  in Ring.mem-subring-mem-ring[of  $R Vr K v$ ],
       assumption+)
apply (simp add:Ring.ring-inv1-1,
       frule Ring.ring-is-ag[of  $R$ ],
       frule aGroup.ag-mOp-closed[of  $R c$ ], assumption+)
apply (simp add:Ring.ring-distrib2[THEN sym, of  $R X^{\sim R} d - -_a R c$ ],
       simp add:Ring.Subring-minus-ring-minus[THEN sym],
       frule Ring.ring-is-ag[of  $Vr K v$ ],
       frule aGroup.ag-mOp-closed[of  $Vr K v c$ ], assumption+)
apply (simp add:Ring.Subring-pOp-ring-pOp[THEN sym],
       frule-tac  $x = f n$  in aGroup.ag-pOp-closed[of  $Vr K v -$ 
           $-a (Vr K v) c$ ], assumption+,
       frule-tac  $n = an N$  in vp-apow-ideal[of  $v$ ], simp,
       frule-tac  $I1 = vp K v (Vr K v) (an N)$  and

```

```

c1 = f n ±(Vr K v) ¬a(Vr K v) c and p1 = (f n ±(Vr K v) ¬a(Vr K v) c)
·rR (X^R d) in PolynRg.monomial-P-mod-mod[THEN sym, of R Vr K v
X - - - d], assumption+, simp, simp)
apply (simp add: Vr-pOp-f-pOp, simp add: Vr-mOp-f-mOp)
done

lemma (in Corps) mono-P-limit-unique:valuation K v;
PolynRg R (Vr K v) X; ∀ n. f n ∈ carrier (Vr K v);
∀ n. F n = (f n) ·rR (X^R d); b ∈ carrier (Vr K v); c ∈ carrier (Vr K v);
Plimit R X K v F (b ·rR (X^R d)); Plimit R X K v F (c ·rR (X^R d))] ⇒
b ·rR (X^R d) = c ·rR (X^R d)
apply (frule PolynRg.is-Ring)
apply (frule-tac mPlimit-uniqueTr[of v R X f F d b], assumption+,
frule-tac mPlimit-uniqueTr[of v R X f F d c], assumption+)
apply (frule Vr-ring[of v],
frule PolynRg.subring[of R Vr K v X],
frule Vr-mem-f-mem[of v b], assumption+,
frule Vr-mem-f-mem[of v c], assumption+,
frule limit-unique[of b f v c])
apply (rule allI, simp add: Vr-mem-f-mem, assumption+, simp)
done

lemma (in Corps) Plimit-deg:valuation K v; PolynRg R (Vr K v) X;
∀ n. F n ∈ carrier R; ∀ n. deg R (Vr K v) X (F n) ≤ (an d);
p ∈ carrier R; Plimit R X K v F p] ⇒ deg R (Vr K v) X p ≤ (an d)
apply (frule PolynRg.is-Ring, frule Vr-ring[of v])
apply (case-tac p = 0R, simp add:deg-def)
apply (rule contrapos-pp, simp+,
simp add:aneg-le,
frule PolynRg.s-cf-expr[of R Vr K v X p], assumption+, (erule conjE)+,
frule PolynRg.s-cf-deg[of R Vr K v X p], assumption+,
frule PolynRg.pol-coeff-mem[of R Vr K v X s-cf R (Vr K v) X p
fst (s-cf R (Vr K v) X p)], assumption+, simp,
frule Vr-mem-f-mem[of v snd (s-cf R (Vr K v) X p)
(fst (s-cf R (Vr K v) X p))], assumption+)

apply (frule val-nonzero-noninf[of v
snd (s-cf R (Vr K v) X p) (fst (s-cf R (Vr K v) X p))], assumption,
simp add: Vr-0-f-0,
frule val-pos-mem-Vr[THEN sym, of v snd (s-cf R (Vr K v) X p)
(fst (s-cf R (Vr K v) X p))], assumption+, simp,
frule value-in-aug-inf[of v snd (s-cf R (Vr K v) X p)
(fst (s-cf R (Vr K v) X p))], assumption+,
cut-tac mem-ant[of v (snd (s-cf R (Vr K v) X p)
(fst (s-cf R (Vr K v) X p)))], simp add:aug-inf-def,
erule exE, simp, simp only:ant-0[THEN sym], simp only:ale-zle,
frule-tac z = z in zpos-nat, erule exE, simp,
thin-tac z = int n)

```

```

apply (simp add:pol-limit-def)
apply (rotate-tac 5, drule-tac x = Suc n in spec)
apply (erule exE)
apply (rotate-tac -1,
         drule-tac x = Suc M in spec, simp del:npow-suc,
         drule-tac x = Suc M in spec,
         drule-tac x = Suc M in spec)

apply (frule PolynRg.polyn-minus[of R Vr K v X s-cf R (Vr K v) X p
                                fst (s-cf R (Vr K v) X p)], assumption+, simp,
frule PolynRg.minus-pol-coeff[of R Vr K v X s-cf R (Vr K v) X p],
assumption+, drule sym,
frule-tac x = deg R (Vr K v) X (F (Suc M)) in ale-less-trans[of -
                    an d deg R (Vr K v) X p], assumption+,
frule-tac p = F (Suc M) and d = deg-n R (Vr K v) X p in
PolynRg.pol-expr-edeg[of R Vr K v X], assumption+,
frule-tac x = deg R (Vr K v) X (F (Suc M)) and
y = deg R (Vr K v) X p in aless-imp-le,
subst PolynRg.deg-an[THEN sym, of R Vr K v X p], assumption+,
erule exE, (erule conjE)+,
frule-tac c = f in PolynRg.polyn-add1[of R Vr K v X -
                                (fst (s-cf R (Vr K v) X p), λj. -a Vr K v snd (s-cf R (Vr K v) X p) j)],
assumption+, simp,
thin-tac -aR p = polyn-expr R X (fst (s-cf R (Vr K v) X p))
(fst (s-cf R (Vr K v) X p), λj. -a Vr K v snd (s-cf R (Vr K v) X p) j),
thin-tac polyn-expr R X (fst (s-cf R (Vr K v) X p))
(s-cf R (Vr K v) X p) = p,
thin-tac F (Suc M) = polyn-expr R X (fst (s-cf R (Vr K v) X p)) f,
thin-tac polyn-expr R X (fst (s-cf R (Vr K v) X p)) f ±R
polyn-expr R X (fst (s-cf R (Vr K v) X p))
(fst (s-cf R (Vr K v) X p), λj. -a Vr K v snd (s-cf R (Vr K v) X p) j) =
polyn-expr R X (fst (s-cf R (Vr K v) X p)) (add-cf (Vr K v) f
(fst (s-cf R (Vr K v) X p), λj. -a Vr K v snd (s-cf R (Vr K v) X p) j))

apply (frule-tac n = an (Suc n) in vp-apow-ideal[of v], simp,
frule-tac p1 = polyn-expr R X (fst (s-cf R (Vr K v) X p))(add-cf (Vr K v) f
(fst (s-cf R (Vr K v) X p), λj. -a Vr K v snd (s-cf R (Vr K v) X p) j)) and
I1 = vp K v (Vr K v) (an (Suc n)) and c1 = add-cf (Vr K v) f (fst
(s-cf R (Vr K v) X p), λj. -a Vr K v snd (s-cf R (Vr K v) X p) j) in
PolynRg.P-mod-mod[THEN sym, of R Vr K v X], assumption+,
rule PolynRg.polyn-mem[of R Vr K v X], assumption+,
rule PolynRg.add-cf-pol-coeff[of R Vr K v X], assumption+,
simp add:PolynRg.add-cf-len,
rule PolynRg.add-cf-pol-coeff[of R Vr K v X], assumption+,
simp add:PolynRg.add-cf-len,

```

```

simp add:PolynRg.add-cf-len)
apply (drule-tac x = fst (s-cf R (Vr K v) X p) in spec, simp,
      thin-tac P-mod R (Vr K v) X (vp K v (Vr K v) (an (Suc n)))
      (polyn-expr R X (fst (s-cf R (Vr K v) X p)) (add-cf (Vr K v) f
      (fst (s-cf R (Vr K v) X p), λj. -a Vr K v snd (s-cf R (Vr K v) X p) j))),,
      simp add:add-cf-def)

apply (frule-tac p = polyn-expr R X (fst (s-cf R (Vr K v) X p)) f and
      c = f and j = fst f in PolynRg.pol-len-gt-deg[of R Vr K v X],
      assumption+, simp, drule sym, simp add:PolynRg.deg-an) apply simp
apply (rotate-tac -4, drule sym, simp)
apply (frule Ring.ring-is-ag[of Vr K v],
      frule-tac x = snd (s-cf R (Vr K v) X p) (fst f) in
      aGroup.ag-mOp-closed[of Vr K v], assumption+,
      simp add:aGroup.ag-l-zero)
apply (frule-tac I = vp K v (Vr K v) (an (Suc n)) and
      x = -a Vr K v snd (s-cf R (Vr K v) X p) (fst f) in
      Ring.ideal-inv1-closed[of Vr K v], assumption+)
apply (simp add:aGroup.ag-inv-inv)
apply (frule-tac n = an (Suc n) and x = snd (s-cf R (Vr K v) X p) (fst f)
      in n-value-x-1[of v], simp+)
apply (frule-tac x = snd (s-cf R (Vr K v) X p) (fst f) in
      n-val-le-val[of v], assumption+, simp add:ant-int)
apply (drule-tac i = an (Suc n) and
      j = n-val K v (snd (s-cf R (Vr K v) X p) (fst f)) and
      k = v (snd (s-cf R (Vr K v) X p) (fst f)) in ale-trans,
      assumption+)
apply (simp add:ant-int ale-natle)
done

lemma (in Corps) Plimit-deg1:[valuation K v; Ring R; PolynRg R (Vr K v) X;
  ∀ n. F n ∈ carrier R; ∀ n. deg R (Vr K v) X (F n) ≤ ad;
  p ∈ carrier R; Plimit R X K v F p] ⇒ deg R (Vr K v) X p ≤ ad
apply (frule Vr-ring[of v])
apply (case-tac ∀ n. F n = 0_R)
apply (frule Plimit-deg[of v R X F 0 p], assumption+,
      rule allI, simp add:deg-def, assumption+)
apply (case-tac p = 0_R, simp add:deg-def,
      frule PolynRg.nonzero-deg-pos[of R Vr K v X p], assumption+,
      simp,
      frule PolynRg.pols-const[of R Vr K v X p], assumption+,
      simp,
      frule PolynRg.pols-const[of R Vr K v X p], assumption+,
      simp add:ale-refl)
apply (subgoal-tac p = 0_R, simp)
apply (thin-tac p ≠ 0_R)
apply (rule contrapos-pp, simp+)

apply (frule n-val-valuation[of v])

```

```

apply (frule val-nonzero-z[of n-val K v p])
apply (simp add: Vr-mem-f-mem)
apply (frule PolynRg.subring[of R Vr K v X])
apply (simp only:Ring.Subring-zero-ring-zero[THEN sym, of R Vr K v])
apply (simp add: Vr-0-f-0, erule exE)
apply (frule val-pos-mem-Vr[THEN sym, of v p])
apply (simp add: Vr-mem-f-mem, simp)
apply (frule val-pos-n-val-pos[of v p])
apply (simp add: Vr-mem-f-mem, simp)
apply (simp add:ant-0[THEN sym])
apply (frule-tac z = z in zpos-nat, erule exE)
apply (unfold pol-limit-def, erule conjE)
apply (rotate-tac -1, drule-tac x = Suc n in spec)
apply (subgoal-tac  $\neg (\exists M. \forall m. M < m \longrightarrow$ 
      P-mod R (Vr K v) X (vp K v (Vr K v) (an (Suc n))) (F m  $\pm_R -a_R p)))$ )
apply blast
apply (thin-tac  $\exists M. \forall m. M < m \longrightarrow$ 
      P-mod R (Vr K v) X (vp K v (Vr K v) (an (Suc n))) (F m  $\pm_R -a_R p)))$ )
apply simp
apply (subgoal-tac M < (Suc M)  $\wedge$ 
       $\neg P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an (Suc n))) (\mathbf{0}_R \pm_R -a_R p))$ )
apply blast
apply simp
apply (frule Ring.ring-is-ag[of R])
apply (frule aGroup.ag-mOp-closed[of R p], assumption)
apply (simp add:aGroup.ag-l-zero)
apply (frule Ring.ring-is-ag[of Vr K v])
apply (frule aGroup.ag-mOp-closed[of Vr K v p], assumption)
apply (frule-tac n = an (Suc n) in vp-apow-ideal[of v], simp)
apply (frule PolynRg.subring[of R Vr K v X])
apply (simp add:Ring.Subring-minus-ring-minus[THEN sym, of R Vr K v])
apply (simp add:PolynRg.P-mod-coeffTr[of R Vr K v X - -a(Vr K v) p])
apply (rule contrapos-pp, simp+)

apply (frule-tac I = vp K v (Vr K v) (an (Suc n)) in
      Ring.ideal-inv1-closed[of Vr K v - -a(Vr K v) p], assumption+)
apply (simp add:aGroup.ag-inv-inv)
apply (frule-tac n = an (Suc n) in n-value-x-1[of v - p], simp)
apply assumption
apply simp
apply (simp add:ant-int, simp add:ale-natle)

apply (fold pol-limit-def)
apply (case-tac ad =  $\infty$ , simp)
apply simp apply (erule exE)
apply (subgoal-tac  $0 \leq ad$ )
apply (frule Plimit-deg[of v R X F na ad p], assumption+)
apply (simp add:an-na)+
```

```

apply (drule-tac x = n in spec,
      drule-tac x = n in spec)

apply (frule-tac p = F n in PolynRg.nonzero-deg-pos[of R Vr K v X],
      assumption+)
apply (rule-tac j = deg R (Vr K v) X (F n) in ale-trans[of 0 - ad],
      assumption+)
done

lemma (in Corps) Plimit-ldeg:[valuation K v; PolynRg R (Vr K v) X;
  ∀ n. F n ∈ carrier R; p ∈ carrier R;
  ∀ n. deg R (Vr K v) X (F n) ≤ an (Suc d);
  Plimit R X K v F p] ⟹ Plimit R X K v (Pseql R X K v d F)
  (ldeg-p R (Vr K v) X d p)
apply (frule Vr-ring[of v], frule PolynRg.is-Ring,
      frule Ring.ring-is-ag[of R])
apply (frule Plimit-deg[of v R X F Suc d p], assumption+)
apply (simp add:Pseql-def, simp add:pol-limit-def)
apply (rule conjI, rule allI)
apply (rule PolynRg.ldeg-p-mem, assumption+, simp+)
apply (rule allI)
apply (rotate-tac -5, drule-tac x = N in spec, erule exE)
apply (subgoal-tac ∀ m > M. P-mod R (Vr K v) X (vp K v (Vr K v) (an N))
  (ldeg-p R (Vr K v) X d (F m) ±R -aR (ldeg-p R (Vr K v) X d p)),
  blast)
apply (rule allI, rule impI)
apply (rotate-tac -2,
      frule-tac x = m in spec,
      thin-tac ∀ m. M < m →
      P-mod R (Vr K v) X (vp K v (Vr K v) (an N)) ( F m ±R -aR p),
      simp)
apply (subst v-ldeg-p-mOp[of v R X - d], assumption+)
apply (subst v-ldeg-p-pOp[of v R X - -aR p], assumption+)
apply (simp, rule aGroup.ag-mOp-closed, assumption, simp, simp)
apply (frule PolynRg.deg-minus-eq1 [THEN sym, of R Vr K v X p],
      assumption+)
apply simp
apply (rule-tac p = F m ±R -aR p and N = N in PCauchy-lTr[of v
  R X - d ], assumption+)
apply (rule-tac x = F m in aGroup.ag-pOp-closed[of R - -aR p],
      assumption+)
apply (simp, rule aGroup.ag-mOp-closed, assumption+)
apply (frule PolynRg.deg-minus-eq1 [of R Vr K v X p], assumption+)
apply (rule PolynRg.polyn-deg-add4[of R Vr K v X - -aR p Suc d],
      assumption+)
apply (simp, rule aGroup.ag-mOp-closed, assumption, simp+)
done

lemma (in Corps) Plimit-hdeg:[valuation K v; PolynRg R (Vr K v) X;

```

```

 $\forall n. F n \in \text{carrier } R; \forall n. \deg R (\text{Vr } K v) X (F n) \leq an (\text{Suc } d);$ 
 $p \in \text{carrier } R; \text{Plimit}_{R X K v} F p] \implies$ 
 $\text{Plimit}_{R X K v} (\text{Psegh}_{R X K v d} F) (\text{hdeg-}p R (\text{Vr } K v) X (\text{Suc } d) p)$ 
apply (frule Vr-ring[of v], frule PolynRg.is-Ring,
         frule Ring.ring-is-ag[of R])
apply (frule Plimit-deg[of v R X F Suc d p], assumption+)
apply (simp add:Psegh-def, simp add:pol-limit-def)
apply (rule conjI, rule allI)
apply (rule PolynRg.hdeg-p-mem, assumption+, simp+)
apply (rule allI)
apply (rotate-tac -5, drule-tac x = N in spec)
apply (erule exE)
apply (subgoal-tac  $\forall m > M. P\text{-mod } R (\text{Vr } K v) X (\text{vp } K v (\text{Vr } K v) (an N))$ 
        ( $\text{hdeg-}p R (\text{Vr } K v) X (\text{Suc } d) (F m) \pm_R -_a R (\text{hdeg-}p R (\text{Vr } K v) X (\text{Suc } d) p)$ ),
        blast)
apply (rule allI, rule impI)
apply (rotate-tac -2,
         drule-tac x = m in spec, simp)
apply (subst v-hdeg-p-mOp[of v R X - d], assumption+)
apply (subst v-hdeg-p-pOp[of v R X - -a R p], assumption+)
apply (simp, rule aGroup.ag-mOp-closed, assumption, simp, simp)
apply (frule PolynRg.deg-minus-eq1 [THEN sym, of R Vr K v X p], assumption+)
apply simp
apply (rule-tac p = F m \pm_R -_a R p and N = N in PCauchy-hTr[of v R X - d],
         assumption+)
apply (rule-tac x = F m in aGroup.ag-pOp-closed[of R - -a R p],
         assumption+)
apply (simp, rule aGroup.ag-mOp-closed, assumption+)
apply (frule PolynRg.deg-minus-eq1 [of R Vr K v X p], assumption+)
apply (rule PolynRg.polyn-deg-add4[of R Vr K v X - -a R p Suc d],
         assumption+)
apply (simp, rule aGroup.ag-mOp-closed, assumption, simp+)
done

```

```

lemma (in Corps) P-limit-uniqueTr: $\llbracket \text{valuation } K v; \text{PolynRg } R (\text{Vr } K v) X \rrbracket \implies$ 
 $\forall F. ((\forall n. F n \in \text{carrier } R) \wedge (\forall n. \deg R (\text{Vr } K v) X (F n) \leq (an d))) \longrightarrow$ 
 $(\forall p1 p2. p1 \in \text{carrier } R \wedge p2 \in \text{carrier } R \wedge \text{Plimit}_{R X K v} F p1 \wedge$ 
 $\text{Plimit}_{R X K v} F p2 \longrightarrow p1 = p2))$ 
apply (frule PolynRg.is-Ring)
apply (induct-tac d)
apply (rule allI, rule impI, (rule allI)+, rule impI)
apply (erule conjE)+
apply (subgoal-tac  $\forall n. F n \in \text{carrier } (\text{Vr } K v)$ )
apply (frule Vr-ring[of v])
apply (frule PolynRg.X-mem-R[of R Vr K v X])
apply (frule-tac f = F and F = F and d = 0 and b = p1 and c = p2 in
         mono-P-limit-unique[of v R X], assumption+)
apply (rule allI, drule-tac x = n in spec,

```

```

simp add:Ring.ring-r-one)
apply (frule-tac F = F and p = p1 in Plimit-deg[of v R X - 0],
      assumption+, simp add:deg-0-const,
      frule-tac F = F and p = p2 in Plimit-deg[of v R X - 0],
      assumption+, simp add:deg-0-const)
apply (simp add:Ring.ring-r-one)+
apply (simp add:deg-0-const)

apply (rename-tac d)
apply (rule allI, rule impI)
apply (erule conjE)
apply ((rule allI)+, rule impI, (erule conjE)+)
apply (frule-tac F = F and p = p1 and d = d in Plimit-ldeg[of v R X],
       assumption+,
       frule-tac F = F and p = p2 and d = d in Plimit-ldeg[of v R X],
       assumption+,
       frule-tac F = F and p = p1 and d = d in Plimit-hdeg[of v R X],
       assumption+,
       frule-tac F = F and p = p2 and d = d in Plimit-hdeg[of v R X],
       assumption+)
apply (frule-tac a = Pseql R X K v d F in forall-spec)
apply (rule conjI)
apply (rule allI)
apply (rule Pseql-mem, assumption+, simp)
apply (rule allI, simp)
apply (rule allI)
apply (subst Pseql-def)
apply (rule-tac p = F n and d = d in PolynRg.deg-ldeg-p[of R Vr K v X],
       assumption+) apply (simp add:Vr-ring)
apply simp
apply (thin-tac  $\forall F. (\forall n. F n \in \text{carrier } R) \wedge$ 
            $(\forall n. \text{deg } R (Vr K v) X (F n) \leq an d) \longrightarrow$ 
            $(\forall p1 p2.$ 
              $p1 \in \text{carrier } R \wedge$ 
              $p2 \in \text{carrier } R \wedge$ 
              $\text{Plimit } R X K v F p1 \wedge \text{Plimit } R X K v F p2 \longrightarrow$ 
              $p1 = p2))$ )
apply (frule Vr-ring[of v])
apply (frule-tac F = F and d = Suc d and p = p1 in
       Plimit-deg[of v R X], assumption+,
       frule-tac F = F and d = Suc d and p = p2 in
       Plimit-deg[of v R X], assumption+)
apply (subgoal-tac (ldeg-p R (Vr K v) X d p1) = (ldeg-p R (Vr K v) X d p2))
apply (subgoal-tac hdeg-p R (Vr K v) X (Suc d) p1 =
       hdeg-p R (Vr K v) X (Suc d) p2)

apply (frule-tac p = p1 and d = d in PolynRg.decompos-p[of R Vr K v X],
       assumption+,

```

```

frule-tac  $p = p2$  and  $d = d$  in PolynRg.decompos-p[of R Vr K v X],
assumption+)

apply simp
apply (thin-tac Plimit R X K v Pseql R X K v d F (ldeg-p R (Vr K v) X d p1),
       thin-tac Plimit R X K v Pseql R X K v d F (ldeg-p R (Vr K v) X d p2),
       thin-tac  $\forall p1 p2. p1 \in \text{carrier } R \wedge p2 \in \text{carrier } R \wedge$ 
              Plimit R X K v Pseql R X K v d F p1  $\wedge$ 
              Plimit R X K v Pseql R X K v d F p2  $\longrightarrow p1 = p2$ )
apply (simp only:hdeg-p-def)
apply (rule-tac  $f = \lambda j. \text{snd}(\text{scf-d } R (Vr K v) X (F j)) (\text{Suc } d)$ ) ( $\text{Suc } d$ )
and  $F = \text{Pseqh } R X K v d F$ 
and  $b = (\text{snd}(\text{scf-d } R (Vr K v) X p1 (\text{Suc } d))) (\text{Suc } d)$ 
and  $d = \text{Suc } d$  and  $c = \text{snd}(\text{scf-d } R (Vr K v) X p2 (\text{Suc } d)) (\text{Suc } d)$  in
mono-P-limit-unique[of v R X], assumption+)

apply (rule allI)
apply (frule-tac  $p = F n$  and  $d = \text{Suc } d$  in
           PolynRg.scf-d-pol[of R Vr K v X])
apply (simp del:npow-suc)+
apply (frule-tac  $c = \text{scf-d } R (Vr K v) X (F n) (\text{Suc } d)$  and
                $j = \text{Suc } d$  in PolynRg.pol-coeff-mem[of R Vr K v X])
apply simp apply (simp del:npow-suc)+
apply (rule allI)
apply (frule-tac  $c = \text{scf-d } R (Vr K v) X (F n) (\text{Suc } d)$  and
                $j = \text{Suc } d$  in PolynRg.pol-coeff-mem[of R Vr K v X])
apply (frule-tac  $p = F n$  and  $d = \text{Suc } d$  in PolynRg.scf-d-pol[of R
                     Vr K v X], (simp del:npow-suc)+)
apply (cut-tac  $p = F n$  and  $d = \text{Suc } d$  in PolynRg.scf-d-pol[of R
                     Vr K v X], (simp del:npow-suc)+)

apply (subst Pseqh-def) apply (simp only:hdeg-p-def)
apply (frule-tac  $p = p1$  and  $d = \text{Suc } d$  in PolynRg.scf-d-pol[of R Vr K v X],
assumption+)
apply (rule-tac  $c = \text{scf-d } R (Vr K v) X p1 (\text{Suc } d)$  and
                $j = \text{Suc } d$  in PolynRg.pol-coeff-mem[of R Vr K v X], assumption+,
simp, simp)
apply (frule-tac  $p = p2$  and  $d = \text{Suc } d$  in PolynRg.scf-d-pol[of R Vr K v X],
assumption+)
apply (rule-tac  $c = \text{scf-d } R (Vr K v) X p2 (\text{Suc } d)$  and
                $j = \text{Suc } d$  in PolynRg.pol-coeff-mem[of R Vr K v X], assumption+,
simp, simp) apply simp apply simp
apply (rotate-tac -4,
drule-tac  $x = \text{ldeg-p } R (Vr K v) X d p1$  in spec,
rotate-tac -1,
drule-tac  $x = \text{ldeg-p } R (Vr K v) X d p2$  in spec)
apply (simp add:PolynRg.ldeg-p-mem)
done

lemma (in Corps) P-limit-unique:[valuation K v; Completev K;
PolynRg R (Vr K v) X;  $\forall n. F n \in \text{carrier } R$ ];

```

```

 $\forall n. \deg R (Vr K v) X (F n) \leq (an d); p1 \in \text{carrier } R; p2 \in \text{carrier } R;$ 
 $P\text{-limit}_R X K v F p1; P\text{-limit}_R X K v F p2] \implies p1 = p2$ 
apply (frule P-limit-uniqueTr[of v R X d], assumption)
apply blast
done

lemma (in Corps) P-limitTr: $\llbracket \text{valuation } K v; \text{Complete}_v K; \text{PolynRg } R (Vr K v)$ 
 $X \rrbracket$ 
 $\implies \forall F. ((\forall n. F n \in \text{carrier } R) \wedge (\forall n. \deg R (Vr K v) X (F n) \leq (an d)) \wedge$ 
 $(\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow$ 
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N)) (F n \pm_R -_a R (F m))) \longrightarrow$ 
 $(\exists p \in \text{carrier } R. P\text{-limit}_R X K v F p))$ 
apply (frule PolynRg.is-Ring)
apply (frule Vr-ring[of v])
apply (induct-tac d)

apply simp
apply (rule allI, rule impI, (erule conjE)+)
apply (frule-tac F = F and f = F in monomial-P-limit[of v R X - - 0],
assumption+)
apply (rule allI)
apply (rotate-tac 5, drule-tac x = n in spec)
apply (simp add:deg-0-const)
apply (rule allI)
apply (drule-tac x = n in spec,
thin-tac  $\forall n. \deg R (Vr K v) X (F n) \leq 0$ )
apply (frule PolynRg.X-mem-R[of R Vr K v X],
frule PolynRg.is-Ring,
simp add:Ring.ring-r-one, assumption)

apply (erule bxE)
apply (frule PolynRg.subring[of R Vr K v X])
apply (cut-tac x = b in Ring.mem-subring-mem-ring[of R Vr K v])
apply (frule PolynRg.X-mem-R[of R Vr K v X], assumption+)
apply (simp add:Ring.ring-r-one, blast)

apply (rule allI, rule impI) apply (rename-tac d F)
apply (erule conjE)+
apply (subgoal-tac  $(\forall n. (Pseql R X K v d F) n \in \text{carrier } R) \wedge$ 
 $(\forall n. \deg R (Vr K v) X ((Pseql R X K v d F) n) \leq an d) \wedge$ 
 $(\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow$ 
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
 $((Pseql R X K v d F) n \pm_R -_a R ((Pseql R X K v d F) m)))$ )
apply (frule-tac a = Pseql R X K v d F in forall-spec, assumption)
apply (erule bxE)
apply (thin-tac  $\forall F. (\forall n. F n \in \text{carrier } R) \wedge$ 
 $(\forall n. \deg R (Vr K v) X (F n) \leq an d) \wedge$ 
 $(\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow$ 
```

```

 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N)) (F n \pm_R -_a R (F m)) \longrightarrow$ 
 $(\exists p \in carrier R. Plimit_{R X K v} F p),$ 
 $thin\text{-tac } (\forall n. (Pseql_{R X K v d} F) n \in carrier R) \wedge$ 
 $(\forall n. deg R (Vr K v) X ((Pseql_{R X K v d} F) n) \leq an d) \wedge$ 
 $(\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
 $((Pseql_{R X K v d} F) n \pm_R -_a R ((Pseql_{R X K v d} F) m)))$ 
apply (subgoal-tac  $\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow$ 
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
 $((Pseqh_{R X K v d} F) n \pm_R -_a R ((Pseqh_{R X K v d} F) m)))$ 
apply (frule-tac  $f = \lambda j. snd (scf-d R (Vr K v) X (F j) (Suc d)) (Suc d)$ )
and  $F = Pseqh_{R X K v d} F$  and  $d = Suc d$  in monomial-P-limt[of v R X],
assumption+)
apply (rule allI)
apply (drule-tac  $x = n$  in spec,
drule-tac  $x = n$  in spec,
frule-tac  $p = F n$  and  $d = Suc d$  in PolynRg.scf-d-pol[of R Vr K v X], assumption+, (erule conjE)+)
apply (rule-tac  $c = scf-d R (Vr K v) X (F n) (Suc d)$  and  $j = Suc d$  in
PolynRg.pol-coeff-mem[of R Vr K v X], assumption+)
apply simp
apply (rule allI)
apply (thin-tac  $\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow$ 
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
 $((Pseqh_{R X K v d} F) n \pm_R -_a R ((Pseqh_{R X K v d} F) m)))$ 
apply (simp only:Pseqh-def hdeg-p-def, assumption, erule bexE)

apply (thin-tac  $\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow$ 
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N)) (F n \pm_R -_a R (F m)),$ 
thin-tac  $\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow$ 
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
 $((Pseqh_{R X K v d} F) n \pm_R -_a R ((Pseqh_{R X K v d} F) m)))$ 
apply (subgoal-tac  $Plimit_{R X K v} F (p \pm_R b \cdot_r R (X^{\sim R} (Suc d))),$ 
subgoal-tac  $p \pm_R b \cdot_r R (X^{\sim R} (Suc d)) \in carrier R$ , blast)

apply (frule PolynRg.X-mem-R[of R Vr K v X],
frule-tac  $n = Suc d$  in Ring,npClose[of R X], assumption+,
frule Ring.ring-is-ag[of R],
rule aGroup.ag-pOp-closed[of R], assumption+,
rule Ring.ring-tOp-closed, assumption+)
apply (frule PolynRg.substring[of R Vr K v X],
simp add:Ring.mem-substring-mem-ring, assumption)

apply (simp del:npow-suc add:pol-limit-def,
rule allI,
subgoal-tac  $\forall n. F n = (Pseql_{R X K v d} F) n \pm_R ((Pseqh_{R X K v d} F) n),$ 
simp del:npow-suc,
```

$\text{subgoal-tac } \forall m. (Pseql_R X K v d F) m \pm_R ((Pseqh_R X K v d F) m) \pm_R$
 $-_a R (p \pm_R (b \cdot_r R (X^{\sim R} (\text{Suc } d)))) = (Pseql_R X K v d F) m \pm_R -_a R p$
 $\pm_R ((Pseqh_R X K v d F) m \pm_R -_a R (b \cdot_r R (X^{\sim R} (\text{Suc } d)))),$
 $\text{simp del:npow-suc})$

apply (*thin-tac* $\forall m. (Pseql_R X K v d F) m \pm_R (Pseqh_R X K v d F) m \pm_R$
 $-_a R (p \pm_R b \cdot_r R X^{\sim R} (\text{Suc } d)) = (Pseql_R X K v d F) m \pm_R -_a R p \pm_R$
 $((Pseqh_R X K v d F) m \pm_R -_a R b \cdot_r R X^{\sim R} (\text{Suc } d)))$)

apply (*erule conjE*)+

apply (*rotate-tac* -3 ,
drule-tac $x = N$ **in** *spec*, *erule exE*)

apply (*rotate-tac* 1 ,
drule-tac $x = N$ **in** *spec*, *erule exE*)

apply (*rename-tac* $d F p b N M1 M2$)

apply (*subgoal-tac* $\forall m. (\max M1 M2) < m \rightarrow$
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$
 $((Pseql_R X K v d F) m \pm_R -_a R p \pm_R$
 $((Pseqh_R X K v d F) m \pm_R -_a R (b \cdot_r R (X^{\sim R} (\text{Suc } d))))), \text{blast}$)

apply (*rule allI*, *rule impI*)

apply (*rotate-tac* -2 ,
drule-tac $x = m$ **in** *spec*, *simp del:npow-suc*)

apply (*erule conjE*)

apply (*rotate-tac* -5 ,
drule-tac $x = m$ **in** *spec*, *simp del:npow-suc*)

apply (*frule-tac* $n = an N$ **in** *vp-apow-ideal*[*of v*], *simp del:npow-suc*)

apply (*frule Ring.ring-is-ag*[*of R*])

apply (*rule-tac* $I = vp K v (Vr K v) (an N)$ **and**
 $p = (Pseql_R X K v d F) m \pm_R -_a R p$ **and**
 $q = (Pseqh_R X K v d F) m \pm_R -_a R (b \cdot_r R (X^{\sim R} (\text{Suc } d)))$ **in**
 $\text{PolynRg.P-mod-add}[R Vr K v X], \text{assumption+})$

apply (*rule aGroup.ag-pOp-closed*, *assumption+*)

apply (*simp add:Pseql-mem*, *rule aGroup.ag-mOp-closed*, *assumption+*)

apply (*rule aGroup.ag-pOp-closed*[*of R*], *assumption*)

apply (*simp add:Pseqh-mem*, *rule aGroup.ag-mOp-closed*, *assumption*)

apply (*rule Ring.ring-tOp-closed*, *assumption*)

apply (*frule PolynRg.subring*[*of R Vr K v X*])

apply (*simp add:Ring.mem-subring-mem-ring*)

apply (*frule PolynRg.X-mem-R*[*of R Vr K v X*])

apply (*rule Ring.npClose*, *assumption+*)

apply (*rule allI*)

apply (*thin-tac* $\forall n. (Pseql_R X K v d F) n \pm_R ((Pseqh_R X K v d F) n) \in \text{carrier } R$)

apply (*erule conjE*)+

apply (*thin-tac* $\forall n. \deg R (Vr K v) X$)

```

 $((Pseql_R X K v d F) n \pm_R (Pseqh_R X K v d F) n) \leq an(Suc d))$ 
apply (thin-tac  $\forall N. \exists M. \forall m > M. P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
 $((Pseql_R X K v d F) m \pm_R -_a R p),$ 
thin-tac  $\forall N. \exists M. \forall m > M. P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
 $((Pseqh_R X K v d F) m \pm_R -_a R b \cdot_r R X^{\sim R} (Suc d)),$ 
thin-tac  $\forall n. F n = (Pseql_R X K v d F) n \pm_R ((Pseqh_R X K v d F) n))$ 
apply (drule-tac  $x = m$  in spec,
 $drule-tac x = m$  in spec)
apply (subgoal-tac  $b \cdot_r R X^{\sim R} (Suc d) \in carrier R$ )
apply (frule Ring.ring-is-ag[of R])
apply (frule-tac  $x = p$  in aGroup.ag-mOp-closed[of R], assumption+)
apply (subst aGroup.ag-pOp-assoc[of R], assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+)
apply (frule-tac  $x1 = -_a R p$  and  $y1 = (Pseqh_R X K v d F) m$  and  $z1 =$ 
 $-_a R (b \cdot_r R (X^{\sim R} (Suc d)))$  in aGroup.ag-pOp-assoc[THEN sym, of R],
assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+, simp del:npow-suc)
apply (subst aGroup.ag-pOp-assoc[THEN sym], assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+,
rule aGroup.ag-pOp-closed, assumption+)
apply (subst aGroup.ag-add4-rel[of R], assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+)
apply (subst aGroup.ag-p-inv[THEN sym, of R], assumption+, simp del:npow-suc)

apply (rule Ring.ring-tOp-closed, assumption+,
frule PolynRg.substring[of R Vr K v X],
simp add:Ring.mem-subring-mem-ring,
rule Ring.npClose, assumption+, simp add:PolynRg.X-mem-R)
apply (rule allI)
apply (rule-tac  $F = F$  and  $n = n$  and  $d = d$  in Pseq-decompos[of v R X],
assumption+, simp, simp)
apply (rule allI)
apply (rotate-tac  $-3$ , drule-tac  $x = N$  in spec)
apply (erule exE)
apply (subgoal-tac  $\forall n m. M < n \wedge M < m \rightarrow$ 
 $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
 $((Pseqh_R X K v d F) n \pm_R -_a R ((Pseqh_R X K v d F) m)))$ 
apply blast
apply ((rule allI)+, rule impI)
apply (rotate-tac  $-2$ ,
drule-tac  $x = n$  in spec,
rotate-tac  $-1$ ,
drule-tac  $x = m$  in spec,
simp)
apply (simp only: Pseqh-def)
apply (subst v-hdeg-p-mOp[of v R X], assumption+)

```

```

apply simp+

apply (frule Ring.ring-is-ag[of R])
apply (subst v-hdeg-p-pOp[of v R X], assumption+)
apply (simp, rule aGroup.ag-mOp-closed, assumption, simp, simp,
         frule-tac p1 = F m in PolynRg.deg-minus-eq1 [THEN sym, of R Vr K v
         X])
apply simp
apply (rotate-tac -1, frule sym,
         thin-tac deg R (Vr K v) X (F m) = deg R (Vr K v) X (-aR (F m)), simp)
apply (rule PCauchy-hTr[of v R X], assumption+)
apply (rule-tac x = F n and y = -aR (F m) in aGroup.ag-pOp-closed[of R],
         assumption+,
         simp, rule aGroup.ag-mOp-closed, assumption+, simp)
apply (frule-tac p = F m in PolynRg.deg-minus-eq1 [of R Vr K v X],
         simp,
         rule-tac p = F n and q = -aR (F m) and n = Suc d in
         PolynRg.polyn-deg-add4[of R Vr K v X], assumption+,
         simp, rule aGroup.ag-mOp-closed, assumption, simp+)

apply (rule conjI, rule allI, rule Pseql-mem, assumption+, simp)
apply (rule allI, simp)

apply (thin-tac  $\forall F. (\forall n. F n \in \text{carrier } R) \wedge$ 
          $(\forall n. \text{deg } R (Vr K v) X (F n) \leq \text{an } d) \wedge$ 
          $(\forall N. \exists M. \forall n m. M < n \wedge M < m \longrightarrow$ 
          $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N)) (F n \pm_R -aR (F m))) \longrightarrow$ 
          $(\exists p \in \text{carrier } R. \text{Plimit}_R X K v F p)$ )

apply (rule conjI, rule allI)
apply (subst Pseql-def)
apply (rule-tac p = F n and d = d in PolynRg.deg-ldeg-p[of R Vr K v X],
         assumption+)
apply simp+

apply (simp only: Pseql-def)
apply (rule allI)
apply (rotate-tac -1, drule-tac x = N in spec)
apply (erule exE)
apply (subgoal-tac  $\forall n m. M < n \wedge M < m \longrightarrow$ 
          $P\text{-mod } R (Vr K v) X (vp K v (Vr K v) (an N))$ 
          $(ldeg-p R (Vr K v) X d (F n) \pm_R -aR (ldeg-p R (Vr K v) X d (F m)))$ )
apply blast
apply (((rule allI)+, rule impI, erule conjE)
apply (rotate-tac -3, drule-tac x = n in spec,
         rotate-tac -1, drule-tac x = m in spec, simp)

apply (subst v-ldeg-p-mOp[of v R X], assumption+, simp+)

```

```

apply (frule Ring.ring-is-ag[of R])
apply (subst v-ldeg-p-pOp[of v R X], assumption+, simp,
       rule aGroup.ag-mOp-closed, assumption, simp, simp,
       frule-tac p = F m in PolynRg.deg-minus-eq1[of R Vr K v X], simp)
apply simp
apply (rule PCauchy-lTr[of v R X], assumption+)
apply (rule-tac x = F n and y = -_aR (F m) in aGroup.ag-pOp-closed[of R],
       assumption+,
       simp, rule aGroup.ag-mOp-closed, assumption+, simp)

apply (frule-tac p = F m in PolynRg.deg-minus-eq1[of R Vr K v X], simp,
       rule-tac p = F n and q = -_aR (F m) and n = Suc d in
       PolynRg.polyn-deg-add4[of R Vr K v X], assumption+,
       simp, rule aGroup.ag-mOp-closed, assumption, simp+)
done

lemma (in Corps) PCauchy-Plimit:[valuation K v; Completev K;
  PolynRg R (Vr K v) X; PCauchyR X K v F] ==>
  ∃ p ∈ carrier R. PlimitR X K v F p
apply (simp add:pol-Cauchy-seq-def)
apply ((erule conjE)+, erule exE)
apply (frule-tac d = d in P-limitTr[of v R X], assumption+)
apply (drule-tac a = F in forall-spec, simp)
apply assumption
done

lemma (in Corps) P-limit-mult:[valuation K v; PolynRg R (Vr K v) X;
  ∀ n. F n ∈ carrier R; ∀ n. G n ∈ carrier R; p1 ∈ carrier R; p2 ∈ carrier R;
  Plimit R X K v F p1; Plimit R X K v G p2] ==>
  Plimit R X K v (λn. (F n) ·_rR (G n)) (p1 ·_rR p2)
apply (frule Vr-ring[of v],
       frule PolynRg.is-Ring,
       frule Ring.ring-is-ag[of R])
apply (simp add:pol-limit-def)
apply (rule conjI)
apply (rule allI)
apply (drule-tac x = n in spec,
       drule-tac x = n in spec)

apply (simp add:Ring.ring-tOp-closed[of R])

apply (rule allI)
apply (rotate-tac 6,
       drule-tac x = N in spec,
       drule-tac x = N in spec)
apply (erule exE, erule exE, rename-tac N M1 M2)
apply (subgoal-tac ∀ m. (max M1 M2) < m —>
  P-mod R (Vr K v) X (vp K v (Vr K v) (an N))
  ((F m) ·_rR (G m) ±_R -_aR (p1 ·_rR p2)))

```

```

apply blast

apply (rule allI, rule impI, simp, erule conjE)
apply (rotate-tac -4,
       drule-tac x = m in spec,
       drule-tac x = m in spec, simp)
apply (subgoal-tac (F m) ·_R (G m) ±_R -_aR p1 ·_R p2 =
       ((F m) ±_R -_aR p1) ·_R (G m) ±_R p1 ·_R ((G m) ±_R -_aR p2), simp)

apply (frule-tac n = an N in vp-apow-ideal[of v])
apply simp
apply (rule-tac I = vp K v (Vr K v) (an N) and
       p = ((F m) ±_R -_aR p1) ·_R (G m) and q = p1 ·_R ((G m) ±_R -_aR p2)
       in PolynRg.P-mod-add[of R Vr K v X],
       assumption+)
apply (rule Ring.ring-tOp-closed[of R], assumption+)
apply (rule aGroup.ag-pOp-closed, assumption) apply simp
apply (rule aGroup.ag-mOp-closed, assumption+) apply simp
apply (rule Ring.ring-tOp-closed[of R], assumption+)
apply (rule aGroup.ag-pOp-closed, assumption) apply simp
apply (rule aGroup.ag-mOp-closed, assumption+)
apply (rule Ring.whole-ideal[of Vr K v])
apply (frule-tac I = vp K v (Vr K v) (an N) and J = carrier (Vr K v) and
       p = F m ±_R -_aR p1 and q = G m in PolynRg.P-mod-mult1[of R Vr K v X],
       assumption+,
       rule aGroup.ag-pOp-closed, assumption+, simp, rule aGroup.ag-mOp-closed,
       assumption+) apply simp apply assumption
apply (rotate-tac 8,
       drule-tac x = m in spec)
apply (case-tac G m = 0_R, simp add:P-mod-def)
apply (frule-tac p = G m in PolynRg.s-cf-expr[of R Vr K v X], assumption+,
       (erule conjE)+)
thm PolynRg.P-mod-mod
apply (frule-tac I1 = carrier (Vr K v) and p1 = G m and
       c1 = s-cf R (Vr K v) X (G m) in PolynRg.P-mod-mod[THEN sym,
       of R Vr K v X], assumption+)
apply (simp,
       thin-tac P-mod R (Vr K v) X (carrier (Vr K v)) (G m) =
       (∀j≤fst (s-cf R (Vr K v) X (G m)).
        snd (s-cf R (Vr K v) X (G m)) j ∈ carrier (Vr K v)))
apply (rule allI, rule impI)
apply (simp add:PolynRg.pol-coeff-mem)
apply (simp add:Ring.idealprod-whole-r[of Vr K v])

apply (cut-tac I = carrier (Vr K v) and J = vp K v (Vr K v) (an N) and
       p = p1 and q = G m ±_R -_aR p2 in PolynRg.P-mod-mult1[of R Vr K v X],
       assumption+)
apply (simp only: Ring.whole-ideal, assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+, simp, rule aGroup.ag-mOp-closed,
       assumption+)

```

```

assumption+)
apply (frule PolynRg.s-cf-expr0[of R Vr K v X p1], assumption+)
thm PolynRg.P-mod-mod
apply (cut-tac I1 = carrier (Vr K v) and p1 = p1 and
      c1 = s-cf R (Vr K v) X p1 in PolynRg.P-mod-mod[THEN sym,
      of R Vr K v X], assumption+)
apply (simp add:Ring.whole-ideal, assumption+)
apply (simp, simp, simp, (erule conjE)+,
      thin-tac P-mod R (Vr K v) X (carrier (Vr K v)) p1 =
      (forall j leq fst (s-cf R (Vr K v) X p1).
       snd (s-cf R (Vr K v) X p1) j in carrier (Vr K v)))
apply (rule allI, rule impI)
apply (simp add:PolynRg.pol-coeff-mem, assumption)
apply (simp add:Ring.idealprod-whole-l[of Vr K v])
apply (drule-tac x = m in spec,
      drule-tac x = m in spec)
apply (frule aGroup.ag-mOp-closed[of R p1], assumption,
      frule aGroup.ag-mOp-closed[of R p2], assumption )
apply (simp add:Ring.ring-distrib1 Ring.ring-distrib2)
apply (subst aGroup.pOp-assocTr43[of R], assumption+,
      (rule Ring.ring-tOp-closed, assumption+)+)
apply (simp add:Ring.ring-inv1-1[THEN sym],
      simp add:Ring.ring-inv1-2[THEN sym])
apply (frule-tac x = p1 and y = G m in Ring.ring-tOp-closed, assumption+,
      frule-tac x = F m and y = G m in Ring.ring-tOp-closed, assumption+,
      simp add:aGroup.ag-l-inv1 aGroup.ag-r-zero)
done

definition
Hfst :: [-, 'b ⇒ ant, ('b, 'm1) Ring-scheme, 'b,'b, ('b set, 'm2) Ring-scheme, 'b
set, 'b, 'b, 'b, nat] ⇒ 'b
((11Hfst - - - - -) [67,67,67,67,67,67,67,67,67,67,68]67) where
Hfst_K v R X t S Y f g h m = fst (Hpr_R (Vr K v) X t S Y f g h m)

definition
Hsnd :: [-, 'b ⇒ ant, ('b, 'm1) Ring-scheme, 'b,'b, ('b set, 'm2) Ring-scheme, 'b
set, 'b, 'b, 'b, nat] ⇒ 'b
((11Hsnd - - - - -) [67,67,67,67,67,67,67,67,67,67,68]67) where
Hsnd_K v R X t S Y f g h m = snd (Hpr_R (Vr K v) X t S Y f g h m)

lemma (in Corps) Hensel-starter: [| valuation K v; Complete_v K;
PolynRg R (Vr K v) X; PolynRg S ((Vr K v) /_r (vp K v)) Y;
t in carrier (Vr K v); vp K v = (Vr K v) ◇_p t;
f in carrier R; f ≠ 0_R; g' in carrier S; h' in carrier S;
0 < deg S ((Vr K v) /_r (vp K v)) Y g';
0 < deg S ((Vr K v) /_r (vp K v)) Y h';
((erH R (Vr K v) X S ((Vr K v) /_r (vp K v)) Y
      (pj (Vr K v) (vp K v))) f) = g' ·_S h';

```

$\text{rel-prime-pol} S ((\text{Vr } K \ v) /_r (\text{vp } K \ v)) \ Y \ g' \ h \] \implies$
 $\exists g \ h. \ g \neq \mathbf{0}_R \wedge h \neq \mathbf{0}_R \wedge g \in \text{carrier } R \wedge h \in \text{carrier } R \wedge$
 $\deg R (\text{Vr } K \ v) \ X \ g \leq \deg S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t)) \ Y$
 $(\text{erH } R (\text{Vr } K \ v) \ X \ S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t))) \ Y$
 $(\text{pj } (\text{Vr } K \ v) ((\text{Vr } K \ v) \ \diamond_p \ t)) \ g) \wedge (\deg R (\text{Vr } K \ v) \ X \ h +$
 $\deg S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t)) \ Y (\text{erH } R (\text{Vr } K \ v) \ X \ S$
 $((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t)) \ Y (\text{pj } (\text{Vr } K \ v) ((\text{Vr } K \ v) \ \diamond_p \ t)) \ g)$
 $\leq \deg R (\text{Vr } K \ v) \ X \ f) \wedge$
 $(\text{erH } R (\text{Vr } K \ v) \ X \ S ((\text{Vr } K \ v) /_r (\text{vp } K \ v)) \ Y$
 $(\text{pj } (\text{Vr } K \ v) (\text{vp } K \ v))) \ g = g' \wedge$
 $(\text{erH } R (\text{Vr } K \ v) \ X \ S ((\text{Vr } K \ v) /_r (\text{vp } K \ v)) \ Y$
 $(\text{pj } (\text{Vr } K \ v) (\text{vp } K \ v))) \ h = h' \wedge$
 $0 < \deg S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t)) \ Y$
 $(\text{erH } R (\text{Vr } K \ v) \ X \ S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t))) \ Y$
 $(\text{pj } (\text{Vr } K \ v) ((\text{Vr } K \ v) \ \diamond_p \ t)) \ g) \wedge$
 $0 < \deg S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t)) \ Y$
 $(\text{erH } R (\text{Vr } K \ v) \ X \ S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t))) \ Y$
 $(\text{pj } (\text{Vr } K \ v) ((\text{Vr } K \ v) \ \diamond_p \ t)) \ h) \wedge$
 $\text{rel-prime-pol} S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t)) \ Y$
 $(\text{erH } R (\text{Vr } K \ v) \ X \ S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t))) \ Y$
 $(\text{pj } (\text{Vr } K \ v) ((\text{Vr } K \ v) \ \diamond_p \ t)) \ g)$
 $(\text{erH } R (\text{Vr } K \ v) \ X \ S ((\text{Vr } K \ v) /_r ((\text{Vr } K \ v) \ \diamond_p \ t))) \ Y$
 $(\text{pj } (\text{Vr } K \ v) ((\text{Vr } K \ v) \ \diamond_p \ t)) \ h) \wedge$
 $P\text{-mod } R (\text{Vr } K \ v) \ X ((\text{Vr } K \ v) \ \diamond_p \ t) (f \pm_R -a_R (g \cdot_R h))$
apply (*frule Vr-ring[of v]*,
frule PolynRg.subring[of R Vr K v X],
frule vp-maximal [of v], *frule PolynRg.is-Ring*,
frule Ring.subring-Ring[of R Vr K v], *assumption+*,
frule Ring.residue-field-cd[of Vr K v vp K v], *assumption+*,
frule Corps.field-is-ring[of Vr K v /_r vp K v],
frule pj-Hom[of Vr K v vp K v], *frule vp-ideal[of v]*,
simp add:Ring.maximal-ideal-ideal)
apply (*frule Corps.field-is-idom[of (Vr K v) /_r (vp K v)]*,
frule Vr-integral[of v], *simp*,
frule Vr-mem-f-mem[of v t], *assumption+*)
apply (*frule PolynRg.erH-inv[of R Vr K v X Vr K v \diamond_p t S Y g]*,
assumption+, *simp add:Ring.maximal-ideal-ideal*,
simp add:PolynRg.is-Ring, *assumption+*, *erule bxE*, *erule conjE*)
apply (*frule PolynRg.erH-inv[of R Vr K v X Vr K v \diamond_p t S Y h]*,
assumption+, *simp add:Ring.maximal-ideal-ideal*,
simp add:PolynRg.is-Ring, *assumption+*, *erule bxE*, *erule conjE*)
apply (*rename-tac g0 h0*)
apply (*subgoal-tac g0 \neq \mathbf{0}_R \wedge h0 \neq \mathbf{0}_R \wedge*
deg R (Vr K v) X g0 \leq
deg S (Vr K v /_r (Vr K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K \ v) \ X \ S
 $(\text{Vr } K \ v /_r (\text{Vr } K \ v \ \diamond_p \ t)) \ Y (\text{pj } (\text{Vr } K \ v) (\text{Vr } K \ v \ \diamond_p \ t)) \ g0) \wedge$
deg R (Vr K v) X h0 + deg S (Vr K v /_r (Vr K v \ \diamond_p \ t)) Y
 $(\text{erH } R (\text{Vr } K \ v) \ X \ S (\text{Vr } K \ v /_r (\text{Vr } K \ v \ \diamond_p \ t)) \ Y$
 $(\text{pj } (\text{Vr } K \ v) (\text{Vr } K \ v \ \diamond_p \ t)) \ g0) \leq \deg R (\text{Vr } K \ v) \ X \ f \wedge$

$0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0) \wedge$
 $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0) \wedge$
 $\text{rel-prime-pols } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0)$
 $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0) \wedge$
 $P\text{-mod } R (\text{Vr } K v) X (\text{Vr } K v \diamond_p t) (f \pm_R -a_R g0 \cdot_r R h0))$
apply (*thin-tac* $g' \in \text{carrier } S$,
thin-tac $h' \in \text{carrier } S$,
thin-tac $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y g'$,
thin-tac $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y h'$,
thin-tac $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) f = g' \cdot_r S h'$,
thin-tac $\text{rel-prime-pols } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y g' h'$,
thin-tac $\text{Corps } (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t))$,
thin-tac $\text{Ring } (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t))$,
thin-tac $\text{vp } K v = \text{Vr } K v \diamond_p t)$
apply *blast*
apply (*rule conjI*)
apply (*thin-tac* $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y h'$,
thin-tac $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) f = g' \cdot_r S h'$,
thin-tac $\text{rel-prime-pols } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y g' h'$,
thin-tac $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0 = h'$,
thin-tac $\deg R (\text{Vr } K v) X h0 \leq \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y h')$
apply (*rule contrapos-pp, simp+*)
apply (*simp add:PolynRg.erH-rHom-0[of R Vr K v X S*
 $\text{Vr } K v /_r (\text{Vr } K v \diamond_p t) Y \text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)]$)
apply (*rotate-tac -3, drule sym, simp add:deg-def*)
apply (*drule aless-imp-le[of 0 -∞],*
cut-tac minf-le-any[of 0],
frule ale-antisym[of 0 -∞], simp only:ant-0[THEN sym], simp)

apply (*rule conjI*)
apply (*thin-tac* $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y g'$,
thin-tac $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) f = g' \cdot_r S h'$,
thin-tac $\text{rel-prime-pols } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y g' h'$,
thin-tac $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0 = g'$,
thin-tac $\deg R (\text{Vr } K v) X h0 \leq \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y h')$
apply (*rule contrapos-pp, simp+,*
simp add:PolynRg.erH-rHom-0[of R Vr K v X S
 $\text{Vr } K v /_r (\text{Vr } K v \diamond_p t) Y \text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)]$)
apply (*rotate-tac -2, drule sym, simp add:deg-def*)

```

apply (frule aless-imp-le[of 0 -∞], thin-tac  $0 < -\infty$ ,
         cut-tac minf-le-any[of 0],
         frule ale-antisym[of 0 -∞], simp only:ant-0[THEN sym], simp)
apply (frule-tac  $x = \deg R (\forall r K v) X h0$  and
          $y = \deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y h'$  and
          $z = \deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y g'$  in aadd-le-mono)
apply (simp add:PolynRg.deg-mult-pols1[THEN sym, of S
          $\forall r K v /_r (\forall r K v \diamond_p t) Y h' g']$ )
apply (frule PolynRg.is-Ring[of S  $\forall r K v /_r (\forall r K v \diamond_p t) Y$ ],
         simp add:Ring.ring-tOp-commute[of S h' g'])
apply (rotate-tac 11, drule sym)
apply simp
apply (frule PolynRg.erH-rHom[of R  $\forall r K v X S$ 
          $(\forall r K v) /_r (\forall r K v \diamond_p t) Y pj (\forall r K v) (\forall r K v \diamond_p t)$ ],
         assumption+)
apply (frule PolynRg.pHom-dec-deg[of R  $\forall r K v X S (\forall r K v) /_r (\forall r K v \diamond_p t)$ 
          $Y erH R (\forall r K v) X S ((\forall r K v) /_r (\forall r K v \diamond_p t))$ 
          $Y (pj (\forall r K v) (\forall r K v \diamond_p t)) f$ ], assumption+)
apply (frule-tac  $i = \deg R (\forall r K v) X h0 +$ 
          $\deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y g'$  in ale-trans[of -
          $\deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(erH R (\forall r K v) X S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(pj (\forall r K v) (\forall r K v \diamond_p t)) f) \deg R (\forall r K v) X f$ ],
         assumption+) apply simp
apply (thin-tac  $\deg R (\forall r K v) X h0 +$ 
          $\deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y g' \leq \deg R (\forall r K v) X f$ ,
         thin-tac  $\deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(erH R (\forall r K v) X S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(pj (\forall r K v) (\forall r K v \diamond_p t)) f) \leq \deg R (\forall r K v) X f$ ,
         thin-tac  $0 < \deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y g'$ ,
         thin-tac  $0 < \deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y h'$ ,
         thin-tac  $\deg R (\forall r K v) X h0 + \deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y g'$ 
          $\leq \deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(erH R (\forall r K v) X S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(pj (\forall r K v) (\forall r K v \diamond_p t)) f)$ ,
         thin-tac rel-prime-pols  $S (\forall r K v /_r (\forall r K v \diamond_p t)) Y g' h'$ )
apply (rotate-tac 12, drule sym)
apply (drule sym)
apply simp
apply (frule-tac  $x = g0$  and  $y = h0$  in Ring.ring-tOp-closed[of R],
         assumption+)
apply (thin-tac  $\deg R (\forall r K v) X h0 \leq \deg S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(erH R (\forall r K v) X S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(pj (\forall r K v) (\forall r K v \diamond_p t)) h0)$ ,
         thin-tac  $h' = erH R (\forall r K v) X S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(pj (\forall r K v) (\forall r K v \diamond_p t)) h0$ ,
         thin-tac  $g' = erH R (\forall r K v) X S (\forall r K v /_r (\forall r K v \diamond_p t)) Y$ 
          $(pj (\forall r K v) (\forall r K v \diamond_p t)) g0$ ,

```

$\text{thin-tac } \deg R (\text{Vr } K v) X g0 \leq \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0))$
apply (*subst PolynRg.P-mod-diff[THEN sym, of R Vr K v X Vr K v \diamond_p t] S Y f*, *assumption+*) **apply** (*simp add:Ring.maximal-ideal-ideal, assumption+*)
apply (*rotate-tac 12, drule sym*)
apply (*subst PolynRg.erH-mult[of R Vr K v X S Vr K v /_r (Vr K v \diamond_p t) Y], assumption+*)
done

lemma *aadd-plus-le-plus:* $\llbracket a \leq (a'::\text{ant}); b \leq b' \rrbracket \implies a + b \leq a' + b'$
apply (*frule aadd-le-mono[of a a' b]*)
apply (*frule aadd-le-mono[of b b' a']*)
apply (*simp add:aadd-commute[of - a']*)
done

lemma (in Corps) *Hfst-PCauchy:* $\llbracket \text{valuation } K v; \text{Complete}_v K;$
 $\text{PolynRg } R (\text{Vr } K v) X; \text{PolynRg } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y; g0 \in \text{carrier } R;$
 $h0 \in \text{carrier } R; f \in \text{carrier } R; f \neq \mathbf{0}_R; g0 \neq \mathbf{0}_R; h0 \neq \mathbf{0}_R;$
 $t \in \text{carrier } (\text{Vr } K v); vp K v = \text{Vr } K v \diamond_p t;$
 $\deg R (\text{Vr } K v) X g0 \leq \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0);$
 $\deg R (\text{Vr } K v) X h0 + \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X$
 S
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0)$
 $\leq \deg R (\text{Vr } K v) X f;$
 $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0);$
 $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0);$
 $\text{rel-prime-polS } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0)$
 $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0);$

 $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) f =$
 $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0 \cdot_{rS}$
 $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0 \rrbracket \implies$
 $\text{PCauchy } R X K v \text{ Hfst } K v R X t S Y f g0 h0$

apply (*frule Vr-integral[of v], frule vp-ideal[of v], frule Vr-ring, frule pj-Hom[of Vr K v vp K v], assumption+, simp add:PolynRg.erH-mult[THEN sym, of R Vr K v X S Vr K v /_r (Vr K v \diamond_p t) Y pj (Vr K v) (Vr K v \diamond_p t) g0 h0],*)

```

frule PolynRg.P-mod-diff[THEN sym, of R Vr K v X Vr K v  $\diamond_p$  t S Y
f g0  $\cdot_r R$  h0], assumption+,
frule PolynRg.is-Ring[of R], rule Ring.ring-tOp-closed,
assumption+)
apply (simp add:pol-Cauchy-seq-def, rule conjI)
apply (rule allI)

apply (frule-tac t = t and g = g0 and h = h0 and m = n in
PolynRg.P-mod-diffxxx5-2[of R Vr K v X - S Y f],
rule Vr-integral[of v], assumption+, simp add:vp-gen-nonzero,
drule sym, simp add:vp-maximal, assumption+)

apply (subst Hfst-def)
apply (rule cart-prod-fst, assumption)
apply (rule conjI)
apply (subgoal-tac  $\forall n.$  deg R (Vr K v) X (Hfst K v R X t S Y f g0 h0 n)  $\leq$ 
an (deg-n R (Vr K v) X f))
apply blast
apply (rule allI)
apply (frule Vr-integral[of v],
frule-tac t = t and g = g0 and h = h0 and m = n in
PolynRg.P-mod-diffxxx5-4[of R Vr K v X - S Y f], assumption+,
simp add:vp-gen-nonzero,
drule sym, simp add:vp-maximal, assumption+)
apply (subst PolynRg.deg-an[THEN sym], (erule conjE)+, assumption+)
apply (simp add:Hfst-def, (erule conjE)+)
apply (frule-tac i = deg R (Vr K v) X (fst (Hpr R (Vr K v) X t S Y f g0 h0 n)))
and
j = deg R (Vr K v) X g0 and k = deg S (Vr K v /r (Vr K v  $\diamond_p$  t)) Y
(erH R (Vr K v) X S (Vr K v /r (Vr K v  $\diamond_p$  t)) Y
(pj (Vr K v) (Vr K v  $\diamond_p$  t)) g0) in ale-trans, assumption+,
frule PolynRg.nonzero-deg-pos[of R Vr K v X h0], assumption+,
frule-tac x = 0 and y = deg R (Vr K v) X h0 and z = deg S
(Vr K v /r (Vr K v  $\diamond_p$  t)) Y (erH R (Vr K v) X S (Vr K v /r (Vr K v  $\diamond_p$  t)) Y
(pj (Vr K v) (Vr K v  $\diamond_p$  t)) g0) in aadd-le-mono, simp add:aadd-0-l)
apply (rule allI)
apply (subgoal-tac  $\forall n m.$  N < n  $\wedge$  N < m  $\longrightarrow$ 
P-mod R (Vr K v) X (Vr K v  $\diamond_p$  t (Vr K v) (an N))
( Hfst K v R X t S Y f g0 h0 n  $\pm_R$   $-a_R$  (Hfst K v R X t S Y f g0 h0 m)))
apply blast
apply ((rule allI)+, rule impI, (erule conjE)+,
frule Vr-integral[of v], frule vp-gen-nonzero[of v t], assumption+,
frule vp-maximal[of v])
apply (frule-tac t = t and g = g0 and h = h0 and m = n in
PolynRg.P-mod-diffxxx5-2[of R Vr K v X - S Y f], assumption+, simp,
assumption+,
frule-tac t = t and g = g0 and h = h0 and m = m in
PolynRg.P-mod-diffxxx5-2[of R Vr K v X - S Y f], assumption+, simp,
assumption+,

```

```

frule-tac  $t = t$  and  $g = g0$  and  $h = h0$  and  $m = N$  in
  PolynRg.P-mod-difxxx5-2[of  $R \ Vr K v X - S Y f$ ], assumption+, simp,
  assumption+,
frule-tac  $x = Hpr R (Vr K v) X t S Y f g0 h0 m$  in cart-prod-fst[of -
  carrier  $R$  carrier  $R$ ],
frule-tac  $x = Hpr R (Vr K v) X t S Y f g0 h0 N$  in cart-prod-fst[of -
  carrier  $R$  carrier  $R$ ],
frule-tac  $x = Hpr R (Vr K v) X t S Y f g0 h0 n$  in cart-prod-fst[of -
  carrier  $R$  carrier  $R$ ],
thin-tac  $Hpr R (Vr K v) X t S Y f g0 h0 n \in$  carrier  $R \times$  carrier  $R$ ,
thin-tac  $Hpr R (Vr K v) X t S Y f g0 h0 m \in$  carrier  $R \times$  carrier  $R$ )
apply (frule PolynRg.is-Ring)
apply (case-tac  $N = 0$ , simp add:r-apow-def)
apply (rule-tac  $p = Hfst K v R X t S Y f g0 h0 n \pm_R$ 
            $-_a R (Hfst K v R X t S Y f g0 h0 m)$  in
  PolynRg.P-mod-whole[of  $R \ Vr K v X$ ], assumption+,
  frule Ring.ring-is-ag[of  $R$ ], simp add:Hfst-def,
  rule aGroup.ag-pOp-closed, assumption+, rule aGroup.ag-mOp-closed,
  assumption+)

apply (frule-tac  $t = t$  and  $g = g0$  and  $h = h0$  and  $m = N$  and  $n = n - N$  in
  PolynRg.P-mod-difxxx5-3[of  $R \ Vr K v X - S Y f$ ], assumption+,
  simp+, (erule conjE)+,
frule-tac  $t = t$  and  $g = g0$  and  $h = h0$  and  $m = N$  and  $n = m - N$  in
  PolynRg.P-mod-difxxx5-3[of  $R \ Vr K v X - S Y f$ ], assumption+,
  simp, (erule conjE)+,
thin-tac P-mod  $R (Vr K v) X (Vr K v \diamond_p (t \setminus (Vr K v) N))$  (snd
  ( $Hpr R (Vr K v) X t S Y f g0 h0 N \pm_R -_a R (snd (Hpr R (Vr K v) X t S Y f g0 h0$ 
 $n)))$ ),
thin-tac P-mod  $R (Vr K v) X (Vr K v \diamond_p (t \setminus (Vr K v) N))$  (snd
  ( $Hpr R (Vr K v) X t S Y f g0 h0 N \pm_R -_a R (snd (Hpr R (Vr K v) X t S Y f g0 h0$ 
 $m)))$ )
apply (frule Vr-ring[of  $v$ ],
  simp only:Ring.principal-ideal-n-pow1[THEN sym],
  drule sym, simp, frule vp-ideal[of  $v$ ],
  simp add:Ring.ring-pow-apow,
  frule-tac  $n = an N$  in vp-apow-ideal[of  $v$ ], simp,
  frule Ring.ring-is-ag[of  $R$ ],
  frule-tac  $x = fst (Hpr R (Vr K v) X t S Y f g0 h0 n)$  in
    aGroup.ag-mOp-closed[of  $R$ ], assumption)
apply (frule-tac  $I = vp K v (Vr K v) (an N)$  and
   $p = (fst (Hpr R (Vr K v) X t S Y f g0 h0 N)) \pm_R$ 
   $-_a R (fst (Hpr R (Vr K v) X t S Y f g0 h0 n))$  in PolynRg.P-mod-minus[of  $R$ 
   $Vr K v X$ ], assumption+)

apply (rule aGroup.ag-pOp-closed, assumption+,
  simp add:aGroup.ag-p-inv aGroup.ag-inv-inv)

```

```

apply (frule Ring.ring-is-ag,
  frule-tac  $x = -_R \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 N)$  and
   $y = \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 n)$  in aGroup.ag-pOp-commute)
apply(rule aGroup.ag-mOp-closed, assumption+, simp,
  thin-tac  $-_R \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 N) \pm_R$ 
   $\text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 n) = \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0$ 
 $n) \pm_R$ 
 $-_R \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 N))$ 
apply (frule-tac  $I = vp K v (\text{Vr } K v) (\text{an } N)$  and
   $p = \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 n) \pm_R$ 
 $-_R \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 N)$  and
   $q = \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 N) \pm_R$ 
 $-_R \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 m)$  in PolynRg.P-mod-add[of
 $R \text{ Vr } K v X$ ], assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+,
  rule aGroup.ag-mOp-closed, assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+,
  rule aGroup.ag-mOp-closed, assumption+)
apply (frule-tac  $x = \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 N)$  in
  aGroup.ag-mOp-closed, assumption+,
  frule-tac  $x = \text{fst} (\text{Hpr}_R (\text{Vr } K v) X t S Y f g0 h0 m)$  in
  aGroup.ag-mOp-closed, assumption+,
  simp add:aGroup.pOp-assocTr43[of R] aGroup.ag-l-inv1 aGroup.ag-r-zero)
apply (simp add:Hfst-def)
done

lemma (in Corps) Hsnd-PCauchy: $\llbracket \text{valuation } K v; \text{Complete}_v K;$ 
 $\text{PolynRg } R (\text{Vr } K v) X; \text{PolynRg } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y; g0 \in \text{carrier } R;$ 
 $h0 \in \text{carrier } R; f \in \text{carrier } R; f \neq \mathbf{0}_R; g0 \neq \mathbf{0}_R; h0 \neq \mathbf{0}_R;$ 
 $t \in \text{carrier } (\text{Vr } K v); vp K v = \text{Vr } K v \diamond_p t;$ 
 $\deg R (\text{Vr } K v) X g0 \leq \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$ 
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (pj (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0);$ 
 $\deg R (\text{Vr } K v) X h0 + \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X$ 
 $S$ 
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (pj (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0)$ 
 $\leq \deg R (\text{Vr } K v) X f;$ 
 $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$ 
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (pj (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0);$ 
 $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$ 
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (pj (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0);$ 
 $\text{rel-prime-polS } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (\text{erH } R (\text{Vr } K v) X S$ 
 $(\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y (pj (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0)$ 
 $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
 $(pj (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0);$ 
 $\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
 $(pj (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) f =$ 

```

$$\begin{aligned}
& erH R (Vr K v) X S (Vr K v /_r (Vr K v \diamond_p t)) Y \\
& \quad (pj (Vr K v) (Vr K v \diamond_p t)) g0 \cdot_r S \\
& erH R (Vr K v) X S (Vr K v /_r (Vr K v \diamond_p t)) Y \\
& \quad (pj (Vr K v) (Vr K v \diamond_p t)) h0] \implies \\
& PCauchy R X K v Hsnd K v R X t S Y f g0 h0
\end{aligned}$$

apply (*frule Vr-integral[of v]*, *frule vp-ideal[of v]*,
frule Vr-ring, *frule pj-Hom[of Vr K v vp K v]*, *assumption+*,
simp add:PolynRg.erH-mult[THEN sym, of R Vr K v X S
 $Vr K v /_r (Vr K v \diamond_p t) Y pj (Vr K v) (Vr K v \diamond_p t) g0 h0],$
frule PolynRg.P-mod-diff[THEN sym, of R Vr K v X Vr K v \diamond_p t S Y
 $f g0 \cdot_r h0]$, *assumption+*,
frule PolynRg.is-Ring[of R], *rule Ring.ring-tOp-closed*,
assumption+)

apply (*simp add:pol-Cauchy-seq-def*, *rule conjI*)
apply (*rule allI*)

apply (*frule-tac t = t and g = g0 and h = h0 and m = n in*
PolynRg.P-mod-diffxxx5-2[of R Vr K v X - S Y f],
rule Vr-integral[of v], *assumption+*, *simp add:vp-gen-nonzero*,
drule sym, *simp add:vp-maximal, assumption+*)

apply (*subst Hsnd-def*)
apply (*rule cart-prod-snd*, *assumption*)
apply (*rule conjI*)
apply (*subgoal-tac $\forall n. \deg R (Vr K v) X (Hsnd K v R X t S Y f g0 h0 n) \leq$*
 $an (\deg-n R (Vr K v) X f))$)

apply *blast*
apply (*rule allI*)
apply (*frule Vr-integral[of v]*,
frule-tac t = t and g = g0 and h = h0 and m = n in
PolynRg.P-mod-diffxxx5-4[of R Vr K v X - S Y f], *assumption+*,
simp add:vp-gen-nonzero,
drule sym, *simp add:vp-maximal, assumption+*)

apply (*subst PolynRg.deg-an[THEN sym]*, (*erule conjE*)+, *assumption+*)

apply (*simp add:Hsnd-def*)
apply (*rule allI*)
apply (*subgoal-tac $\forall n m. N < n \wedge N < m \longrightarrow$*
 $P\text{-mod } R (Vr K v) X (Vr K v \diamond_p t (Vr K v) (an N))$
 $(Hsnd K v R X t S Y f g0 h0 n \pm_R -a R (Hsnd K v R X t S Y f g0 h0 m)))$)

apply *blast*
apply ((*rule allI*)+, *rule impI*, (*erule conjE*)+)
apply (*frule Vr-integral[of v]*, *frule vp-gen-nonzero[of v t]*, *assumption+*)
apply (*frule vp-maximal[of v]*)
apply (*frule-tac t = t and g = g0 and h = h0 and m = n in*
PolynRg.P-mod-diffxxx5-2[of R Vr K v X - S Y f], *assumption+, simp*,
assumption+)

apply (*frule-tac t = t and g = g0 and h = h0 and m = m in*

$\text{PolynRg.P-mod-difxxx5-2}[\text{of } R \text{ Vr } K v X - S Y f]$, assumption+ , simp ,
 assumption+ ,
 $\text{frule-tac } t = t \text{ and } g = g0 \text{ and } h = h0 \text{ and } m = N \text{ in}$
 $\text{PolynRg.P-mod-difxxx5-2}[\text{of } R \text{ Vr } K v X - S Y f]$, assumption+ , simp ,
 assumption+ ,
 $\text{frule-tac } x = Hpr_R (\text{Vr } K v) X t S Y f g0 h0 m \text{ in } \text{cart-prod-snd}[\text{of } -$
 $\text{carrier } R \text{ carrier } R]$,
 $\text{frule-tac } x = Hpr_R (\text{Vr } K v) X t S Y f g0 h0 N \text{ in } \text{cart-prod-snd}[\text{of } -$
 $\text{carrier } R \text{ carrier } R]$,
 $\text{frule-tac } x = Hpr_R (\text{Vr } K v) X t S Y f g0 h0 n \text{ in } \text{cart-prod-snd}[\text{of } -$
 $\text{carrier } R \text{ carrier } R]$,
 $\text{thin-tac } Hpr_R (\text{Vr } K v) X t S Y f g0 h0 n \in \text{carrier } R \times \text{carrier } R$,
 $\text{thin-tac } Hpr_R (\text{Vr } K v) X t S Y f g0 h0 m \in \text{carrier } R \times \text{carrier } R)$
apply ($\text{frule PolynRg.is-Ring}$)
apply ($\text{case-tac } N = 0$, $\text{simp add:r-apow-def}$)
apply ($\text{rule-tac } p = \text{Hsnd}_{K v R X t S Y f g0 h0 n \pm_R}$
 $-_{aR} (\text{Hsnd}_{K v R X t S Y f g0 h0 m}) \text{ in}$
 $\text{PolynRg.P-mod-whole}[\text{of } R \text{ Vr } K v X]$, assumption+ ,
 $\text{frule Ring.ring-is-ag}[\text{of } R]$, simp add:Hsnd-def ,
 $\text{rule aGroup.ag-pOp-closed}$, assumption+ , $\text{rule aGroup.ag-mOp-closed}$,
 $\text{assumption+})$

apply ($\text{frule-tac } t = t \text{ and } g = g0 \text{ and } h = h0 \text{ and } m = N \text{ and } n = n - N \text{ in}$
 $\text{PolynRg.P-mod-difxxx5-3}[\text{of } R \text{ Vr } K v X - S Y f]$, $\text{assumption+})$
apply simp+
apply (erule conjE)
apply ($\text{frule-tac } t = t \text{ and } g = g0 \text{ and } h = h0 \text{ and } m = N \text{ and } n = m - N \text{ in}$
 $\text{PolynRg.P-mod-difxxx5-3}[\text{of } R \text{ Vr } K v X - S Y f]$, $\text{assumption+})$
apply $\text{simp apply (erule conjE)}$
apply ($\text{thin-tac P-mod } R (\text{Vr } K v) X (\text{Vr } K v \diamond_p (t \wedge (\text{Vr } K v) N)) (\text{fst}$
 $(Hpr_R (\text{Vr } K v) X t S Y f g0 h0 N) \pm_R -_{aR} (\text{fst} (Hpr_R (\text{Vr } K v) X t S Y f g0 h0$
 $n)))$,
 $\text{thin-tac P-mod } R (\text{Vr } K v) X (\text{Vr } K v \diamond_p (t \wedge (\text{Vr } K v) N)) (\text{fst}$
 $(Hpr_R (\text{Vr } K v) X t S Y f g0 h0 N) \pm_R -_{aR} (\text{fst} (Hpr_R (\text{Vr } K v) X t S Y f g0 h0$
 $m)))$)
apply ($\text{frule Vr-ring}[of v]$)
apply ($\text{simp only:Ring.principal-ideal-n-pow1}[\text{THEN sym}]$)
apply ($\text{drule sym, simp, frule vp-ideal}[of v]$)
apply ($\text{simp add:Ring.ring-pow-apow}$,
 $\text{frule-tac } n = an N \text{ in } \text{vp-apow-ideal}[\text{of } v]$, simp ,
 $\text{frule Ring.ring-is-ag}[\text{of } R]$,
 $\text{frule-tac } x = \text{snd} (Hpr_R (\text{Vr } K v) X t S Y f g0 h0 n) \text{ in}$
 $\text{aGroup.ag-mOp-closed}[\text{of } R]$, $\text{assumption})$
apply ($\text{frule-tac } I = \text{vp } K v (\text{Vr } K v) (an N) \text{ and}$
 $p = (\text{snd} (Hpr_R (\text{Vr } K v) X t S Y f g0 h0 N)) \pm_R$
 $-_{aR} (\text{snd} (Hpr_R (\text{Vr } K v) X t S Y f g0 h0 n)) \text{ in } \text{PolynRg.P-mod-minus}[\text{of } R$
 $\text{Vr } K v X]$, $\text{assumption+})$

```

apply (rule aGroup.ag-pOp-closed, assumption+
      simp add:aGroup.ag-p-inv aGroup.ag-inv-inv)
apply (frule Ring.ring-is-ag,
      frule-tac  $x = -_a R \text{ snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 N)$  and
       $y = \text{snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 n)$  in aGroup.ag-pOp-commute)
apply (rule aGroup.ag-mOp-closed, assumption+, simp,
      thin-tac  $-_a R \text{ snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 N) \pm_R$ 
       $\text{snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 n) = \text{snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0$ 
       $n) \pm_R$ 
       $-_a R \text{ snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 N))$ 
apply (frule-tac  $I = vp K v (\text{Vr } K v) (an N)$  and
       $p = \text{snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 n) \pm_R$ 
       $-_a R \text{ snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 N)$  and
       $q = \text{snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 N) \pm_R$ 
       $-_a R \text{ snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 m)$  in PolynRg.P-mod-add[of
       $R \text{ Vr } K v X], assumption+$ )
apply (rule aGroup.ag-pOp-closed, assumption+,
      rule aGroup.ag-mOp-closed, assumption+)
apply (rule aGroup.ag-pOp-closed, assumption+,
      rule aGroup.ag-mOp-closed, assumption+)
apply (frule-tac  $x = \text{snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 N)$  in
      aGroup.ag-mOp-closed, assumption+,
      frule-tac  $x = \text{snd } (\text{Hpr } R (\text{Vr } K v) X t S Y f g0 h0 m)$  in
      aGroup.ag-mOp-closed, assumption+,
      simp add:aGroup.pOp-assocTr43[R] aGroup.ag-l-inv1 aGroup.ag-r-zero)
apply (simp add:Hsnd-def)
done

lemma (in Corps) H-Plimit-f:[valuation K v; Completev K;
   $t \in \text{carrier } (\text{Vr } K v); vp K v = \text{Vr } K v \diamond_p t;$ 
   $\text{PolynRg } R (\text{Vr } K v) X; \text{PolynRg } S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y;$ 
   $f \in \text{carrier } R; f \neq \mathbf{0}_R; g0 \in \text{carrier } R; h0 \in \text{carrier } R; g0 \neq \mathbf{0}_R;$ 
   $h0 \neq \mathbf{0}_R;$ 
   $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
   $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
   $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0);$ 
   $0 < \deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
   $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
   $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) h0);$ 
   $\deg R (\text{Vr } K v) X h0 +$ 
   $\deg S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
   $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
   $(\text{pj } (\text{Vr } K v) (\text{Vr } K v \diamond_p t)) g0) \leq \deg R (\text{Vr } K v) X f;$ 
  rel-prime-pol S ( $\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 
   $(\text{erH } R (\text{Vr } K v) X S (\text{Vr } K v /_r (\text{Vr } K v \diamond_p t)) Y$ 

```

$$\begin{aligned}
& (pj (Vr K v) (Vr K v \diamond_p t)) g0 \\
(erH R (Vr K v) X S (Vr K v /_r (Vr K v \diamond_p t)) Y & \\
(pj (Vr K v) (Vr K v \diamond_p t)) h0; \\
erH R (Vr K v) X S (Vr K v /_r (Vr K v \diamond_p t)) Y & \\
(pj (Vr K v) (Vr K v \diamond_p t)) f = \\
erH R (Vr K v) X S (Vr K v /_r (Vr K v \diamond_p t)) Y & \\
(pj (Vr K v) (Vr K v \diamond_p t)) g0 \cdot_{rS} \\
erH R (Vr K v) X S (Vr K v /_r (Vr K v \diamond_p t)) Y & \\
(pj (Vr K v) (Vr K v \diamond_p t)) h0; \\
\deg R (Vr K v) X g0 & \\
\leq \deg S (Vr K v /_r (Vr K v \diamond_p t)) Y & \\
(erH R (Vr K v) X S (Vr K v /_r (Vr K v \diamond_p t)) Y & \\
(pj (Vr K v) (Vr K v \diamond_p t)) g0); \\
g \in carrier R; h \in carrier R; & \\
Plimit R X K v (Hfst K v R X t S Y f g0 h0) g; & \\
Plimit R X K v (Hsnd K v R X t S Y f g0 h0) h; & \\
Plimit R X K v (\lambda n. (Hfst K v R X t S Y f g0 h0 n) \cdot_{rR} & \\
(Hsnd K v R X t S Y f g0 h0 n)) (g \cdot_{rR} h) \llbracket & \\
\implies Plimit R X K v (\lambda n. (Hfst K v R X t S Y f g0 h0 n) \cdot_{rR} & \\
(Hsnd K v R X t S Y f g0 h0 n)) f & \\
\text{apply}(frule Vr-integral[of v], frule vp-ideal[of v], & \\
frule Vr-ring, frule pj-Hom[of Vr K v vp K v], assumption+, & \\
simp add:PolynRg.erH-mult[THEN sym, of R Vr K v X S & \\
Vr K v /_r (Vr K v \diamond_p t) Y pj (Vr K v) (Vr K v \diamond_p t) g0 h0]) & \\
\text{apply}(frule PolynRg.P-mod-diff[of R Vr K v X Vr K v \diamond_p t S Y & \\
f g0 \cdot_{rR} h0], assumption+, & \\
frule PolynRg.is-Ring[of R], rule Ring.ring-tOp-closed, & \\
assumption+, simp) & \\
\text{apply } (simp add:PolynRg.erH-mult[of R Vr K v X S & \\
Vr K v /_r (Vr K v \diamond_p t) Y pj (Vr K v) (Vr K v \diamond_p t) g0 h0]) & \\
\text{apply } (frule PolynRg.is-Ring[of R]) & \\
\text{apply } (frule Hfst-PCauchy[of v R X S t Y g0 h0 f], assumption+, & \\
frule Hsnd-PCauchy[of v R X S t Y g0 h0 f], assumption+) & \\
\text{apply } (subst pol-limit-def) & \\
\text{apply } (rule conjI) & \\
\text{apply } (rule allI) & \\
\text{apply } (rule Ring.ring-tOp-closed, assumption) & \\
\text{apply } (simp add:pol-Cauchy-seq-def, simp add:pol-Cauchy-seq-def) & \\
\text{apply } (rule allI) & \\
\text{apply } (subgoal-tac \forall m>N. P-mod R (Vr K v) X (vp K v (Vr K v) (an N)) & \\
((Hfst K v R X t S Y f g0 h0 m) \cdot_{rR} (Hsnd K v R X t S Y f g0 h0 m) & \\
\pm_R \neg_a R f)) & \\
\text{apply } blast & \\
\text{apply } (rule allI, rule impI, frule Vr-integral[of v]) & \\
\text{apply } (frule-tac t = t \text{ and } g = g0 \text{ and } h = h0 \text{ and } m = m - Suc 0 \text{ in } &
\end{aligned}$$

$\text{PolynRg.P-mod-difffxxx5-1}[\text{of } R \text{ Vr } K \text{ v } X - S \text{ Y}], \text{ assumption+},$
 $\text{simp add:vp-gen-nonzero[of v]},$
 $\text{frule vp-maximal[of v], simp, assumption+})$
apply ((erule conjE)+, simp del:npow-suc Hpr-Suc)
apply (frule Ring.ring-is-ag[of R])
apply (thin-tac $0 < \deg S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) g0),$
 $\text{thin-tac } 0 < \deg S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) h0),$
 $\text{thin-tac rel-prime-polS } S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) g0)$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) h0),$
 $\text{thin-tac P-mod } R (\text{Vr } K \text{ v }) X (\text{Vr } K \text{ v } \diamond_p t) (f \pm_R -_{aR} g0 \cdot_{rR} h0),$
 $\text{thin-tac erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) f =$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) g0) \cdot_{rS}$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) h0),$
 $\text{thin-tac deg } R (\text{Vr } K \text{ v }) X g0 \leq \deg S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) g0),$
 $\text{thin-tac deg } R (\text{Vr } K \text{ v }) X h0 + \deg S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) g0) \leq \deg R (\text{Vr } K \text{ v }) X f,$
 $\text{thin-tac erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) (\text{fst } (\text{Hpr } R (\text{Vr } K \text{ v }) X t S Y f g0 h0 m)) =$
 $\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) g0,$
 $\text{thin-tac erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) (\text{snd } (\text{Hpr } R (\text{Vr } K \text{ v }) X t S Y f g0 h0 m)) =$
 $\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) h0,$
 $\text{thin-tac deg } R (\text{Vr } K \text{ v }) X (\text{fst } (\text{Hpr } R (\text{Vr } K \text{ v }) X t S Y f g0 h0 m))$
 $\leq \deg S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) g0),$
 $\text{thin-tac P-mod } R (\text{Vr } K \text{ v }) X (\text{Vr } K \text{ v } \diamond_p (t \wedge (\text{Vr } K \text{ v } m)))$
 $(\text{fst } (\text{Hpr } R (\text{Vr } K \text{ v }) X t S Y f g0 h0 (m - \text{Suc } 0))) \pm_R$
 $-_{aR} (\text{fst } (\text{Hpr } R (\text{Vr } K \text{ v }) X t S Y f g0 h0 m)),$
 $\text{thin-tac deg } R (\text{Vr } K \text{ v }) X (\text{snd } (\text{Hpr } R (\text{Vr } K \text{ v }) X t S Y f g0 h0 m)) +$
 $\deg S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{erH } R (\text{Vr } K \text{ v }) X S (\text{Vr } K \text{ v } /_r (\text{Vr } K \text{ v } \diamond_p t)) Y$
 $(\text{pj } (\text{Vr } K \text{ v }) (\text{Vr } K \text{ v } \diamond_p t)) g0) \leq \deg R (\text{Vr } K \text{ v }) X f,$

```

thin-tac P-mod R (Vr K v) X (Vr K v  $\diamond_p$  (t  $\prec$  Vr K v) m))
  (snd (Hpr R (Vr K v) X t S Yf g0 h0 (m - Suc 0))  $\pm_R$ 
   - $_aR$  (snd (Hpr R (Vr K v) X t S Yf g0 h0 m))))
apply (case-tac N = 0, simp add:r-apow-def)
apply (rule-tac p = (Hfst K v R X t S Yf g0 h0 m)  $\cdot_r R$  (Hsnd K v R X t S Yf g0 h0
m)
       $\pm_R$  - $_aR$  f in PolynRg.P-mod-whole[of R Vr K v X], assumption+)
apply (simp add:Hfst-def Hsnd-def)
apply (frule-tac x = Hpr R (Vr K v) X t S Yf g0 h0 m in cart-prod-fst[of - carrier
R carrier R])
apply (frule-tac x = Hpr R (Vr K v) X t S Yf g0 h0 m in cart-prod-snd[of -
carrier R carrier R])
apply (frule Ring.ring-is-ag[of R],
       rule aGroup.ag-pOp-closed, assumption)
apply (rule Ring.ring-tOp-closed, assumption+)
apply (rule aGroup.ag-mOp-closed, assumption+)

apply (frule-tac g = f  $\pm_R$  - $_aR$  (fst (Hpr R (Vr K v) X t S Yf g0 h0 m))  $\cdot_r R$ 
      (snd (Hpr R (Vr K v) X t S Yf g0 h0 m)) and m = N - Suc 0 and n = m
in
  PolynRg.P-mod-n-m[of R Vr K v X], assumption+)
apply (frule-tac x = Hpr R (Vr K v) X t S Yf g0 h0 m in cart-prod-fst[of - carrier
R carrier R],
       frule-tac x = Hpr R (Vr K v) X t S Yf g0 h0 m in cart-prod-snd[of - carrier
R carrier R])

apply (rule aGroup.ag-pOp-closed, assumption+, rule aGroup.ag-mOp-closed,
       assumption)
apply (rule Ring.ring-tOp-closed, assumption+)
apply (subst Suc-le-mono[THEN sym], simp)
apply assumption
apply (simp del:npow-suc)
apply (simp only:Ring.principal-ideal-n-pow1[THEN sym, of Vr K v])
apply (cut-tac n = N in an-neq-inf)
apply (subgoal-tac an N  $\neq$  0)
apply (subst r-apow-def, simp) apply (simp add:na-an)
apply (frule Ring.principal-ideal[of Vr K v t], assumption)
apply (frule-tac I = Vr K v  $\diamond_p$  t and n = N in Ring.ideal-pow-ideal[of Vr K v],
assumption+)
apply (frule-tac x = Hpr R (Vr K v) X t S Yf g0 h0 m in cart-prod-fst[of - carrier
R carrier R],
       frule-tac x = Hpr R (Vr K v) X t S Yf g0 h0 m in cart-prod-snd[of - carrier
R carrier R])

apply (thin-tac PCauchy R X K v Hfst K v R X t S Yf g0 h0,
       thin-tac PCauchy R X K v Hsnd K v R X t S Yf g0 h0)
apply (simp add:Hfst-def Hsnd-def)

```

```

apply (frule-tac  $x = fst (Hpr R (Vr K v) X t S Y f g0 h0 m)$  and  $y = snd (Hpr R (Vr K v) X t S Y f g0 h0 m)$  in Ring.ring-tOp-closed[of  $R$ ], assumption+)

apply (frule-tac  $x = (fst (Hpr R (Vr K v) X t S Y f g0 h0 m)) \cdot_r R (snd (Hpr R (Vr K v) X t S Y f g0 h0 m))$  in aGroup.ag-mOp-closed[of  $R$ ], assumption+)
apply (frule-tac  $I = Vr K v \diamond_p t \diamond (Vr K v) N$  and
 $p = f \pm_R -_a R (fst (Hpr R (Vr K v) X t S Y f g0 h0 m)) \cdot_r R$ 
 $(snd (Hpr R (Vr K v) X t S Y f g0 h0 m))$  in
PolynRg.P-mod-minus[of  $R$  Vr  $K v X$ ], assumption+)
apply (frule Ring.ring-is-ag[of  $R$ ])
apply (rule aGroup.ag-pOp-closed, assumption+)
apply (simp add:aGroup.ag-p-inv, simp add:aGroup.ag-inv-inv,
frule aGroup.ag-mOp-closed[of  $R f$ ], assumption+)
apply (simp add:aGroup.ag-pOp-commute[of  $R -_a R f$ ])
apply (subst an-0[THEN sym])
apply (subst aneq-natneq[of - 0], thin-tac an  $N \neq \infty$ , simp)
done

theorem (in Corps) Hensel: valuation  $K v$ ; Complete $_v K$ ;
PolynRg  $R (Vr K v) X$ ; PolynRg  $S ((Vr K v) /_r (vp K v)) Y$ ;
 $f \in carrier R$ ;  $f \neq \mathbf{0}_R$ ;  $g' \in carrier S$ ;  $h' \in carrier S$ ;
 $0 < deg S ((Vr K v) /_r (vp K v)) Y g'$ ;
 $0 < deg S ((Vr K v) /_r (vp K v)) Y h'$ ;
 $((erH R (Vr K v) X S ((Vr K v) /_r (vp K v)) Y$ 
 $(pj (Vr K v) (vp K v))) f) = g' \cdot_r S h'$ ;
rel-prime-pols  $S ((Vr K v) /_r (vp K v)) Y g' h' \Rightarrow$ 
 $\exists g h. g \in carrier R \wedge h \in carrier R \wedge$ 
 $deg R (Vr K v) X g \leq deg S ((Vr K v) /_r (vp K v)) Y g' \wedge$ 
 $f = g \cdot_r R h$ 
apply (frule PolynRg.is-Ring[of  $R$  Vr  $K v X$ ],
frule PolynRg.is-Ring[of  $S$  Vr  $K v /_r vp K v Y$ ],
frule vp-gen-t[of  $v$ ], erule bxE,
frule-tac  $t = t$  in vp-gen-nonzero[of  $v$ ], assumption)
apply (frule-tac  $t = t$  in Hensel-starter[of  $v R X S Y - f g' h'$ ], assumption+)
apply ((erule exE)+, (erule conjE)+, rename-tac  $g0 h0$ )
apply (frule Vr-ring[of  $v$ ], frule Vr-integral[of  $v$ ])
apply (rotate-tac 22, drule sym, drule sym, simp)
apply (frule vp-maximal[of  $v$ ], simp)
apply (frule-tac  $mx = Vr K v \diamond_p t$  in Ring.residue-field-cd[of  $Vr K v$ ],
assumption)
apply (frule-tac  $mx = Vr K v \diamond_p t$  in Ring.maximal-ideal-ideal[of  $Vr K v$ ],
assumption)
apply (frule-tac  $I = Vr K v \diamond_p t$  in Ring.qring-ring[of  $Vr K v$ ],
assumption+)
apply (frule-tac  $B = Vr K v /_r (Vr K v \diamond_p t)$  and  $h = pj (Vr K v) (Vr K v \diamond_p t)$  in PolynRg.erH-rHom[of  $R$  Vr  $K v X S - Y$ ], assumption+)
apply (simp add:pj-Hom)
apply (frule-tac  $?g0.0 = g0$  and  $?h0.0 = h0$  in Hfst-PCauchy[of  $v R X S - Y -$ 

```

```

f], assumption+)
apply (frule-tac ?g0.0 = g0 and ?h0.0 = h0 in Hsnd-PCauchy[of v R X S - Y -
f], assumption+)
apply (frule-tac F = λj. Hfst_K v R X t S Y f g0 h0 j in PCauchy-Plimit[of v R X]
, assumption+)
apply (frule-tac F = λj. Hsnd_K v R X t S Y f g0 h0 j in PCauchy-Plimit[of v R X]
, assumption+)
apply ((erule bexE)+, rename-tac g0 h0 g h)

apply (frule-tac F = λj. Hfst_K v R X t S Y f g0 h0 j and
G = λj. Hsnd_K v R X t S Y f g0 h0 j and ?p1.0 = g and ?p2.0 = h
in P-limit-mult[of v R X], assumption+, rule allI)

apply (simp add:pol-Cauchy-seq-def)
apply (simp add:pol-Cauchy-seq-def, assumption+)
apply (frule-tac t = t and ?g0.0 = g0 and ?h0.0 = h0 and g = g and h = h
in
H-Plimit-f[of v - R X S Y f], assumption+)
apply (frule-tac F = λn. (Hfst_K v R X t S Y f g0 h0 n) ·_R (Hsnd_K v R X t S Y f g0 h0
n) and ?p1.0 = g ·_R h and d = na (deg R (Vr K v) X g0 + deg R (Vr K v) X
f) in P-limit-unique[of v R X - - - f], assumption+)
apply (rule allI)
apply (frule PolynRg.is-Ring[of R],
rule Ring.ring-tOp-closed, assumption)
apply (simp add:pol-Cauchy-seq-def)
apply (simp add:pol-Cauchy-seq-def)
apply (rule allI)
apply (thin-tac Plimit_R X K v (Hfst_K v R X t S Y f g0 h0) g,
thin-tac Plimit_R X K v (Hsnd_K v R X t S Y f g0 h0) h,
thin-tac Plimit_R X K v (λn. (Hfst_K v R X t S Y f g0 h0 n) ·_R
(Hsnd_K v R X t S Y f g0 h0 n)) (g ·_R h),
thin-tac Plimit_R X K v (λn. (Hfst_K v R X t S Y f g0 h0 n) ·_R
(Hsnd_K v R X t S Y f g0 h0 n)) f)
apply (subst PolynRg.deg-mult-pols1[of R Vr K v X], assumption+)

apply (simp add:pol-Cauchy-seq-def, simp add:pol-Cauchy-seq-def,
thin-tac g' = erH R (Vr K v) X S (Vr K v /_r (Vr K v ◇_p t)) Y
(pj (Vr K v) (Vr K v ◇_p t)) g0,
thin-tac h' = erH R (Vr K v) X S (Vr K v /_r (Vr K v ◇_p t)) Y
(pj (Vr K v) (Vr K v ◇_p t)) h0)
apply (subst PolynRg.deg-mult-pols1[THEN sym, of R Vr K v X], assumption+)
apply (simp add:pol-Cauchy-seq-def, simp add:pol-Cauchy-seq-def)

apply (frule-tac p1 = g0 in PolynRg.deg-mult-pols1[THEN sym, of R Vr K v X -
f], assumption+, simp)
apply (frule-tac x = g0 in Ring.ring-tOp-closed[of R - f], assumption+)

```

```

apply (frule-tac p = g0 ·_R f in PolynRg.nonzero-deg-pos[of R Vr K v X],
      assumption+)
apply (frule-tac p = g0 ·_R f in PolynRg.deg-in-aug-minf[of R Vr K v X],
      assumption+, simp add:aug-minf-def,
      simp add:PolynRg.polyn-ring-integral[of R Vr K v X],
      simp add:Idomain.idom-tOp-nonzeros[of R - f],
      frule-tac p = g0 ·_R f in PolynRg.deg-noninf[of R Vr K v X],
      assumption+)
apply (simp add:an-na)
apply (subst PolynRg.deg-mult-pols1[of R Vr K v X], assumption+,
      simp add:pol-Cauchy-seq-def, simp add:pol-Cauchy-seq-def)
apply (frule-tac t = t and g = g0 and h = h0 and m = n in
      PolynRg.P-mod-difffxx5-4[of R Vr K v X - S Y f], assumption+)
apply (erule conjE)

apply (frule-tac a = deg R (Vr K v) X (fst (Hpr R (Vr K v) X t S Y f g0 h0 n))
      and a' = deg R (Vr K v) X g0 and b = deg R (Vr K v) X (snd (Hpr R (Vr K v) X t S Y f g0 h0 n)) and b' = deg R (Vr K v) X f in aadd-plus-le-plus, assumption+)
apply simp
apply (simp add:Hfst-def Hsnd-def)

apply (rule Ring.ring-tOp-closed, assumption+)
apply (rotate-tac -1, drule sym)
apply (frule-tac F = λj. Hfst_K v R X t S Y f g0 h0 j and p = g and ad = deg S
      (Vr K v /_r (Vr K v ◇_p t)) Y (erH R (Vr K v) X S (Vr K v /_r (Vr K v ◇_p t)) Y
      (pj (Vr K v) (Vr K v ◇_p t)) g0) in Plimit-deg1[of v R X], assumption+,
      simp add:pol-Cauchy-seq-def)
apply (rule allI)
apply (frule-tac t = t and g = g0 and h = h0 and m = n in
      PolynRg.P-mod-difffxx5-4[of R Vr K v X - S Y f], assumption+,
      erule conjE)
apply (frule-tac i = deg R (Vr K v) X (fst (Hpr R (Vr K v) X t S Y f g0 h0 n))
      and
      j = deg R (Vr K v) X g0 and k = deg S (Vr K v /_r (Vr K v ◇_p t)) Y
      (erH R (Vr K v) X S (Vr K v /_r (Vr K v ◇_p t)) Y
      (pj (Vr K v) (Vr K v ◇_p t)) g0) in ale-trans, assumption+)
apply (subst Hfst-def, assumption+, blast)
done

end

```