

Computational p -Adics

Jeremy Sylvestre University of Alberta (Augustana Campus)

February 6, 2026

Abstract

We develop a framework for reasoning computationally about p -adic fields via formal Laurent series with integer coefficients. We define a ring-class quotient type consisting of equivalence classes of prime-indexed sequences of formal Laurent series, in which every p -adic field is contained as a subring. Via a further quotient of the ring of p -adic integers (that is, the elements of depth at least 0) by the prime ideal (that is, the subring of elements of depth at least 1), we define a ring-class type in which every prime-order finite field is contained as a subring. Finally, some topological properties are developed using depth as a proxy for a metric, Hensel's Lemma is proved, and the compactness of local rings of integers is established.

Contents

1	Distinguished Algebraic Quotients	4
1.1	Quotient groups	4
1.1.1	Class definitions	4
1.1.2	The quotient group type	5
1.2	Quotient rings	7
1.2.1	Class definitions	7
1.2.2	The quotient ring type	7
1.3	Quotients of commutative rings	8
1.3.1	Class definitions	8
1.3.2	Instances and instantiations	9
2	Additional facts about polynomials	10
2.1	General facts	10
2.2	Derivatives	11
2.3	Additive polynomials	11
3	Common structures for adelic types	13
3.1	Preliminaries	13
3.2	Existence of primes	14

3.3	Generic algebraic equality	15
3.4	Indexed equality	19
3.4.1	General structure	19
3.5	Indexed depth functions	28
3.5.1	General structure	28
3.5.2	Depth of inverses and division	37
3.6	Global equality and restriction of places	38
3.6.1	Locales	38
3.6.2	Embedding of base type constants	47
3.7	Topological patterns	68
3.7.1	By place	68
3.7.2	Globally	78
3.8	Notation	83
4	Equivalence of sequences of formal Laurent series relative to divisibility of partial sums by a fixed prime	84
4.1	Preliminaries	84
4.1.1	Lift/transfer of equivalence relations.	84
4.1.2	An additional fact about products/sums	85
4.1.3	An additional fact about algebraic powers of functions	85
4.1.4	Some additional facts about formal power and Laurent series	85
4.2	Partial evaluation of formal power series	86
4.3	Vanishing of formal power series relative to divisibility of all partial sums	88
4.4	Equivalence and depth of formal Laurent series relative to a prime	89
4.4.1	Definition of equivalence and basic facts	89
4.4.2	Depth as maximum subdegree of equivalent series	93
4.5	Reduction relative to a prime	96
4.5.1	Of scalars	96
4.5.2	Inductive partial reduction of formal Laurent series	97
4.5.3	Full reduction of formal Laurent series	98
4.5.4	Algebraic properties of reduction	100
4.6	Inversion relative to a prime	102
4.6.1	Of scalars	102
4.6.2	Inductive partial inversion of formal Laurent series	103
4.6.3	Full inversion of formal Laurent series	106
4.6.4	Algebraic properties of inversion	108
4.7	Topology by place relative to indexed depth for formal Laurent series	109
4.7.1	General pattern for constructing sequence limits	109
4.7.2	Completeness	110
4.8	Transfer to prime-indexed sequences of formal Laurent series	110

4.8.1	Equivalence and depth	110
4.8.2	Inversion	114
4.8.3	Topology via global bounded depth	117
5	p-Adic fields as equivalence classes of sequences of formal Laurent series	117
5.1	Preliminaries	117
5.2	The type definition as a quotient	118
5.3	Algebraic instantiations	119
5.4	Equivalence and depth relative to a prime	120
5.5	Embeddings of the base ring, <i>nat</i> , <i>int</i> , and <i>rat</i>	127
5.5.1	Embedding of the base ring	127
5.5.2	Embedding of the field of fractions of the base ring	127
5.5.3	Characteristic zero embeddings	128
5.6	Topologies on products of p-adic fields	129
5.6.1	By local place	129
5.6.2	By bounded global depth	131
5.7	Hiding implementation details	134
5.8	The subring of adelic integers	134
5.8.1	Type definition as a subtype of adeles	134
5.8.2	Algebraic instantiaions	135
5.8.3	Equivalence and depth relative to a prime	136
5.8.4	Inverses	141
5.8.5	The ideal of primes	145
5.8.6	Topology p-open neighbourhoods	147
5.8.7	Topology via global depth	149
6	Finite fields of prime order as quotients of the ring of integers of p-adic fields	153
6.1	The type	153
6.1.1	The prime ideal as the distinguished ideal of the ring of adelic integers	153
6.1.2	The finite field product type as a quotient	153
6.1.3	Algebraic instantiations	154
6.2	Equivalence relative to a prime	156
6.2.1	Equivalence	156
6.2.2	Embedding of constants	159
6.2.3	Division and inverses	160
6.3	Special case of integer	160
7	Compactness of local ring of integers	161
7.1	Preliminaries	161
7.2	Sequential compactness	161
7.2.1	Creating a Cauchy subsequence	161

7.2.2	Proving sequential compactness	163
7.3	Finite-open-cover compactness	164

theory *Distinguished-Quotients*

imports *Main*

begin

1 Distinguished Algebraic Quotients

We define quotient groups and quotient rings modulo distinguished normal subgroups and ideals, respectively, leading to the construction of a field as a commutative ring with 1 modulo a maximal ideal.

class *distinguished-subset* =
fixes *distinguished-subset* :: 'a set (\mathcal{N})

hide-const *distinguished-subset*

1.1 Quotient groups

1.1.1 Class definitions

Here we set up subclasses of *group-add* with a distinguished subgroup.

class *group-add-with-distinguished-subgroup* = *group-add* + *distinguished-subset* +
assumes *distset-nonempty*: $\mathcal{N} \neq \{\}$
and *distset-diff-closed*:
 $g \in \mathcal{N} \implies h \in \mathcal{N} \implies g - h \in \mathcal{N}$
begin

lemma *distset-zero-closed* : $0 \in \mathcal{N}$
 \langle *proof* \rangle

lemma *distset-uminus-closed*: $a \in \mathcal{N} \implies -a \in \mathcal{N}$
 \langle *proof* \rangle

lemma *distset-add-closed*:
 $a \in \mathcal{N} \implies b \in \mathcal{N} \implies a + b \in \mathcal{N}$
 \langle *proof* \rangle

lemma *distset-uminus-plus-closed*:
 $a \in \mathcal{N} \implies b \in \mathcal{N} \implies -a + b \in \mathcal{N}$
 \langle *proof* \rangle

lemma *distset-lcoset-closed1*:

$a \in \mathcal{N} \implies -a + b \in \mathcal{N} \implies b \in \mathcal{N}$
 ⟨proof⟩

lemma *distset-lcoset-closed2*:

$b \in \mathcal{N} \implies -a + b \in \mathcal{N} \implies a \in \mathcal{N}$
 ⟨proof⟩

lemma *reflp-lcosetrel*: $-x + x \in \mathcal{N}$

⟨proof⟩

lemma *symp-lcosetrel*: $-a + b \in \mathcal{N} \implies -b + a \in \mathcal{N}$

⟨proof⟩

lemma *transp-lcosetrel*:

$-x + z \in \mathcal{N}$ **if** $-x + y \in \mathcal{N}$ **and** $-y + z \in \mathcal{N}$
 ⟨proof⟩

end

class *group-add-with-distinguished-normal-subgroup* = *group-add-with-distinguished-subgroup*

+

assumes *distset-normal*: $((+) g) \text{ ' } \mathcal{N} = (\lambda x. x + g) \text{ ' } \mathcal{N}$

class *ab-group-add-with-distinguished-subgroup* =

ab-group-add + *group-add-with-distinguished-subgroup*

begin

subclass *group-add-with-distinguished-normal-subgroup*

⟨proof⟩

end

1.1.2 The quotient group type

Here we define a quotient type relative to the left-coset relation, and instantiate it as a *group-add*.

quotient-type (overloaded) *'a coset-by-dist-sbgrp* =

'a::group-add-with-distinguished-subgroup / $\lambda g h :: 'a. -g + h \in (\mathcal{N}::'a \text{ set})$

morphisms *distinguished-quotient-coset-rep* *distinguished-quotient-coset*

⟨proof⟩

lemmas *distinguished-quotient-coset-rep-inverse* =

Quotient-abs-rep[*OF Quotient-coset-by-dist-sbgrp*]

and *coset-by-dist-sbgrp-eq-iff* =

Quotient-rel-rep[*OF Quotient-coset-by-dist-sbgrp*]

and *distinguished-quotient-coset-eqI* =

Quotient-rel-abs[*OF Quotient-coset-by-dist-sbgrp*]

and *eq-to-distinguished-quotient-cosetI* =

Quotient-rel-abs2[*OF Quotient-coset-by-dist-sbgrp*]

lemma *coset-coset-by-dist-sbgrp-cases*

[*case-names coset, cases type: coset-by-dist-sbgrp*]:
($\bigwedge y. z = \text{distinguished-quotient-coset } y \implies P$) $\implies P$
{*proof*}

lemmas *distinguished-quotient-coset-eq-iff = coset-by-dist-sbgrp.abs-eq-iff*

lemma *distinguished-quotient-coset-rep-coset-equiv*:

– *distinguished-quotient-coset-rep (distinguished-quotient-coset r) + r ∈ N*
{*proof*}

instantiation *coset-by-dist-sbgrp ::*

(*group-add-with-distinguished-normal-subgroup*) *group-add*

begin

lift-definition *zero-coset-by-dist-sbgrp :: 'a coset-by-dist-sbgrp*

is *0::'a* {*proof*}

lemmas *zero-distinguished-quotient [simp] = zero-coset-by-dist-sbgrp.abs-eq[symmetric]*

lemma *zero-distinguished-quotient-eq-iffs*:

(*distinguished-quotient-coset x = 0*) = ($x \in \mathcal{N}$)
($X = 0$) = (*distinguished-quotient-coset-rep X ∈ N*)
{*proof*}

lift-definition *plus-coset-by-dist-sbgrp ::*

'a coset-by-dist-sbgrp \Rightarrow *'a coset-by-dist-sbgrp* \Rightarrow
'a coset-by-dist-sbgrp

is $\lambda x y::'a. x + y$

{*proof*}

lemmas *plus-coset-by-dist-sbgrp [simp] = plus-coset-by-dist-sbgrp.abs-eq*

lift-definition *uminus-coset-by-dist-sbgrp ::*

'a coset-by-dist-sbgrp \Rightarrow *'a coset-by-dist-sbgrp*

is $\lambda x::'a. -x$

{*proof*}

lemmas *uminus-coset-by-dist-sbgrp [simp] =*

uminus-coset-by-dist-sbgrp.abs-eq

lift-definition *minus-coset-by-dist-sbgrp ::*

'a coset-by-dist-sbgrp \Rightarrow *'a coset-by-dist-sbgrp* \Rightarrow
'a coset-by-dist-sbgrp

is $\lambda x y::'a. x - y$

{*proof*}

lemmas *minus-coset-by-dist-sbgrp [simp] = minus-coset-by-dist-sbgrp.abs-eq*

instance $\langle proof \rangle$

end

instance *coset-by-dist-sbgrp* ::
 (*ab-group-add-with-distinguished-subgroup*) *ab-group-add*
 $\langle proof \rangle$

1.2 Quotient rings

1.2.1 Class definitions

Now we set up subclasses of *ring* with a distinguished ideal.

class *ring-with-distinguished-ideal* = *ring* + *ab-group-add-with-distinguished-subgroup* +
 assumes *distset-leftideal* : $a \in \mathcal{N} \implies r * a \in \mathcal{N}$
 and *distset-rightideal*: $a \in \mathcal{N} \implies a * r \in \mathcal{N}$

class *ring-1-with-distinguished-ideal* = *ring-1* + *ring-with-distinguished-ideal* +
 assumes *distset-no1*: $1 \notin \mathcal{N}$
 — For unitary rings we do not allow the quotient to be trivial to ensure that 0 and 1 will be different.

1.2.2 The quotient ring type

We just reuse the $'a$ *coset-by-dist-sbgrp* type to define a quotient ring, and instantiate it as a *ring*.

type-synonym $'a$ *distinguished-quotient-ring* = $'a$ *coset-by-dist-sbgrp*
 — This is intended to be used with *ring-with-distinguished-ideal* or one of its subclasses.

instantiation *coset-by-dist-sbgrp* :: (*ring-with-distinguished-ideal*) *ring*
begin

lift-definition *times-coset-by-dist-sbgrp* ::
 $'a$ *distinguished-quotient-ring* \Rightarrow $'a$ *distinguished-quotient-ring* \Rightarrow
 $'a$ *distinguished-quotient-ring*
 is $\lambda x y :: 'a. x * y$
 $\langle proof \rangle$

lemmas *times-coset-by-dist-sbgrp* [*simp*] = *times-coset-by-dist-sbgrp.abs-eq*

instance $\langle proof \rangle$

end

instantiation *coset-by-dist-sbgrp* :: (*ring-1-with-distinguished-ideal*) *ring-1*

begin

lift-definition *one-coset-by-dist-sbgrp* :: '*a* distinguished-quotient-ring
is 1::'*a* <proof>

lemmas *one-coset-by-dist-sbgrp* [simp] = *one-coset-by-dist-sbgrp.abs-eq*[*symmetric*]

instance <proof>

end

1.3 Quotients of commutative rings

For a commutative ring, the quotient modulo a prime ideal is an integral domain, and the quotient modulo a maximal ideal is a field.

1.3.1 Class definitions

class *comm-ring-with-distinguished-ideal* = *comm-ring* + *ab-group-add-with-distinguished-subgroup*
+

assumes *distset-ideal* : $a \in \mathcal{N} \implies r * a \in \mathcal{N}$

begin

subclass *ring-with-distinguished-ideal*
<proof>

definition *distideal-extension* :: '*a* \Rightarrow '*a* set

where *distideal-extension* *a* $\equiv \{x. \exists r z. z \in \mathcal{N} \wedge x = r * a + z\}$

lemma *distideal-extension*: $\mathcal{N} \subseteq \text{distideal-extension } a$

<proof>

lemma *distideal-extension-diff-closed*:

$r \in \text{distideal-extension } a \implies s \in \text{distideal-extension } a \implies$

$r - s \in \text{distideal-extension } a$

<proof>

lemma *distideal-extension-ideal*:

$x \in \text{distideal-extension } a \implies r * x \in \text{distideal-extension } a$

<proof>

end

class *comm-ring-1-with-distinguished-ideal* = *ring-1* + *comm-ring-with-distinguished-ideal*

+

assumes *distset-no1*:

$1 \notin \mathcal{N}$

— Don't allow the quotient to be trivial.

begin

```

subclass ring-1-with-distinguished-ideal <proof>

lemma distideal-extension-by:  $a \in \text{distideal-extension } a$ 
  <proof>

end

class comm-ring-1-with-distinguished-pideal = comm-ring-1-with-distinguished-ideal
+
  assumes distset-pideal:  $r * s \in \mathcal{N} \implies r \in \mathcal{N} \vee s \in \mathcal{N}$ 

class comm-ring-1-with-distinguished-maxideal = comm-ring-1-with-distinguished-ideal
+
  assumes distset-maxideal:
    [
       $\mathcal{N} \subseteq I; 1 \notin I;$ 
       $\forall r s. r \in I \wedge s \in I \longrightarrow r - s \in I;$ 
       $\forall a r. a \in I \longrightarrow r * a \in I$ 
    ]  $\implies I = \mathcal{N}$ 
begin

lemma distset-maxideal-vs-distideal-extension:
   $1 \notin \text{distideal-extension } a \implies \text{distideal-extension } a = \mathcal{N}$ 
  <proof>

subclass comm-ring-1-with-distinguished-pideal
  <proof>

end

```

1.3.2 Instances and instantiations

```

instance coset-by-dist-sbgrp :: (comm-ring-with-distinguished-ideal) comm-ring
  <proof>

instance coset-by-dist-sbgrp :: (comm-ring-1-with-distinguished-ideal) comm-ring-1
  <proof>

instance coset-by-dist-sbgrp :: (comm-ring-1-with-distinguished-pideal) idom
  <proof>

lemma inverse-in-quotient-mod-maxideal:
   $\exists y. -(y * x) + 1 \in \mathcal{N}$  if  $x \notin \mathcal{N}$ 
  for  $x :: 'a::\text{comm-ring-1-with-distinguished-maxideal}$ 
  <proof>

instantiation coset-by-dist-sbgrp :: (comm-ring-1-with-distinguished-maxideal) field
begin

```

lift-definition *inverse-coset-by-dist-sbgrp* ::
'a distinguished-quotient-ring \Rightarrow *'a distinguished-quotient-ring*
is $\lambda x::'a.$ if $x \in \mathcal{N}$ then 0 else (SOME $y::'a.$ $-(y*x) + 1 \in \mathcal{N}$)
 <proof>

lemma *inverse-coset-by-dist-sbgrp-eq0* [simp]:
 $x \in \mathcal{N} \Longrightarrow \text{inverse} (\text{distinguished-quotient-coset } x) = 0$
 <proof>

lemma *coset-by-dist-sbgrp-inverse-rep*:
 $x \notin \mathcal{N} \Longrightarrow -(y * x) + 1 \in \mathcal{N} \Longrightarrow$
 $\text{inverse} (\text{distinguished-quotient-coset } x) = \text{distinguished-quotient-coset } y$
 <proof>

definition *divide-coset-by-dist-sbgrp-def* [simp]:
 $x \text{ div } y = (x::'a \text{ distinguished-quotient-ring}) * \text{inverse } y$

lemma *coset-by-dist-sbgrp-divide-rep*:
 $y \notin \mathcal{N} \Longrightarrow -(z * y) + 1 \in \mathcal{N} \Longrightarrow$
 $\text{distinguished-quotient-coset } x \text{ div distinguished-quotient-coset } y$
 $= \text{distinguished-quotient-coset } (x * z)$
 <proof>

instance <proof>

end

end

theory *Polynomial-Extras*

imports *HOL-Computational-Algebra.Polynomial*

begin

2 Additional facts about polynomials

2.1 General facts

lemma *set-strip-while*: $\text{set} (\text{strip-while } P \text{ } xs) \subseteq \text{set } xs$
 <proof>

lemma *pCons-induct'* [case-names 0 pCons0 pCons]:
 $P \text{ } p$
if $\text{zero} \quad : P \text{ } 0$

and $pCons0$: $\bigwedge a. a \neq 0 \implies P (pCons a 0)$
and $pCons-n0$: $\bigwedge a p. p \neq 0 \implies P p \implies P (pCons a p)$
 <proof>

lemma $pCons-decomp$: $pCons a p = pCons a 0 + pCons 0 p$
 <proof>

lemma $coeffs-smult-eq-smult-coeffs$:
 $coeffs (smult x p) = map ((* x) (strip-while (\lambda c. x * c = 0) (coeffs p)))$
 <proof>

lemma $coeffs-smult-eq-smult-coeffs-no-zero-divisors$:
 $coeffs (smult x p) = map ((* x) (coeffs p))$ **if** $x \neq 0$
for $x :: 'a::\{comm-semiring-0, semiring-no-zero-divisors\}$ **and** $p :: 'a poly$
 <proof>

2.2 Derivatives

The type sort on $pderiv$ is overly restrictive for algebraic purposes, so here is the relevant material with the type sort relaxed.

function $polyderiv$:: $('a :: comm-monoid-add) poly \Rightarrow 'a poly$
where $polyderiv (pCons a p) = (if p = 0 then 0 else p + pCons 0 (polyderiv p))$
 <proof>

termination $polyderiv$
 <proof>

declare $polyderiv.simps[simp del]$

lemma $polyderiv-0$ [$simp$]: $polyderiv 0 = 0$
 <proof>

lemma $polyderiv-pCons$: $polyderiv (pCons a p) = p + pCons 0 (polyderiv p)$
 <proof>

lemma $polyderiv-1$ [$simp$]: $polyderiv 1 = 0$
 <proof>

2.3 Additive polynomials

The Binomial Theorem implies that to every polynomial can be associated a finite list of polynomials that describe how the original polynomial evaluates on binomial arguments.

function $additive-poly-poly$:: $'a::comm-semiring-1 poly \Rightarrow 'a poly poly$
where
 $additive-poly-poly (pCons a p) =$
 (if $p = 0$
 then $[:a:]$)

$$\text{else smult } (pCons\ 0\ 1)\ (additive\text{-poly-poly } p) + pCons\ [:a:]\ (additive\text{-poly-poly } p)$$

$$\langle proof \rangle$$

termination *additive-poly-poly*
 $\langle proof \rangle$

declare *additive-poly-poly.simps*[*simp del*]

lemma *additive-poly-poly-0* [*simp*]: *additive-poly-poly 0 = 0*
 $\langle proof \rangle$

lemma *additive-poly-poly-pCons0*:
 $a \neq 0 \implies additive\text{-poly-poly } [:a:] = [[:a:]]$
 $\langle proof \rangle$

lemma *additive-poly-poly-pCons*:
 $p \neq 0 \implies$
 $additive\text{-poly-poly } (pCons\ a\ p) =$
 $smult\ (pCons\ 0\ 1)\ (additive\text{-poly-poly } p) + pCons\ [:a:]\ (additive\text{-poly-poly } p)$
 $\langle proof \rangle$

lemma *additive-poly-poly*: $poly\ p\ (x + y) = poly\ (poly\ (additive\text{-poly-poly } p)\ [:x:])\ y$
 $\langle proof \rangle$

lemma *additive-poly-poly-coeff0*: $coeff\ (additive\text{-poly-poly } p)\ 0 = p$
 $\langle proof \rangle$

lemma *additive-poly-poly-coeff1*: $coeff\ (additive\text{-poly-poly } p)\ 1 = polyderiv\ p$
 $\langle proof \rangle$

lemma *additive-poly-poly-eq0-iff*: $additive\text{-poly-poly } p = 0 \iff p = 0$
 $\langle proof \rangle$

lemma *additive-poly-poly-degree*: $degree\ (additive\text{-poly-poly } p) = degree\ p$
 $\langle proof \rangle$

lemma *poly-additive-poly-poly-pCons*:
 $poly\ (additive\text{-poly-poly } (pCons\ a\ p))\ [:x:] =$
 $pCons\ a\ (poly\ (additive\text{-poly-poly } p)\ [:x:]) + smult\ x\ (poly\ (additive\text{-poly-poly } p)\ [:x:])$
if $p \neq 0$
 $\langle proof \rangle$

lemma *poly-additive-poly-poly-eq0-iff*:
 $poly\ (additive\text{-poly-poly } p)\ [:x:] = 0 \iff p = 0$
 $\langle proof \rangle$

lemma *degree-poly-additive-poly-poly*:
 $\text{degree } (\text{poly } (\text{additive-poly-poly } p) [:x:]) = \text{degree } p$
 ⟨proof⟩

lemma *coeff-poly-additive-poly-poly*:
 $\text{coeff } (\text{poly } (\text{additive-poly-poly } p) [:x:]) n = \text{poly } (\text{coeff } (\text{additive-poly-poly } p) n) x$
 ⟨proof⟩

lemma *additive-poly-poly-decomp*:
 $\text{poly } p (x + y) = (\sum_{i \leq \text{degree } p} (\text{poly } (\text{coeff } (\text{additive-poly-poly } p) i) x) * y \wedge i)$
 ⟨proof⟩

end

theory *Parametric-Equiv-Depth*

imports

HOL.Rat
HOL-Computational-Algebra.Fraction-Field
HOL-Computational-Algebra.Primes
HOL-Library.Function-Algebras
HOL-Library.Set-Algebras
HOL.Topological-Spaces
Polynomial-Extras

begin

3 Common structures for adelic types

3.1 Preliminaries

lemma *ex-least-nat-le'*:
 fixes $P :: \text{nat} \Rightarrow \text{bool}$
 assumes $P n$
 shows $\exists k \leq n. (\forall i < k. \neg P i) \wedge P k$
 ⟨proof⟩

lemma *ex-ex1-least-nat-le*:
 fixes $P :: \text{nat} \Rightarrow \text{bool}$
 assumes $\exists n. P n$
 shows $\exists! k. P k \wedge (\forall i. P i \longrightarrow k \leq i)$
 ⟨proof⟩

lemma *least-le-least*:
 $\text{Least } Q \leq \text{Least } P$

```

if  $P : \exists!x. P x \wedge (\forall y. P y \longrightarrow x \leq y)$ 
and  $Q : \exists!x. Q x \wedge (\forall y. Q y \longrightarrow x \leq y)$ 
and  $PQ: \forall x. P x \longrightarrow Q x$ 
for  $P Q :: 'a::order \Rightarrow bool$ 
   $\langle proof \rangle$ 

```

lemma *linorder-wlog'* [*case-names le sym*]:

```

  [
     $\wedge a b. f a \leq f b \Longrightarrow P a b;$ 
     $\wedge a b. P b a \Longrightarrow P a b$ 
  ]  $\Longrightarrow P a b$ 
for  $f :: 'a \Rightarrow 'b::linorder$ 
   $\langle proof \rangle$ 

```

lemma *log-powi-cancel* [*simp*]:

```

   $a > 0 \Longrightarrow a \neq 1 \Longrightarrow \log a (a \text{ powi } b) = b$ 
   $\langle proof \rangle$ 

```

3.2 Existence of primes

```

class factorial-comm-ring = factorial-semiring + comm-ring
begin subclass comm-ring-1  $\langle proof \rangle$  end

```

```

class factorial-idom = factorial-semiring + idom
begin subclass factorial-comm-ring  $\langle proof \rangle$  end

```

```

class nontrivial-factorial-semiring = factorial-semiring +
  assumes nontrivial-elem-exists:  $\exists x. x \neq 0 \wedge \text{normalize } x \neq 1$ 
begin

```

```

  lemma primeE:
    obtains  $p$  where prime  $p$ 
   $\langle proof \rangle$ 

```

```

  lemma primes-exist:  $\exists p. \text{prime } p$ 
   $\langle proof \rangle$ 

```

end

```

class nontrivial-factorial-comm-ring = nontrivial-factorial-semiring + comm-ring
begin
  subclass factorial-comm-ring  $\langle proof \rangle$ 
end

```

```

class nontrivial-factorial-idom = nontrivial-factorial-semiring + idom
begin
  subclass nontrivial-factorial-comm-ring  $\langle proof \rangle$ 
  subclass factorial-idom  $\langle proof \rangle$ 
end

```

```

instance int :: nontrivial-factorial-idom
⟨proof⟩

typedef (overloaded) 'a prime =
  {p::'a::nontrivial-factorial-semiring. prime p}
  ⟨proof⟩

lemma Rep-prime-n0: Rep-prime p ≠ 0
  ⟨proof⟩

lemma Rep-prime-not-unit: ¬ is-unit (Rep-prime p)
  ⟨proof⟩

abbreviation pdvd ::
  'a::nontrivial-factorial-semiring prime ⇒ 'a ⇒ bool (infix pdvd 50)
  where p pdvd a ≡ (Rep-prime p) dvd a
abbreviation pmultiplicity p ≡ multiplicity (Rep-prime p)

lemma multiplicity-distinct-primes:
  p ≠ q ⇒ pmultiplicity p (Rep-prime q) = 0
  ⟨proof⟩

```

3.3 Generic algebraic equality

```

context
  fixes R
  assumes sympR : symp R
  and transpR: transp R
begin

lemma transp-left: R x z ⇒ R y z ⇒ R x y
  ⟨proof⟩

lemma transp-right: R x y ⇒ R x z ⇒ R y z
  ⟨proof⟩

lemma transp-iffs:
  assumes R x y shows R x z ↔ R y z and R z x ↔ R z y
  ⟨proof⟩

lemma transp-cong: R w z if R x w and R y z and R x y
  ⟨proof⟩

lemma transp-cong-sym: R x y if R x w and R y z and R w z
  ⟨proof⟩

end

```

lemma *equivp-imp-symp*: $\text{equivp } R \implies \text{symp } R$
 ⟨*proof*⟩

locale *generic-alg-equality* =
 fixes *alg-eq* :: 'a \Rightarrow 'a \Rightarrow bool
 assumes *equivp*: *equivp alg-eq*
begin

lemmas *refl* = *equivp-reflp* [*OF equivp*]
and *sym* [*sym*] = *equivp-symp* [*OF equivp*]
and *trans* [*trans*] = *equivp-transp* [*OF equivp*]
and *trans-left* = *transp-left* [*OF equivp-imp-symp equivp-imp-transp, OF equivp equivp*]
and *trans-right* = *transp-right* [*OF equivp-imp-symp equivp-imp-transp, OF equivp equivp*]
and *trans-iffs* = *transp-iffs* [*OF equivp-imp-symp equivp-imp-transp, OF equivp equivp*]
and *cong* = *transp-cong* [*OF equivp-imp-symp equivp-imp-transp, OF equivp equivp*]
and *cong-sym* = *transp-cong-sym* [*OF equivp-imp-symp equivp-imp-transp, OF equivp equivp*]

lemma *sym-iff*: $\text{alg-eq } x \ y = \text{alg-eq } y \ x$
 ⟨*proof*⟩

end

locale *ab-group-alg-equality* = *generic-alg-equality alg-eq*
 for *alg-eq* :: 'a::ab-group-add \Rightarrow 'a \Rightarrow bool
 +
 assumes *conv-0*: $\text{alg-eq } x \ y \longleftrightarrow \text{alg-eq } (x - y) \ 0$
begin

lemmas *trans0-iff* = *trans-iffs(1)[of - - 0]*

lemma *add-equiv0-left*: $\text{alg-eq } (x + y) \ y$ **if** $\text{alg-eq } x \ 0$
 ⟨*proof*⟩

lemma *add-equiv0-right*: $\text{alg-eq } (x + y) \ x$ **if** $\text{alg-eq } y \ 0$
 ⟨*proof*⟩

lemma *add*:
 $\text{alg-eq } (x1 + x2) \ (y1 + y2)$ **if** $\text{alg-eq } x1 \ y1$ **and** $\text{alg-eq } x2 \ y2$
 ⟨*proof*⟩

lemma *add-left*: $\text{alg-eq } (x + y) \ (x + y')$ **if** $\text{alg-eq } y \ y'$
 ⟨*proof*⟩

lemma *add-right*: $\text{alg-eq } (x + y) (x' + y)$ **if** $\text{alg-eq } x x'$
<proof>

lemma *add-0-left*: $\text{alg-eq } (x + y) y$ **if** $\text{alg-eq } x 0$
<proof>

lemma *add-0-right*: $\text{alg-eq } (x + y) x$ **if** $\text{alg-eq } y 0$
<proof>

lemma *sum-zeros*:
 $\text{finite } A \implies \forall a \in A. \text{alg-eq } (f a) 0 \implies \text{alg-eq } (\text{sum } f A) 0$
<proof>

lemma *uminus*: $\text{alg-eq } x y \implies \text{alg-eq } (-x) (-y)$
<proof>

lemma *uminus-iff*:
 $\text{alg-eq } x y \iff \text{alg-eq } (-x) (-y)$
<proof>

lemma *add-0*: $\text{alg-eq } (x + y) 0$ **if** $\text{alg-eq } x 0$ **and** $\text{alg-eq } y 0$
<proof>

lemma *minus*:
 $\text{alg-eq } x1 y1 \implies \text{alg-eq } x2 y2 \implies$
 $\text{alg-eq } (x1 - x2) (y1 - y2)$
<proof>

lemma *minus-0*: $\text{alg-eq } (x - y) 0$ **if** $\text{alg-eq } x 0$ **and** $\text{alg-eq } y 0$
<proof>

lemma *minus-left*: $\text{alg-eq } (x - y) (x - y')$ **if** $\text{alg-eq } y y'$
<proof>

lemma *minus-right*: $\text{alg-eq } (x - y) (x' - y)$ **if** $\text{alg-eq } x x'$
<proof>

lemma *minus-0-left*: $\text{alg-eq } (x - y) (-y)$ **if** $\text{alg-eq } x 0$
<proof>

lemma *minus-0-right*: $\text{alg-eq } (x - y) x$ **if** $\text{alg-eq } y 0$
<proof>

end

locale *ring-alg-equality* = *ab-group-alg-equality* *alg-eq*
for $\text{alg-eq} :: 'a::\text{comm-ring} \Rightarrow 'a \Rightarrow \text{bool}$

+
assumes *mult-0-right*: $\text{alg-eq } y \ 0 \implies \text{alg-eq } (x * y) \ 0$
begin

lemma *mult-0-left*: $\text{alg-eq } x \ 0 \implies \text{alg-eq } (x * y) \ 0$
 ⟨*proof*⟩

lemma *mult*:
 $\text{alg-eq } x1 \ y1 \implies \text{alg-eq } x2 \ y2 \implies \text{alg-eq } (x1 * x2) (y1 * y2)$
 ⟨*proof*⟩

lemma *mult-left*: $\text{alg-eq } y \ y' \implies \text{alg-eq } (x * y) (x * y')$
 ⟨*proof*⟩

lemma *mult-right*: $\text{alg-eq } x \ x' \implies \text{alg-eq } (x * y) (x' * y)$
 ⟨*proof*⟩

end

locale *ring-1-alg-equality* = *ring-alg-equality* *alg-eq*
for *alg-eq* :: 'a::comm-ring-1 \Rightarrow 'a \Rightarrow bool
begin

lemma *mult-one-left*: $\text{alg-eq } x \ 1 \implies \text{alg-eq } (x * y) \ y$
 ⟨*proof*⟩

lemma *mult-one-right*: $\text{alg-eq } y \ 1 \implies \text{alg-eq } (x * y) \ x$
 ⟨*proof*⟩

lemma *prod-ones*:
 $\text{finite } A \implies \forall a \in A. \text{alg-eq } (f \ a) \ 1 \implies$
 $\text{alg-eq } (\text{prod } f \ A) \ 1$
 ⟨*proof*⟩

lemma *prod-with-zero*:
 $\text{alg-eq } (\text{prod } f \ (\text{insert } a \ A)) \ 0$ **if** *finite* *A* **and** $\text{alg-eq } (f \ a) \ 0$
 ⟨*proof*⟩

lemma *pow*: $\text{alg-eq } (x \wedge^n) (y \wedge^n)$ **if** $\text{alg-eq } x \ y$
 ⟨*proof*⟩

lemma *pow-equiv0*: $\text{alg-eq } (x \wedge^n) \ 0$ **if** $n > 0$ **and** $\text{alg-eq } x \ 0$
 ⟨*proof*⟩

end

3.4 Indexed equality

3.4.1 General structure

```
locale p-equality =
  fixes p-equal :: 'a ⇒ 'b::comm-ring ⇒ 'b ⇒ bool
  assumes p-alg-equality: ring-alg-equality (p-equal p)
begin

lemmas p-group-alg-equality = ring-alg-equality.axioms(1)[OF p-alg-equality]
lemmas p-generic-alg-equality = ab-group-alg-equality.axioms(1)[OF p-group-alg-equality]

lemmas equivp      = generic-alg-equality.equivp      [OF p-generic-alg-equality]
and refl[simp]     = generic-alg-equality.refl        [OF p-generic-alg-equality]
and sym [sym]      = generic-alg-equality.sym         [OF p-generic-alg-equality]
and trans [trans]  = generic-alg-equality.trans       [OF p-generic-alg-equality]
and sym-iff        = generic-alg-equality.sym-iff     [OF p-generic-alg-equality]
and trans-left     = generic-alg-equality.trans-left  [OF p-generic-alg-equality]
and trans-right    = generic-alg-equality.trans-right [OF p-generic-alg-equality]
and trans-iffs     = generic-alg-equality.trans-iffs  [OF p-generic-alg-equality]
and cong           = generic-alg-equality.cong        [OF p-generic-alg-equality]
and cong-sym       = generic-alg-equality.cong-sym    [OF p-generic-alg-equality]

lemmas conv-0      = ab-group-alg-equality.conv-0     [OF p-group-alg-equality]
and trans0-iff     = ab-group-alg-equality.trans0-iff [OF p-group-alg-equality]
and add-equiv0-left = ab-group-alg-equality.add-equiv0-left [OF p-group-alg-equality]
and add-equiv0-right = ab-group-alg-equality.add-equiv0-right [OF p-group-alg-equality]
and add            = ab-group-alg-equality.add        [OF p-group-alg-equality]
and add-left       = ab-group-alg-equality.add-left   [OF p-group-alg-equality]
and add-right      = ab-group-alg-equality.add-right  [OF p-group-alg-equality]
and add-0-left     = ab-group-alg-equality.add-0-left [OF p-group-alg-equality]
and add-0-right    = ab-group-alg-equality.add-0-right [OF p-group-alg-equality]
and sum-zeros      = ab-group-alg-equality.sum-zeros  [OF p-group-alg-equality]
and uminus         = ab-group-alg-equality.uminus     [OF p-group-alg-equality]
and uminus-iff     = ab-group-alg-equality.uminus-iff [OF p-group-alg-equality]
and add-0          = ab-group-alg-equality.add-0      [OF p-group-alg-equality]
and minus          = ab-group-alg-equality.minus      [OF p-group-alg-equality]
and minus-0       = ab-group-alg-equality.minus-0    [OF p-group-alg-equality]
and minus-left     = ab-group-alg-equality.minus-left [OF p-group-alg-equality]
and minus-right    = ab-group-alg-equality.minus-right [OF p-group-alg-equality]
and minus-0-left   = ab-group-alg-equality.minus-0-left [OF p-group-alg-equality]
and minus-0-right  = ab-group-alg-equality.minus-0-right [OF p-group-alg-equality]

lemmas mult-0-left = ring-alg-equality.mult-0-left [OF p-alg-equality]
and mult-0-right = ring-alg-equality.mult-0-right [OF p-alg-equality]
and mult         = ring-alg-equality.mult         [OF p-alg-equality]
and mult-left    = ring-alg-equality.mult-left    [OF p-alg-equality]
and mult-right   = ring-alg-equality.mult-right   [OF p-alg-equality]

definition globally-p-equal :: 'b ⇒ 'b ⇒ bool
```

where *globally-p-equal-def*[simp]: $\text{globally-p-equal } x \ y = (\forall p::'a. \text{p-equal } p \ x \ y)$

lemma *globally-p-equalI*[intro]: $\text{globally-p-equal } x \ y$ **if** $\bigwedge p::'a. \text{p-equal } p \ x \ y$
 ⟨proof⟩

lemma *globally-p-equalD*: $\text{globally-p-equal } x \ y \implies \text{p-equal } p \ x \ y$
 ⟨proof⟩

lemma *globally-p-unequalE*:
assumes $\neg \text{globally-p-equal } x \ y$
obtains p **where** $\neg \text{p-equal } p \ x \ y$
 ⟨proof⟩

sublocale *globally-p-equality*: *ring-alg-equality* *globally-p-equal*
 ⟨proof⟩

lemmas *globally-p-equal-equivp* = *globally-p-equality-equivp*
and *globally-p-equal-refl*[simp] = *globally-p-equality-refl*
and *globally-p-equal-sym* = *globally-p-equality-sym*
and *globally-p-equal-trans* = *globally-p-equality-trans*
and *globally-p-equal-trans-left* = *globally-p-equality-trans-left*
and *globally-p-equal-trans-right* = *globally-p-equality-trans-right*
and *globally-p-equal-trans-iffs* = *globally-p-equality-trans-iffs*
and *globally-p-equal-cong* = *globally-p-equality-cong*
and *globally-p-equal-conv-0* = *globally-p-equality-conv-0*
and *globally-p-equal-trans0-iff* = *globally-p-equality-trans0-iff*
and *globally-p-equal-add-equiv0-left* = *globally-p-equality-add-equiv0-left*
and *globally-p-equal-add-equiv0-right* = *globally-p-equality-add-equiv0-right*
and *globally-p-equal-add* = *globally-p-equality-add*
and *globally-p-equal-uminus* = *globally-p-equality-uminus*
and *globally-p-equal-uminus-iff* = *globally-p-equality-uminus-iff*
and *globally-p-equal-add-0* = *globally-p-equality-add-0*
and *globally-p-equal-minus* = *globally-p-equality-minus*
and *globally-p-equal-mult-0-left* = *globally-p-equality-mult-0-left*
and *globally-p-equal-mult-0-right* = *globally-p-equality-mult-0-right*
and *globally-p-equal-mult* = *globally-p-equality-mult*
and *globally-p-equal-mult-left* = *globally-p-equality-mult-left*
and *globally-p-equal-mult-right* = *globally-p-equality-mult-right*

lemma *globally-p-equal-transfer-equals-rsp*:
 $\text{p-equal } p \ x1 \ y1 \longleftrightarrow \text{p-equal } p \ x2 \ y2$
if $\text{globally-p-equal } x1 \ x2$ **and** $\text{globally-p-equal } y1 \ y2$
 ⟨proof⟩

end

locale *p-equality-1* = *p-equality* *p-equal*
for $\text{p-equal} :: 'a \Rightarrow 'b::\text{comm-ring-1} \Rightarrow 'b \Rightarrow \text{bool}$

begin

lemma *p-alg-equality-1*: *ring-1-alg-equality* (*p-equal* *p*)
⟨*proof*⟩

lemmas *mult-one-left* = *ring-1-alg-equality.mult-one-left* [*OF p-alg-equality-1*]
and *mult-one-right* = *ring-1-alg-equality.mult-one-right* [*OF p-alg-equality-1*]
and *prod-ones* = *ring-1-alg-equality.prod-ones* [*OF p-alg-equality-1*]
and *pow* = *ring-1-alg-equality.pow* [*OF p-alg-equality-1*]
and *pow-equiv0* = *ring-1-alg-equality.pow-equiv0* [*OF p-alg-equality-1*]

end

locale *p-equality-no-zero-divisors* = *p-equality*

+

assumes *no-zero-divisors*:

$\neg p\text{-equal } p \ x \ 0 \implies \neg p\text{-equal } p \ y \ 0 \implies$
 $\neg p\text{-equal } p \ (x * y) \ 0$

begin

lemma *mult-equiv-0-iff*:

$p\text{-equal } p \ (x * y) \ 0 \longleftrightarrow$
 $p\text{-equal } p \ x \ 0 \vee p\text{-equal } p \ y \ 0$
⟨*proof*⟩

end

locale *p-equality-no-zero-divisors-1* = *p-equality-no-zero-divisors* *p-equal*

for *p-equal* ::

$'a \Rightarrow 'b :: \text{comm-ring-1} \Rightarrow 'b \Rightarrow \text{bool}$

+

assumes *nontrivial*: $\exists x. \neg p\text{-equal } p \ x \ 0$

begin

sublocale *p-equality-1* ⟨*proof*⟩

lemma *one-p-inequal-zero*: $\neg p\text{-equal } p \ (1::'b) \ (0::'b)$
⟨*proof*⟩

lemma *zero-p-inequal-one*: $\neg p\text{-equal } p \ (0::'b) \ (1::'b)$
⟨*proof*⟩

lemma *neg-one-p-inequal-zero*: $\neg p\text{-equal } p \ (-1::'b) \ (0::'b)$
⟨*proof*⟩

lemma *pow-equiv-0-iff*: $p\text{-equal } p \ (x \wedge^n) \ 0 \longleftrightarrow p\text{-equal } p \ x \ 0 \wedge n \neq 0$
⟨*proof*⟩

lemma *pow-equiv-base*: $p\text{-equal } p (x \wedge n) (y \wedge n)$ **if** $p\text{-equal } p x y$
 ⟨*proof*⟩

end

locale *p-equality-div-inv* = *p-equality-no-zero-divisors-1* *p-equal*
for *p-equal* ::

'*a* ⇒ '*b*::{*comm-ring-1*, *inverse*, *divide-trivial*} ⇒
 '*b* ⇒ *bool*

+

assumes *divide-inverse* : $\bigwedge x y :: 'b. x / y = x * \text{inverse } y$
and *inverse-inverse* : $\bigwedge x :: 'b. \text{inverse } (\text{inverse } x) = x$
and *inverse-mult* : $\bigwedge x y :: 'b. \text{inverse } (x * y) = \text{inverse } x * \text{inverse } y$
and *inverse-equiv0-iff*: $p\text{-equal } p (\text{inverse } x) 0 \longleftrightarrow p\text{-equal } p x 0$
and *divide-self-equiv* : $\neg p\text{-equal } p x 0 \implies p\text{-equal } p (x / x) 1$

begin

Global algebra facts

lemma *inverse-eq-divide*: $\text{inverse } x = 1 / x$ **for** $x :: 'b$
 ⟨*proof*⟩

lemma *inverse-div-inverse*: $\text{inverse } x / \text{inverse } y = y / x$ **for** $x y :: 'b$
 ⟨*proof*⟩

lemma *inverse-0[simp]*: $\text{inverse } (0 :: 'b) = 0$
 ⟨*proof*⟩

lemma *inverse-1[simp]*: $\text{inverse } (1 :: 'b) = 1$
 ⟨*proof*⟩

lemma *inverse-pow*: $\text{inverse } (x \wedge n) = \text{inverse } x \wedge n$ **for** $x :: 'b$
 ⟨*proof*⟩

lemma *power-int-minus*: $\text{power-int } x (-n) = \text{inverse } (\text{power-int } x n)$ **for** $x :: 'b$
 ⟨*proof*⟩

lemma *power-int-minus-divide*: $\text{power-int } x (-n) = 1 / (\text{power-int } x n)$ **for** $x :: 'b$
 ⟨*proof*⟩

lemma *power-int-0-left-if*: $\text{power-int } (0 :: 'b) m = (\text{if } m = 0 \text{ then } 1 \text{ else } 0)$
 ⟨*proof*⟩

lemma *power-int-0-left [simp]*: $m \neq 0 \implies \text{power-int } (0 :: 'b) m = 0$
 ⟨*proof*⟩

lemma *power-int-1-left [simp]*: $\text{power-int } (1 :: 'b) n = 1$
 ⟨*proof*⟩

lemma *inverse-power-int*: $\text{inverse} (\text{power-int } x \ n) = \text{power-int } x \ (-n)$ **for** $x :: 'b$
<proof>

lemma *power-int-inverse*:
 $\text{power-int} (\text{inverse } x) \ n = \text{inverse} (\text{power-int } x \ n)$ **for** $x :: 'b$
<proof>

lemma *power-int-mult-distrib*:
 $\text{power-int} (x * y) \ m = \text{power-int } x \ m * \text{power-int } y \ m$ **for** $x \ y :: 'b$
<proof>

lemma *inverse-divide*: $\text{inverse} (a / b) = b / a$ **for** $a \ b :: 'b$
<proof>

lemma *minus-divide-left*: $-(a / b) = (-a) / b$ **for** $a \ b :: 'b$
<proof>

lemma *times-divide-times-eq*: $(a / b) * (c / d) = (a * c) / (b * d)$ **for** $a \ b :: 'b$
<proof>

lemma *divide-pow*: $(x \wedge n) / (y \wedge n) = (x / y) \wedge n$ **for** $x \ y :: 'b$
<proof>

lemma *power-int-divide-distrib*:
 $\text{power-int} (x / y) \ m = \text{power-int } x \ m / \text{power-int } y \ m$ **for** $x \ y :: 'b$
<proof>

lemma *power-int-one-over*: $\text{power-int} (1 / x) \ n = 1 / \text{power-int } x \ n$ **for** $x :: 'b$
<proof>

lemma *add-divide-distrib*: $(a + b) / c = (a / c) + (b / c)$ **for** $a \ b \ c :: 'b$
<proof>

lemma *diff-divide-distrib*: $(a - b) / c = (a / c) - (b / c)$ **for** $a \ b \ c :: 'b$
<proof>

lemma *times-divide-eq-right*: $a * (b / c) = (a * b) / c$ **for** $a \ b \ c :: 'b$
<proof>

lemma *times-divide-eq-left*: $(b / c) * a = (b * a) / c$ **for** $a \ b \ c :: 'b$
<proof>

lemma *divide-divide-eq-left*: $(a / b) / c = a / (b * c)$ **for** $a \ b \ c :: 'b$
<proof>

lemma *swap-numerator*: $a * (b / c) = (a / c) * b$ **for** $a \ b \ c :: 'b$
<proof>

lemma *divide-divide-times-eq*: $(a / b) / (c / d) = (a * d) / (b * c)$ **for** $a b c d :: 'b$

<proof>

Algebra facts by place

lemma *right-inverse-equiv*: $p\text{-equal } p (x * \text{inverse } x) 1$ **if** $\neg p\text{-equal } p x 0$

<proof>

lemma *left-inverse-equiv*: $p\text{-equal } p (\text{inverse } x * x) 1$ **if** $\neg p\text{-equal } p x 0$

<proof>

lemma *inverse-p-unique*:

$p\text{-equal } p (\text{inverse } x) y$ **if** $p\text{-equal } p (x * y) 1$

<proof>

lemma *inverse-equiv-imp-equiv*:

$p\text{-equal } p (\text{inverse } a) (\text{inverse } b) \implies p\text{-equal } p a b$

<proof>

lemma *inverse-pcong*: $p\text{-equal } p (\text{inverse } x) (\text{inverse } y)$ **if** $p\text{-equal } p x y$

<proof>

lemma *inverse-equiv-iff-equiv*:

$p\text{-equal } p (\text{inverse } a) (\text{inverse } b) \iff p\text{-equal } p a b$

<proof>

lemma *power-int-equiv-0-iff*:

$p\text{-equal } p (\text{power-int } x n) 0 \iff p\text{-equal } p x 0 \wedge n \neq 0$

<proof>

lemma *power-diff-conv-inverse-equiv*:

$m \leq n \implies p\text{-equal } p (x \wedge (n - m)) (x \wedge n * \text{inverse } x \wedge m)$

if $\neg p\text{-equal } p x 0$

<proof>

lemma *power-int-add-1-equiv*:

$p\text{-equal } p (\text{power-int } x (m + 1)) (\text{power-int } x m * x)$

if $\neg p\text{-equal } p x 0 \vee m \neq -1$

<proof>

lemma *power-int-add-equiv*:

$\neg p\text{-equal } p x 0 \vee m + n \neq 0 \implies$

$p\text{-equal } p (\text{power-int } x (m + n)) (\text{power-int } x m * \text{power-int } x n)$

<proof>

lemma *power-int-diff-equiv*:

$p\text{-equal } p (\text{power-int } x (m - n)) (\text{power-int } x m / \text{power-int } x n)$

if $\neg p\text{-equal } p x 0 \vee m \neq n$

<proof>

lemma *power-int-minus-mult-equiv*:

$p\text{-equal } p \text{ (power-int } x \text{ (} n - 1 \text{) } * x \text{) (power-int } x \text{ } n)$
if $\neg p\text{-equal } p \text{ } x \text{ } 0 \vee n \neq 0$
(proof)

lemma *power-int-not-equiv-zero*:

$\neg p\text{-equal } p \text{ } x \text{ } 0 \vee n = 0 \implies \neg p\text{-equal } p \text{ (power-int } x \text{ } n) \text{ } 0$
(proof)

lemma *power-int-equiv-base*: $p\text{-equal } p \text{ (power-int } x \text{ } n) \text{ (power-int } y \text{ } n)$ **if** $p\text{-equal } p \text{ } x \text{ } y$

(proof)

lemma *divide-equiv-0-iff*:

$p\text{-equal } p \text{ (} x / y \text{) } 0 \iff p\text{-equal } p \text{ } x \text{ } 0 \vee p\text{-equal } p \text{ } y \text{ } 0$
(proof)

lemma *divide-pcong*: $p\text{-equal } p \text{ (} w / x \text{) (} y / z \text{)}$ **if** $p\text{-equal } p \text{ } w \text{ } y$ **and** $p\text{-equal } p \text{ } x \text{ } z$

(proof)

lemma *divide-left-equiv*: $p\text{-equal } p \text{ } x \text{ } y \implies p\text{-equal } p \text{ (} x / z \text{) (} y / z \text{)}$

(proof)

lemma *divide-right-equiv*: $p\text{-equal } p \text{ } x \text{ } y \implies p\text{-equal } p \text{ (} z / x \text{) (} z / y \text{)}$

(proof)

lemma *add-frac-equiv*:

$p\text{-equal } p \text{ ((} a / b \text{) + (} c / d \text{)) ((} a * d + c * b \text{) / (} b * d \text{))}$
if $\neg p\text{-equal } p \text{ } b \text{ } 0$ **and** $\neg p\text{-equal } p \text{ } d \text{ } 0$
(proof)

lemma *mult-divide-mult-cancel-right*:

$p\text{-equal } p \text{ ((} a * b \text{) / (} c * b \text{)) (} a / c \text{)}$ **if** $\neg p\text{-equal } p \text{ } b \text{ } 0$
(proof)

lemma *mult-divide-mult-cancel-right2*:

$p\text{-equal } p \text{ ((} a * c \text{) / (} c * b \text{)) (} a / b \text{)}$ **if** $\neg p\text{-equal } p \text{ } c \text{ } 0$
(proof)

lemma *mult-divide-mult-cancel-left*:

$p\text{-equal } p \text{ ((} c * a \text{) / (} c * b \text{)) (} a / b \text{)}$ **if** $\neg p\text{-equal } p \text{ } c \text{ } 0$
(proof)

lemma *mult-divide-mult-cancel-left2*:

$p\text{-equal } p \text{ ((} c * a \text{) / (} b * c \text{)) (} a / b \text{)}$ **if** $\neg p\text{-equal } p \text{ } c \text{ } 0$
(proof)

lemma *mult-divide-cancel-right*: $p\text{-equal } p \text{ ((} a * b \text{) / } b \text{) } a$ **if** $\neg p\text{-equal } p \text{ } b \text{ } 0$

<proof>

lemma *mult-divide-cancel-left*: $p\text{-equal } p ((a * b) / a) b \text{ if } \neg p\text{-equal } p a 0$
<proof>

lemma *divide-equiv-equiv*: $(p\text{-equal } p (b / c) a) = (p\text{-equal } p b (a * c)) \text{ if } \neg p\text{-equal } p c 0$
<proof>

lemma *neg-divide-equiv-equiv*:
 $p\text{-equal } p (- (b / c)) a \iff p\text{-equal } p (-b) (a * c) \text{ if } \neg p\text{-equal } p c 0$
<proof>

lemma *equiv-divide-imp*:
 $p\text{-equal } p (a * c) b \implies p\text{-equal } p a (b / c) \text{ if } \neg p\text{-equal } p c 0$
<proof>

lemma *add-divide-equiv-iff*:
 $p\text{-equal } p (x + y / z) ((x * z + y) / z) \text{ if } \neg p\text{-equal } p z 0$
<proof>

lemma *divide-add-equiv-iff*: $p\text{-equal } p (x / z + y) ((x + y * z) / z) \text{ if } \neg p\text{-equal } p z 0$
<proof>

lemma *diff-divide-equiv-iff*: $p\text{-equal } p (x - y / z) ((x * z - y) / z) \text{ if } \neg p\text{-equal } p z 0$
<proof>

lemma *minus-divide-add-equiv-iff*:
 $p\text{-equal } p (- (x / z) + y) ((- x + y * z) / z) \text{ if } \neg p\text{-equal } p z 0$
<proof>

lemma *divide-diff-equiv-iff*: $p\text{-equal } p (x / z - y) ((x - y * z) / z) \text{ if } \neg p\text{-equal } p z 0$
<proof>

lemma *minus-divide-diff-equiv-iff*:
 $p\text{-equal } p (- (x / z) - y) ((- x - y * z) / z) \text{ if } \neg p\text{-equal } p z 0$
<proof>

lemma *divide-mult-cancel-left*: $p\text{-equal } p (a / (a * b)) (1 / b) \text{ if } \neg p\text{-equal } p a 0$
<proof>

lemma *divide-mult-cancel-right*: $p\text{-equal } p (b / (a * b)) (1 / a) \text{ if } \neg p\text{-equal } p b 0$
<proof>

lemma *diff-frac-equiv*:
 $p\text{-equal } p (x / y - w / z) ((x * z - w * y) / (y * z))$

if $\neg p\text{-equal } p \ y \ 0$ **and** $\neg p\text{-equal } p \ z \ 0$
<proof>

lemma *frac-equiv-equiv:*

$p\text{-equal } p \ (x \ / \ y) \ (w \ / \ z) \longleftrightarrow p\text{-equal } p \ (x \ * \ z) \ (w \ * \ y)$
if $\neg p\text{-equal } p \ y \ 0$ **and** $\neg p\text{-equal } p \ z \ 0$
<proof>

lemma *inverse-equiv-1-iff:*

$p\text{-equal } p \ (\text{inverse } x) \ 1 \longleftrightarrow p\text{-equal } p \ x \ 1$
<proof>

lemma *inverse-neg-1-equiv:* $p\text{-equal } p \ (\text{inverse } (-1)) \ (-1)$
<proof>

lemma *globally-inverse-neg-1:* $\text{globally-}p\text{-equal } (\text{inverse } (-1)) \ (-1)$
<proof>

lemma *inverse-uminus-equiv:* $p\text{-equal } p \ (\text{inverse } (-x)) \ (- \ \text{inverse } x)$
<proof>

lemma *minus-divide-right-equiv:* $p\text{-equal } p \ (a \ / \ (- \ b)) \ (- \ (a \ / \ b))$
<proof>

lemma *minus-divide-divide-equiv:* $p\text{-equal } p \ ((- \ a) \ / \ (- \ b)) \ (a \ / \ b)$
<proof>

lemma *divide-minus1-equiv:* $p\text{-equal } p \ (x \ / \ (-1)) \ (- \ x)$
<proof>

lemma *divide-cancel-right:*

$p\text{-equal } p \ (a \ / \ c) \ (b \ / \ c) \longleftrightarrow p\text{-equal } p \ c \ 0 \ \vee \ p\text{-equal } p \ a \ b$
<proof>

lemma *divide-cancel-left:*

$p\text{-equal } p \ (c \ / \ a) \ (c \ / \ b) \longleftrightarrow p\text{-equal } p \ c \ 0 \ \vee \ p\text{-equal } p \ a \ b$
<proof>

lemma *divide-equiv-1-iff:*

$p\text{-equal } p \ (a \ / \ b) \ 1 \longleftrightarrow \neg p\text{-equal } p \ b \ 0 \ \wedge \ p\text{-equal } p \ a \ b$
<proof>

lemma *divide-equiv-minus-1-iff:*

$p\text{-equal } p \ (a \ / \ b) \ (- \ 1) \longleftrightarrow \neg p\text{-equal } p \ b \ 0 \ \wedge \ p\text{-equal } p \ a \ (- \ b)$
<proof>

end

3.5 Indexed depth functions

3.5.1 General structure

locale *p-equality-depth* = *p-equality* +
fixes *p-depth* :: 'a \Rightarrow 'b::comm-ring \Rightarrow int
assumes *depth-of-0*[simp]: *p-depth* p 0 = 0
and *depth-equiv*: *p-equal* p x y \implies *p-depth* p x = *p-depth* p y
and *depth-uminus*: *p-depth* p (-x) = *p-depth* p x
and *depth-pre-nonarch*:
 \neg *p-equal* p x 0 \implies *p-depth* p x < *p-depth* p y \implies
p-depth p (x + y) = *p-depth* p x
 $\llbracket \neg$ *p-equal* p x 0; \neg *p-equal* p (x + y) 0; *p-depth* p x = *p-depth* p y \rrbracket
 \implies *p-depth* p (x + y) \geq *p-depth* p x

begin

lemma *depth-equiv-0*: *p-equal* p x 0 \implies *p-depth* p x = 0

<proof>

lemma *depth-equiv-uminus*: *p-equal* p y (-x) \implies *p-depth* p y = *p-depth* p x

<proof>

lemma *depth-add-equiv0-left*: *p-equal* p x 0 \implies *p-depth* p (x + y) = *p-depth* p y

<proof>

lemma *depth-add-equiv0-right*: *p-equal* p y 0 \implies *p-depth* p (x + y) = *p-depth* p x

<proof>

lemma *depth-add-equiv*:

p-depth p (x + y) = *p-depth* p (w + z) **if** *p-equal* p x w **and** *p-equal* p y z

<proof>

lemma *depth-diff*: *p-depth* p (x - y) = *p-depth* p (y - x)

<proof>

lemma *depth-diff-equiv0-left*: *p-equal* p x 0 \implies *p-depth* p (x - y) = *p-depth* p y

<proof>

lemma *depth-diff-equiv0-right*: *p-equal* p y 0 \implies *p-depth* p (x - y) = *p-depth* p x

<proof>

lemma *depth-diff-equiv*:

p-depth p (x - y) = *p-depth* p (w - z) **if** *p-equal* p x w **and** *p-equal* p y z

<proof>

lemma *depth-pre-nonarch-diff-left*:

p-depth p (x - y) = *p-depth* p x **if** \neg *p-equal* p x 0 **and** *p-depth* p x < *p-depth* p

y

<proof>

lemma *depth-pre-nonarch-diff-right*:

$p\text{-depth } p (x - y) = p\text{-depth } p y$ **if** $\neg p\text{-equal } p y 0$ **and** $p\text{-depth } p y < p\text{-depth } p x$
 ⟨proof⟩

lemma *depth-nonarch*:

$\llbracket \neg p\text{-equal } p x 0; \neg p\text{-equal } p y 0; \neg p\text{-equal } p (x + y) 0 \rrbracket$
 $\implies p\text{-depth } p (x + y) \geq \min (p\text{-depth } p x) (p\text{-depth } p y)$
 ⟨proof⟩

lemma *depth-nonarch-diff*:

$\llbracket \neg p\text{-equal } p x 0; \neg p\text{-equal } p y 0; \neg p\text{-equal } p (x - y) 0 \rrbracket$
 $\implies p\text{-depth } p (x - y) \geq \min (p\text{-depth } p x) (p\text{-depth } p y)$
 ⟨proof⟩

lemma *depth-sum-nonarch*:

$\text{finite } A \implies \neg p\text{-equal } p (\text{sum } f A) 0 \implies$
 $p\text{-depth } p (\text{sum } f A) \geq \text{Min } \{p\text{-depth } p (f a) \mid a. a \in A \wedge \neg p\text{-equal } p (f a) 0 \}$
 ⟨proof⟩

lemma *obtains-depth-sum-nonarch-witness*:

assumes $\text{finite } A$ **and** $\neg p\text{-equal } p (\text{sum } f A) 0$
obtains a
where $a \in A$ **and** $\neg p\text{-equal } p (f a) 0$ **and** $p\text{-depth } p (\text{sum } f A) \geq p\text{-depth } p (f a)$
 ⟨proof⟩

lemma *depth-add-cancel-imp-eq-depth*:

$p\text{-depth } p x = p\text{-depth } p y$ **if** $\neg p\text{-equal } p x 0$ **and** $p\text{-depth } p (x + y) > p\text{-depth } p x$
 ⟨proof⟩

lemma *depth-diff-cancel-imp-eq-depth-left*:

$p\text{-depth } p x = p\text{-depth } p y$ **if** $\neg p\text{-equal } p x 0$ **and** $p\text{-depth } p (x - y) > p\text{-depth } p x$
 ⟨proof⟩

lemma *depth-diff-cancel-imp-eq-depth-right*:

$p\text{-depth } p x = p\text{-depth } p y$ **if** $\neg p\text{-equal } p y 0$ **and** $p\text{-depth } p (x - y) > p\text{-depth } p y$
 ⟨proof⟩

lemma *level-closed-add*:

$p\text{-depth } p (x + y) \geq n$
if $\neg p\text{-equal } p (x + y) 0$ **and** $p\text{-depth } p x \geq n$ **and** $p\text{-depth } p y \geq n$
 ⟨proof⟩

lemma *level-closed-diff*:

$p\text{-depth } p (x - y) \geq n$

if $\neg p\text{-equal } p (x - y) 0$ **and** $p\text{-depth } p x \geq n$ **and** $p\text{-depth } p y \geq n$
(proof)

definition $p\text{-depth-set} :: 'a \Rightarrow \text{int} \Rightarrow 'b \text{ set}$
where $p\text{-depth-set } p n = \{x. \neg p\text{-equal } p x 0 \longrightarrow p\text{-depth } p x \geq n\}$

definition $\text{global-depth-set} :: \text{int} \Rightarrow 'b \text{ set}$
where
 $\text{global-depth-set } n =$
 $\{x. \forall p. \neg p\text{-equal } p x 0 \longrightarrow p\text{-depth } p x \geq n\}$

lemma global-depth-set : $\text{global-depth-set } n = (\bigcap p. p\text{-depth-set } p n)$
(proof)

lemma $p\text{-depth-set}D$:
 $\neg p\text{-equal } p x 0 \Longrightarrow x \in p\text{-depth-set } p n \Longrightarrow$
 $p\text{-depth } p x \geq n$
(proof)

lemma $\text{nonpos-}p\text{-depth-set}D$:
 $n \leq 0 \Longrightarrow x \in p\text{-depth-set } p n \Longrightarrow p\text{-depth } p x \geq n$
(proof)

lemma $\text{global-depth-set}D$:
 $\neg p\text{-equal } p x 0 \Longrightarrow x \in \text{global-depth-set } n \Longrightarrow$
 $p\text{-depth } p x \geq n$
(proof)

lemma $\text{nonpos-global-depth-set}D$:
 $n \leq 0 \Longrightarrow x \in \text{global-depth-set } n \Longrightarrow p\text{-depth } p x \geq n$
(proof)

lemma $p\text{-depth-set-}0$: $0 \in p\text{-depth-set } p n$
(proof)

lemma $\text{global-depth-set-}0$: $0 \in \text{global-depth-set } n$
(proof)

lemma $p\text{-depth-set-equiv}0$: $p\text{-equal } p x 0 \Longrightarrow x \in p\text{-depth-set } p n$
(proof)

lemma $p\text{-depth-set}I$:
 $x \in p\text{-depth-set } p n$ **if** $\neg p\text{-equal } p x 0 \longrightarrow p\text{-depth } p x \geq n$
(proof)

lemma $p\text{-depth-set-by-equiv}I$:
 $x \in p\text{-depth-set } p n$ **if** $p\text{-equal } p x y$ **and** $y \in p\text{-depth-set } p n$
(proof)

lemma *global-depth-setI*:

$x \in \text{global-depth-set } n$

if $\bigwedge p. \neg p\text{-equal } p \ x \ 0 \implies p\text{-depth } p \ x \geq n$

<proof>

lemma *global-depth-setI'*: $x \in \text{global-depth-set } n$ **if** $\bigwedge p. p\text{-depth } p \ x \geq n$

<proof>

lemma *global-depth-setI''*: $x \in \text{global-depth-set } n$ **if** $\bigwedge p. x \in p\text{-depth-set } p \ n$

<proof>

lemma *p-depth-set-antimono*:

$n \leq m \implies p\text{-depth-set } p \ n \supseteq p\text{-depth-set } p \ m$

<proof>

lemma *global-depth-set-antimono*:

$n \leq m \implies \text{global-depth-set } n \supseteq \text{global-depth-set } m$

<proof>

lemma *p-depth-set-add*:

$x + y \in p\text{-depth-set } p \ n$ **if** $x \in p\text{-depth-set } p \ n$ **and** $y \in p\text{-depth-set } p \ n$

<proof>

lemma *global-depth-set-add*:

$x + y \in \text{global-depth-set } n$ **if** $x \in \text{global-depth-set } n$ **and** $y \in \text{global-depth-set } n$

<proof>

lemma *p-depth-set-uminus*: $-x \in p\text{-depth-set } p \ n$ **if** $x \in p\text{-depth-set } p \ n$

<proof>

lemma *global-depth-set-uminus*: $-x \in \text{global-depth-set } n$ **if** $x \in \text{global-depth-set } n$

<proof>

lemma *p-depth-set-minus*:

$x - y \in p\text{-depth-set } p \ n$ **if** $x \in p\text{-depth-set } p \ n$ **and** $y \in p\text{-depth-set } p \ n$

<proof>

lemma *global-depth-set-minus*:

$x - y \in \text{global-depth-set } n$ **if** $x \in \text{global-depth-set } n$ **and** $y \in \text{global-depth-set } n$

<proof>

lemma *p-depth-set-elt-set-plus*:

$x + o \ p\text{-depth-set } p \ n = p\text{-depth-set } p \ n$ **if** $x \in p\text{-depth-set } p \ n$

<proof>

lemma *global-depth-set-elt-set-plus*:

$x + o \ \text{global-depth-set } n = \text{global-depth-set } n$ **if** $x \in \text{global-depth-set } n$

<proof>

lemma *p-depth-set-elt-set-plus-closed*:
 $x + o$ *p-depth-set* p $m \subseteq p$ -depth-set p n **if** $m \geq n$ **and** $x \in p$ -depth-set p n
⟨proof⟩

lemma *global-depth-set-elt-set-plus-closed*:
 $x + o$ *global-depth-set* $m \subseteq$ *global-depth-set* n
if $m \geq n$ **and** $x \in$ *global-depth-set* n
⟨proof⟩

lemma *p-depth-set-plus-coeffs*:
set $xs \subseteq p$ -depth-set p $n \implies$
set $ys \subseteq p$ -depth-set p $n \implies$
set (*plus-coeffs* xs ys) $\subseteq p$ -depth-set p n
⟨proof⟩

lemma *global-depth-set-plus-coeffs*:
set (*plus-coeffs* xs ys) \subseteq *global-depth-set* n
if *set* $xs \subseteq$ *global-depth-set* n **and** *set* $ys \subseteq$ *global-depth-set* n
⟨proof⟩

lemma *p-depth-set-poly-pCons*:
set (*coeffs* (*pCons* a f)) $\subseteq p$ -depth-set p n
if $a \in p$ -depth-set p n **and** *set* (*coeffs* f) $\subseteq p$ -depth-set p n
⟨proof⟩

lemma *p-depth-set-poly-add*:
set (*coeffs* ($f + g$)) $\subseteq p$ -depth-set p n
if *set* (*coeffs* f) $\subseteq p$ -depth-set p n **and** *set* (*coeffs* g) $\subseteq p$ -depth-set p n
⟨proof⟩

lemma *global-depth-set-poly-add*:
set (*coeffs* ($f + g$)) \subseteq *global-depth-set* n
if *set* (*coeffs* f) \subseteq *global-depth-set* n
and *set* (*coeffs* g) \subseteq *global-depth-set* n
⟨proof⟩

end

locale *p-equality-depth-no-zero-divisors* =
p-equality-no-zero-divisors + *p-equality-depth*
+ **assumes** *depth-mult-additive*:
 $\neg p$ -equal p ($x * y$) $0 \implies p$ -depth p ($x * y$) = p -depth p x + p -depth p y
begin

lemma *depth-mult3-additive*:
 p -depth p ($x * y * z$) = p -depth p x + p -depth p y + p -depth p z
if $\neg p$ -equal p ($x * y * z$) 0
⟨proof⟩

lemma *p-depth-set-times*:

$x * y \in p\text{-depth-set } p \ n$

if $n \geq 0$ **and** $x \in p\text{-depth-set } p \ n$ **and** $y \in p\text{-depth-set } p \ n$

<proof>

lemma *global-depth-set-times*:

$x * y \in \text{global-depth-set } n$

if $n \geq 0$ **and** $x \in \text{global-depth-set } n$ **and** $y \in \text{global-depth-set } n$

<proof>

lemma *poly-p-depth-set-poly*:

$\text{set } (\text{coeffs } f) \subseteq p\text{-depth-set } p \ n \implies \text{poly } f \ x \in p\text{-depth-set } p \ n$

if $n \geq 0$ **and** $x \in p\text{-depth-set } p \ n$

<proof>

lemma *poly-global-depth-set-poly*:

$\text{poly } f \ x \in \text{global-depth-set } n$

if $n \geq 0$

and $x \in \text{global-depth-set } n$

and $\text{set } (\text{coeffs } f) \subseteq \text{global-depth-set } n$

<proof>

lemma *p-depth-set-ideal*:

$x * y \in p\text{-depth-set } p \ n$

if $x \in p\text{-depth-set } p \ 0$ **and** $y \in p\text{-depth-set } p \ n$

<proof>

lemma *global-depth-set-ideal*:

$x * y \in \text{global-depth-set } n$

if $x \in \text{global-depth-set } 0$ **and** $y \in \text{global-depth-set } n$

<proof>

lemma *poly-p-depth-set-poly-ideal*:

$\text{set } (\text{coeffs } f) \subseteq p\text{-depth-set } p \ n \implies \text{poly } f \ x \in p\text{-depth-set } p \ n$

if $x \in p\text{-depth-set } p \ 0$

<proof>

lemma *poly-global-depth-set-poly-ideal*:

$\text{poly } f \ x \in \text{global-depth-set } n$

if $x \in \text{global-depth-set } 0$ **and** $\text{set } (\text{coeffs } f) \subseteq \text{global-depth-set } n$

<proof>

lemma *p-depth-set-poly-smult*:

$\text{set } (\text{coeffs } (\text{smult } x \ f)) \subseteq p\text{-depth-set } p \ n$

if $n \geq 0$ **and** $x \in p\text{-depth-set } p \ n$ **and** $\text{set } (\text{coeffs } f) \subseteq p\text{-depth-set } p \ n$

<proof>

lemma *global-depth-set-poly-smult*:

$set (coeffs (smult x f)) \subseteq global\text{-}depth\text{-}set n$
if $n \geq 0$
and $x \in global\text{-}depth\text{-}set n$
and $set (coeffs f) \subseteq global\text{-}depth\text{-}set n$
 ⟨proof⟩

lemma *p-depth-set-poly-smult-ideal:*

$set (coeffs (smult x f)) \subseteq p\text{-}depth\text{-}set p n$
if $x \in p\text{-}depth\text{-}set p 0$ **and** $set (coeffs f) \subseteq p\text{-}depth\text{-}set p n$
 ⟨proof⟩

lemma *global-depth-set-poly-smult-ideal:*

$set (coeffs (smult x f)) \subseteq global\text{-}depth\text{-}set n$
if $x \in global\text{-}depth\text{-}set 0$ **and** $set (coeffs f) \subseteq global\text{-}depth\text{-}set n$
 ⟨proof⟩

lemma *p-depth-set-poly-times:*

$set (coeffs f) \subseteq p\text{-}depth\text{-}set p n \implies$
 $set (coeffs g) \subseteq p\text{-}depth\text{-}set p n \implies$
 $set (coeffs (f * g)) \subseteq p\text{-}depth\text{-}set p n$
if $n \geq 0$
 ⟨proof⟩

lemma *global-depth-set-poly-times:*

$set (coeffs (f * g)) \subseteq global\text{-}depth\text{-}set n$
if $n \geq 0$
and $set (coeffs g) \subseteq global\text{-}depth\text{-}set n$
and $set (coeffs f) \subseteq global\text{-}depth\text{-}set n$
 ⟨proof⟩

lemma *p-depth-set-poly-times-ideal:*

$set (coeffs f) \subseteq p\text{-}depth\text{-}set p 0 \implies$
 $set (coeffs g) \subseteq p\text{-}depth\text{-}set p n \implies$
 $set (coeffs (f * g)) \subseteq p\text{-}depth\text{-}set p n$
 ⟨proof⟩

lemma *global-depth-set-poly-times-ideal:*

$set (coeffs (f * g)) \subseteq global\text{-}depth\text{-}set n$
if $set (coeffs f) \subseteq global\text{-}depth\text{-}set 0$
and $set (coeffs g) \subseteq global\text{-}depth\text{-}set n$
 ⟨proof⟩

end

locale *p-equality-depth-no-zero-divisors-1 =*

p-equality-no-zero-divisors-1 + p-equality-depth-no-zero-divisors
begin

lemma *depth-of-1[simp]*: $p\text{-depth } p \ 1 = 0$
<proof>

lemma *depth-of-neg1*: $p\text{-depth } p \ (-1) = 0$
<proof>

lemma *depth-pow-additive*: $p\text{-depth } p \ (x \wedge n) = \text{int } n * p\text{-depth } p \ x$
<proof>

lemma *depth-prod-summative*:
 $\text{finite } X \implies \neg p\text{-equal } p \ (\prod X) \ 0 \implies$
 $p\text{-depth } p \ (\prod X) = \text{sum } (p\text{-depth } p) \ X$
<proof>

lemma *p-depth-set-1*: $1 \in p\text{-depth-set } p \ 0$
<proof>

lemma *pos-p-depth-set-1*: $n > 0 \implies 1 \notin p\text{-depth-set } p \ n$
<proof>

lemma *global-depth-set-1*: $1 \in \text{global-depth-set } 0$
<proof>

lemma *pos-global-depth-set-1*: $n > 0 \implies 1 \notin \text{global-depth-set } n$
<proof>

lemma *p-depth-set-neg1*: $-1 \in p\text{-depth-set } p \ 0$
<proof>

lemma *global-depth-set-neg1*: $-1 \in \text{global-depth-set } 0$
<proof>

lemma *p-depth-set-pow*:
 $x \wedge \text{Suc } k \in p\text{-depth-set } p \ n$ **if** $n \geq 0$ **and** $x \in p\text{-depth-set } p \ n$
<proof>

lemma *global-depth-set-pow*:
 $x \wedge \text{Suc } k \in \text{global-depth-set } n$ **if** $n \geq 0$ **and** $x \in \text{global-depth-set } n$
<proof>

lemma *p-depth-set-0-pow*: $x \wedge k \in p\text{-depth-set } p \ 0$ **if** $x \in p\text{-depth-set } p \ 0$
<proof>

lemma *global-depth-set-0-pow*: $x \wedge k \in \text{global-depth-set } 0$ **if** $x \in \text{global-depth-set } 0$
<proof>

lemma *p-depth-set-of-nat*: $\text{of-nat } n \in p\text{-depth-set } p \ 0$
<proof>

lemma *global-depth-set-of-nat*: $of\text{-}nat\ n \in global\text{-}depth\text{-}set\ 0$
(*proof*)

lemma *p-depth-set-of-int*: $of\text{-}int\ n \in p\text{-}depth\text{-}set\ p\ 0$
(*proof*)

lemma *global-depth-set-of-int*: $of\text{-}int\ n \in global\text{-}depth\text{-}set\ 0$
(*proof*)

lemma *p-depth-set-polyderiv*:
 $set\ (coeffs\ f) \subseteq p\text{-}depth\text{-}set\ p\ n \implies$
 $set\ (coeffs\ (polyderiv\ f)) \subseteq p\text{-}depth\text{-}set\ p\ n$
(*proof*)

lemma *global-depth-set-polyderiv*:
 $set\ (coeffs\ (polyderiv\ f)) \subseteq global\text{-}depth\text{-}set\ n$
if $set\ (coeffs\ f) \subseteq global\text{-}depth\text{-}set\ n$
(*proof*)

lemma *poly-p-depth-set-polyderiv*:
 $poly\ (polyderiv\ f)\ x \in p\text{-}depth\text{-}set\ p\ n$
if $n \geq 0$ **and** $set\ (coeffs\ f) \subseteq p\text{-}depth\text{-}set\ p\ n$ **and** $x \in p\text{-}depth\text{-}set\ p\ n$
(*proof*)

lemma *poly-global-depth-set-polyderiv*:
 $poly\ (polyderiv\ f)\ x \in global\text{-}depth\text{-}set\ n$
if $n \geq 0$
and $set\ (coeffs\ f) \subseteq global\text{-}depth\text{-}set\ n$
and $x \in global\text{-}depth\text{-}set\ n$
(*proof*)

lemma *p-set-depth-additive-poly-poly*:
 $set\ (coeffs\ f) \subseteq p\text{-}depth\text{-}set\ p\ m \implies$
 $set\ (coeffs\ (coeff\ (additive\text{-}poly\text{-}poly\ f)\ n)) \subseteq p\text{-}depth\text{-}set\ p\ m$
if $m \geq 0$
(*proof*)

lemma *global-depth-set-additive-poly-poly*:
 $set\ (coeffs\ (coeff\ (additive\text{-}poly\text{-}poly\ f)\ n)) \subseteq global\text{-}depth\text{-}set\ m$
if $m \geq 0$ **and** $set\ (coeffs\ f) \subseteq global\text{-}depth\text{-}set\ m$
(*proof*)

lemma *poly-p-depth-set-additive-poly-poly*:
 $poly\ (coeff\ (additive\text{-}poly\text{-}poly\ f)\ n)\ x \in p\text{-}depth\text{-}set\ p\ m$
if $m \geq 0$ **and** $set\ (coeffs\ f) \subseteq p\text{-}depth\text{-}set\ p\ m$ **and** $x \in p\text{-}depth\text{-}set\ p\ m$
(*proof*)

lemma *poly-global-depth-set-additive-poly-poly*:
 $poly\ (coeff\ (additive\text{-}poly\text{-}poly\ f)\ n)\ x \in global\text{-}depth\text{-}set\ m$

if $m \geq 0$
and $\text{set}(\text{coeffs } f) \subseteq \text{global-depth-set } m$
and $x \in \text{global-depth-set } m$
 ⟨proof⟩

lemma *sum-poly-additive-poly-poly-depth-bound*:
fixes $p :: 'a$ **and** $f :: 'b$ *poly* **and** $x \ y :: 'b$ **and** $a \ b \ n :: \text{nat}$
defines $g \equiv \lambda i. \text{poly}(\text{coeff}(\text{additive-poly-poly } f)(\text{Suc } i)) \ x * y \wedge (\text{Suc } i)$
defines $S \equiv \text{sum } g \ \{a..b\}$
assumes $\text{coeffs}: \text{set}(\text{coeffs } f) \subseteq \text{p-depth-set } p \ 0$
and $\text{arg} : x \in \text{p-depth-set } p \ 0$
and $\text{sum} : \neg \text{p-equal } p \ S \ 0$
obtains j **where** $j \in \{a..b\}$ **and** $\text{p-depth } p \ S \geq \text{int}(\text{Suc } j) * \text{p-depth } p \ y$
 ⟨proof⟩

end

3.5.2 Depth of inverses and division

locale *p-equality-depth-div-inv* =
 $\text{p-equality-div-inv} + \text{p-equality-depth-no-zero-divisors-1}$
begin

lemma *inverse-depth*: $\text{p-depth } p (\text{inverse } x) = - \text{p-depth } p \ x$
 ⟨proof⟩

lemma *depth-powi-additive*: $\text{p-depth } p (\text{power-int } x \ n) = n * \text{p-depth } p \ x$
 ⟨proof⟩

lemma *p-depth-set-inverse*: $\text{inverse } x \in \text{p-depth-set } p \ 0$ **if** $\text{p-depth } p \ x = 0$
 ⟨proof⟩

lemma *divide-depth*: $\text{p-depth } p (x / y) = \text{p-depth } p \ x - \text{p-depth } p \ y$ **if** $\neg \text{p-equal } p (x / y) \ 0$
 ⟨proof⟩

lemma *p-depth-set-divide*:
 $x / y \in \text{p-depth-set } p \ n$ **if** $x \in \text{p-depth-set } p \ n$ **and** $\text{p-depth } p \ y = 0$
 ⟨proof⟩

lemma *p-depth-poly-deriv-quotient*:
 $\text{p-depth } p ((\text{poly } f \ a) / (\text{poly}(\text{polyderiv } f) \ a)) \geq n$
if $n \geq 0$
and $\text{set}(\text{coeffs } f) \subseteq \text{p-depth-set } p \ n$
and $a \in \text{p-depth-set } p \ n$
and $\neg \text{p-equal } p (\text{poly } f \ a) \ 0$
and $\neg \text{p-equal } p (\text{poly}(\text{polyderiv } f) \ a) \ 0$
and $\text{p-depth } p (\text{poly } f \ a) \geq 2 * \text{p-depth } p (\text{poly}(\text{polyderiv } f) \ a)$
 ⟨proof⟩

lemma *p-depth-set-poly-deriv-quotient*:
 $(poly\ f\ a) / (poly\ (polyderiv\ f)\ a) \in p\text{-depth-set}\ p\ n$
if $n \geq 0$
and $set\ (coeffs\ f) \subseteq p\text{-depth-set}\ p\ n$
and $a \in p\text{-depth-set}\ p\ n$
and $\neg\ p\text{-equal}\ p\ (poly\ f\ a)\ 0$
and $\neg\ p\text{-equal}\ p\ (poly\ (polyderiv\ f)\ a)\ 0$
and $p\text{-depth}\ p\ (poly\ f\ a) \geq 2 * p\text{-depth}\ p\ (poly\ (polyderiv\ f)\ a)$
 $\langle proof \rangle$

end

3.6 Global equality and restriction of places

3.6.1 Locales

locale *global-p-equality = p-equality*
 $+$
fixes $p\text{-restrict} :: 'b::comm\text{-ring} \Rightarrow ('a \Rightarrow bool) \Rightarrow 'b$
assumes $p\text{-restrict-equiv} : P\ p \Longrightarrow p\text{-equal}\ p\ (p\text{-restrict}\ x\ P)\ x$
and $p\text{-restrict-equiv0} : \neg\ P\ p \Longrightarrow p\text{-equal}\ p\ (p\text{-restrict}\ x\ P)\ 0$
begin

lemma *p-restrict-equiv-sym*: $P\ p \Longrightarrow p\text{-equal}\ p\ x\ (p\text{-restrict}\ x\ P)$
 $\langle proof \rangle$

lemma *p-restrict-equiv0-iff*:
 $p\text{-equal}\ p\ (p\text{-restrict}\ x\ P)\ 0 \longleftrightarrow p\text{-equal}\ p\ x\ 0 \vee \neg\ P\ p$
 $\langle proof \rangle$

lemma *p-restrict-restrict-equiv-conj*:
 $p\text{-equal}\ p\ (p\text{-restrict}\ (p\text{-restrict}\ x\ P)\ Q)\ (p\text{-restrict}\ x\ (\lambda p. P\ p \wedge Q\ p))$
 $\langle proof \rangle$

lemma *p-restrict-restrict-equiv*: $p\text{-equal}\ p\ (p\text{-restrict}\ (p\text{-restrict}\ x\ P)\ P)\ (p\text{-restrict}\ x\ P)$
 $\langle proof \rangle$

lemma *p-restrict-0-equiv0*: $p\text{-equal}\ p\ (p\text{-restrict}\ 0\ P)\ 0$
 $\langle proof \rangle$

lemma *p-restrict-add-equiv*: $p\text{-equal}\ p\ (p\text{-restrict}\ (x + y)\ P)\ (p\text{-restrict}\ x\ P + p\text{-restrict}\ y\ P)$
 $\langle proof \rangle$

lemma *p-restrict-add-mixed-equiv*:
 $p\text{-equal}\ p\ (p\text{-restrict}\ x\ P + p\text{-restrict}\ y\ Q)\ (x + y)$ **if** $P\ p$ **and** $Q\ p$
 $\langle proof \rangle$

lemma *p-restrict-add-mixed-equiv-drop-right:*

p-equal p (p-restrict x P + p-restrict y Q) x if P p and \neg Q p

<proof>

lemma *p-restrict-add-mixed-equiv-drop-left:*

p-equal p (p-restrict x P + p-restrict y Q) y if \neg P p and Q p

<proof>

lemma *p-restrict-add-mixed-equiv-drop-both:*

p-equal p (p-restrict x P + p-restrict y Q) 0 if \neg P p and \neg Q p

<proof>

lemma *p-restrict-uminus-equiv:* *p-equal p (p-restrict (-x) P) (- p-restrict x P)*

<proof>

lemma *p-restrict-minus-equiv:* *p-equal p (p-restrict (x - y) P) (p-restrict x P - p-restrict y P)*

<proof>

lemma *p-restrict-minus-mixed-equiv:*

p-equal p (p-restrict x P - p-restrict y Q) (x - y) if P p and Q p

<proof>

lemma *p-restrict-minus-mixed-equiv-drop-right:*

p-equal p (p-restrict x P - p-restrict y Q) x if P p and \neg Q p

<proof>

lemma *p-restrict-minus-mixed-equiv-drop-left:*

p-equal p (p-restrict x P - p-restrict y Q) (-y) if \neg P p and Q p

<proof>

lemma *p-restrict-minus-mixed-equiv-drop-both:*

p-equal p (p-restrict x P - p-restrict y Q) 0 if \neg P p and \neg Q p

<proof>

lemma *p-restrict-times-equiv:*

*p-equal p ((p-restrict x P) * (p-restrict y Q))*

*(p-restrict (x * y) ($\lambda p. P p \wedge Q p$))*

<proof>

lemma *p-restrict-times-equiv':*

*p-equal p ((p-restrict x P) * (p-restrict y P)) (p-restrict (x * y) P)*

<proof>

lemma *p-restrict-p-equalI:*

p-equal p (p-restrict x P) y

if *P p \longrightarrow p-equal p x y*

and *\neg P p \longrightarrow p-equal p y 0*

<proof>

lemma *p-restrict-p-equal-p-restrictI*:
 $p\text{-equal } p \text{ (p-restrict } x \ P) \text{ (p-restrict } y \ Q)$
if $(P \ p \wedge Q \ p) \longrightarrow p\text{-equal } p \ x \ y$
and $(P \ p \wedge \neg Q \ p) \longrightarrow p\text{-equal } p \ x \ 0$
and $(\neg P \ p \wedge Q \ p) \longrightarrow p\text{-equal } p \ y \ 0$
 $\langle \text{proof} \rangle$

lemma *p-restrict-p-equal-p-restrict-by-sameI*:
 $p\text{-equal } p \text{ (p-restrict } x \ P) \text{ (p-restrict } y \ P) \text{ if } P \ p \longrightarrow p\text{-equal } p \ x \ y$
 $\langle \text{proof} \rangle$

end

locale *global-p-equality-div-inv* =
 $p\text{-equality-div-inv } p\text{-equal} + \text{global-p-equality } p\text{-equal } p\text{-restrict}$
for $p\text{-equal} \quad ::$
 $'a \Rightarrow 'b :: \{\text{comm-ring-1, inverse, divide-trivial}\} \Rightarrow$
 $'b \Rightarrow \text{bool}$
and $p\text{-restrict} :: 'b \Rightarrow ('a \Rightarrow \text{bool}) \Rightarrow 'b$
begin

lemma *inverse-p-restrict-equiv*: $p\text{-equal } p \text{ (inverse (p-restrict } x \ P)) \text{ (p-restrict (inverse } x) \ P)$
 $\langle \text{proof} \rangle$

end

locale *global-p-equal* = global-p-equality
 $+ \text{assumes } \text{global-imp-eq}: \text{globally-p-equal } x \ y \Longrightarrow x = y$
begin

lemma *global-eq-iff*: $\text{globally-p-equal } x \ y \longleftrightarrow x = y$
 $\langle \text{proof} \rangle$

lemma *p-restrict-true[simp]*: $p\text{-restrict } x \ (\lambda x. \ \text{True}) = x$
 $\langle \text{proof} \rangle$

lemma *p-restrict-false[simp]*: $p\text{-restrict } x \ (\lambda x. \ \text{False}) = 0$
 $\langle \text{proof} \rangle$

lemma *p-restrict-restrict*:
 $p\text{-restrict } (p\text{-restrict } x \ P) \ Q = p\text{-restrict } x \ (\lambda p. \ P \ p \wedge Q \ p)$
 $\langle \text{proof} \rangle$

lemma *p-restrict-restrict'[simp]*: $p\text{-restrict } (p\text{-restrict } x \ P) \ P = p\text{-restrict } x \ P$
 $\langle \text{proof} \rangle$

lemma *p-restrict-image-restrict*: $p\text{-restrict } x P = x$ **if** $x \in (\lambda z. p\text{-restrict } z P) \text{ ` } B$
 ⟨proof⟩

lemma *p-restrict-zero*: $p\text{-restrict } 0 P = 0$
 ⟨proof⟩

lemma *p-restrict-add*: $p\text{-restrict } (x + y) P = p\text{-restrict } x P + p\text{-restrict } y P$
 ⟨proof⟩

lemma *p-restrict-uminus*: $p\text{-restrict } (-x) P = - p\text{-restrict } x P$
 ⟨proof⟩

lemma *p-restrict-minus*: $p\text{-restrict } (x - y) P = p\text{-restrict } x P - p\text{-restrict } y P$
 ⟨proof⟩

lemma *p-restrict-times*:
 $(p\text{-restrict } x P) * (p\text{-restrict } y Q) = p\text{-restrict } (x * y) (\lambda p. P p \wedge Q p)$
 ⟨proof⟩

lemma *times-p-restrict*: $x * (p\text{-restrict } y P) = p\text{-restrict } (x * y) P$
 ⟨proof⟩

lemmas *p-restrict-pre-finite-adele-simps* =
p-restrict-zero p-restrict-add p-restrict-uminus p-restrict-minus p-restrict-times

lemma *p-equal-0-iff-p-restrict-eq-0*: $p\text{-equal } p x 0 \longleftrightarrow p\text{-restrict } x ((=) p) = 0$
 ⟨proof⟩

lemma *p-equal-iff-p-restrict-eq*:
 $p\text{-equal } p x y \longleftrightarrow p\text{-restrict } x ((=) p) = p\text{-restrict } y ((=) p)$
 ⟨proof⟩

lemma *p-restrict-eqI*:
 $y = p\text{-restrict } x P$
if $P: \bigwedge p. P p \implies p\text{-equal } p y x$
and $\text{not-}P: \bigwedge p. \neg P p \implies p\text{-equal } p y 0$
 ⟨proof⟩

lemma *p-restrict-eq-p-restrict-mixedI*:
 $p\text{-restrict } x P = p\text{-restrict } y Q$
if $\text{both: } \bigwedge p::'a. P p \implies Q p \implies p\text{-equal } p x y$
and $P : \bigwedge p::'a. P p \implies \neg Q p \implies p\text{-equal } p x 0$
and $Q : \bigwedge p::'a. \neg P p \implies Q p \implies p\text{-equal } p y 0$
 ⟨proof⟩

lemma *p-restrict-eq-p-restrictI*:
 $p\text{-restrict } x P = p\text{-restrict } y P$ **if** $\bigwedge p::'a. P p \implies p\text{-equal } p x y$
 ⟨proof⟩

lemma *p-restrict-eq-p-restrict-iff*:

$p\text{-restrict } x P = p\text{-restrict } y P \longleftrightarrow$
 $(\forall p::'a. P p \longrightarrow p\text{-equal } p x y)$
<proof>

lemma *p-restrict-eq-zero-iff*:

$p\text{-restrict } x P = 0 \longleftrightarrow (\forall p. P p \longrightarrow p\text{-equal } p x 0)$
<proof>

lemma *p-restrict-decomp*:

$x = p\text{-restrict } x P + p\text{-restrict } x (\lambda p. \neg P p)$
<proof>

lemma *restrict-complement*:

$\text{range } (\lambda x. p\text{-restrict } x P) + \text{range } (\lambda x. p\text{-restrict } x (\lambda p. \neg P p))$
 $= \text{UNIV}$
<proof>

end

locale *global-p-equal-no-zero-divisors* =

global-p-equal p-equal p-restrict + p-equality-no-zero-divisors p-equal
for *p-equal* :: *'a* \Rightarrow *'b::comm-ring* \Rightarrow *'b* \Rightarrow *bool*
and *p-restrict* :: *'b* \Rightarrow (*'a* \Rightarrow *bool*) \Rightarrow *'b*

locale *global-p-equal-no-zero-divisors-1* =

global-p-equal-no-zero-divisors p-equal p-restrict + p-equality-no-zero-divisors-1
p-equal
for *p-equal* :: *'a* \Rightarrow *'b::comm-ring-1* \Rightarrow *'b* \Rightarrow *bool*
and *p-restrict* :: *'b* \Rightarrow (*'a* \Rightarrow *bool*) \Rightarrow *'b*

begin

lemma *pow-eq-0-iff*: $x \wedge n = 0 \longleftrightarrow x = 0 \wedge n \neq 0$ **for** *x* :: *'b*

<proof>

end

locale *global-p-equal-div-inv* =

global-p-equality-div-inv p-equal p-restrict
 $+ \text{global-p-equal p-equal p-restrict}$
for *p-equal* ::
'a \Rightarrow *'b::\{comm-ring-1, inverse, divide-trivial\}* \Rightarrow
'b \Rightarrow *bool*
and *p-restrict* :: *'b* \Rightarrow (*'a* \Rightarrow *bool*) \Rightarrow *'b*
begin

lemma *power-int-eq-0-iff*:

$power-int\ x\ n = 0 \iff x = 0 \wedge n \neq 0$ **for** $x :: 'b$
<proof>

lemma *inverse-eq0-iff*: $inverse\ x = 0 \iff x = 0$ **for** $x :: 'b$

<proof>

lemma *inverse-neg-1* [*simp*]: $inverse\ (-1::'b) = -1$

<proof>

lemma *inverse-uminus*: $inverse\ (-x) = -\ inverse\ x$ **for** $x :: 'b$

<proof>

lemma *global-right-inverse*: $x * inverse\ x = 1$ **if** $\forall p. \neg p-equal\ p\ x\ 0$

<proof>

lemma *right-inverse*: $x * inverse\ x = p-restrict\ 1\ (\lambda p. \neg p-equal\ p\ x\ 0)$

<proof>

lemma *global-left-inverse*: $inverse\ x * x = 1$ **if** $\forall p. \neg p-equal\ p\ x\ 0$

<proof>

lemma *left-inverse*: $inverse\ x * x = p-restrict\ 1\ (\lambda p. \neg p-equal\ p\ x\ 0)$

<proof>

lemma *inverse-unique*: $inverse\ x = y$ **if** $x * y = 1$ **for** $x\ y :: 'b$

<proof>

lemma *inverse-p-restrict*: $inverse\ (p-restrict\ x\ P) = p-restrict\ (inverse\ x)\ P$

<proof>

lemma *power-diff-conv-inverse*:

$x \wedge (n - m) = x \wedge n * inverse\ x \wedge m$ **if** $m < n$
for $x :: 'b$

<proof>

lemma *power-int-add-1*: $power-int\ x\ (m + 1) = power-int\ x\ m * x$ **if** $m \neq -1$ **for**

$x :: 'b$

<proof>

lemma *power-int-add*:

$power-int\ x\ (m + n) = power-int\ x\ m * power-int\ x\ n$

if $(\forall p. \neg p-equal\ p\ x\ 0) \vee m + n \neq 0$ **for** $x :: 'b$

<proof>

lemma *power-int-diff*:

$power-int\ x\ (m - n) = power-int\ x\ m / power-int\ x\ n$ **if** $m \neq n$ **for** $x :: 'b$

<proof>

lemma *power-int-minus-mult*: $\text{power-int } x (n - 1) * x = \text{power-int } x n$ **if** $n \neq 0$
for $x :: 'b$

<proof>

lemma *power-int-not-eq-zero*:

$x \neq 0 \vee n = 0 \implies \text{power-int } x n \neq 0$ **for** $x :: 'b$

<proof>

lemma *minus-divide-right*: $a / (-b) = -(a / b)$ **for** $a b :: 'b$

<proof>

lemma *minus-divide-divide*: $(-a) / (-b) = a / b$ **for** $a b :: 'b$

<proof>

lemma *divide-minus1 [simp]*: $x / (-1) = -x$ **for** $x :: 'b$

<proof>

lemma *divide-self*: $x / x = p\text{-restrict } 1$ ($\lambda p. \neg p\text{-equal } p x 0$)

<proof>

lemma *global-divide-self*: $x / x = 1$ **if** $\forall p. \neg p\text{-equal } p x 0$

<proof>

lemma *global-mult-divide-mult-cancel-right*:

$(a * b) / (c * b) = a / c$ **if** $\forall p. \neg p\text{-equal } p b 0$

<proof>

lemma *global-mult-divide-mult-cancel-left*:

$(c * a) / (c * b) = a / b$ **if** $\forall p. \neg p\text{-equal } p c 0$

<proof>

lemma *divide-p-restrict-left*: $(p\text{-restrict } x P) / y = p\text{-restrict } (x / y) P$

<proof>

lemma *divide-p-restrict-right*: $x / (p\text{-restrict } y P) = p\text{-restrict } (x / y) P$

<proof>

lemma *inj-divide-right*:

$\text{inj } (\lambda b. b / a) \longleftrightarrow (\forall p. \neg p\text{-equal } p a 0)$

<proof>

end

locale *global-p-equality-depth* = *global-p-equality* + *p-equality-depth*

begin

lemma *globally-p-equal-p-depth*: $\text{globally-p-equal } x y \implies p\text{-depth } p x = p\text{-depth } p y$

y
<proof>

lemma *p-restrict-depth*:

$P \ p \implies p\text{-depth } p \ (p\text{-restrict } x \ P) = p\text{-depth } p \ x$
 $\neg \ P \ p \implies p\text{-depth } p \ (p\text{-restrict } x \ P) = 0$
<proof>

lemma *p-restrict-add-mixed-depth-drop-right*:

$p\text{-depth } p \ (p\text{-restrict } x \ P + p\text{-restrict } y \ Q) = p\text{-depth } p \ x$ **if** $P \ p$ **and** $\neg \ Q \ p$
<proof>

lemma *p-restrict-add-mixed-depth-drop-left*:

$p\text{-depth } p \ (p\text{-restrict } x \ P + p\text{-restrict } y \ Q) = p\text{-depth } p \ y$ **if** $\neg \ P \ p$ **and** $Q \ p$
<proof>

lemma *p-depth-set-p-restrict*:

$p\text{-restrict } x \ P \in p\text{-depth-set } p \ n$ **if** $P \ p \implies x \in p\text{-depth-set } p \ n$
<proof>

lemma *global-depth-set-p-restrict*:

$p\text{-restrict } x \ P \in \text{global-depth-set } n$ **if** $x \in \text{global-depth-set } n$
<proof>

lemma *p-depth-set-closed-under-p-restrict-image*:

$(\lambda x. p\text{-restrict } x \ P) \ ` \ p\text{-depth-set } p \ n \subseteq p\text{-depth-set } p \ n$
<proof>

lemma *global-depth-set-closed-under-p-restrict-image*:

$(\lambda x. p\text{-restrict } x \ P) \ ` \ \text{global-depth-set } n \subseteq \text{global-depth-set } n$
<proof>

lemma *global-depth-set-p-restrictI*:

$p\text{-restrict } x \ P \in \text{global-depth-set } n$
if $\bigwedge p. P \ p \implies x \in p\text{-depth-set } p \ n$
<proof>

lemma *p-depth-set-image-under-one-p-restrict-image*:

$(\lambda x. p\text{-restrict } x \ ((=) \ p)) \ ` \ p\text{-depth-set } p \ n \subseteq \text{global-depth-set } n$
<proof>

end

locale *global-p-equality-depth-no-zero-divisors-1* =

p-equality-depth-no-zero-divisors-1 *p-equal* *p-depth*
+ *global-p-equality-depth* *p-equal* *p-restrict* *p-depth*
for *p-equal* $::$
 $'a \Rightarrow 'b::\text{comm-ring-1} \Rightarrow 'b \Rightarrow \text{bool}$

```

and p-restrict :: 'b ⇒ ('a ⇒ bool) ⇒ 'b
and p-depth    :: 'a ⇒ 'b ⇒ int

locale global-p-equality-depth-div-inv =
  p-equality-depth-div-inv + global-p-equality-depth
begin

sublocale global-p-equality-depth-div-inv ⟨proof⟩

lemma global-depth-set-inverse:
  p-restrict (inverse x) (λp. p-depth p x = 0) ∈ global-depth-set 0
  ⟨proof⟩

lemma global-depth-set-divide:
  p-restrict (x / y) (λp. p-depth p y = 0) ∈ global-depth-set n
  if x ∈ global-depth-set n
  ⟨proof⟩

lemma global-depth-set-divideI:
  p-restrict (x / y) (λp. p-depth p y = 0) ∈ global-depth-set n
  if ∧p. p-depth p y = 0 ⇒ x ∈ p-depth-set p n
  ⟨proof⟩

end

locale global-p-equal-depth = global-p-equal + global-p-equality-depth
begin

lemma p-depth-set-vs-global-depth-set-image-under-one-p-restrict-image:
  (λx. p-restrict x ((=) p)) ' p-depth-set p n =
  (λx. p-restrict x ((=) p)) ' global-depth-set n
  ⟨proof⟩

end

locale global-p-equal-depth-no-zero-divisors =
  global-p-equal-no-zero-divisors + p-equality-depth-no-zero-divisors
begin
sublocale global-p-equal-depth ⟨proof⟩
end

locale global-p-equal-depth-div-inv =
  global-p-equality-depth-div-inv + global-p-equal-depth-no-zero-divisors
begin
sublocale global-p-equal-div-inv ⟨proof⟩

```

lemma *p-depth-set-eq-coset*:

*p-depth-set p n = (w powi n) *o (p-depth-set p 0)*
if *p-depth p w = 1* **and** $\forall p. \neg p\text{-equal } p \ w \ 0$
<proof>

lemma *global-depth-set-eq-coset*:

*global-depth-set n = (w powi n) *o (global-depth-set 0)* **if** $\forall p. p\text{-depth } p \ w = 1$
<proof>

lemma *p-depth-set-eq-coset'*:

*p-depth-set p 0 = (w powi (-n)) *o (p-depth-set p n)*
if *p-depth p w = 1* **and** $\forall p. \neg p\text{-equal } p \ w \ 0$
<proof>

lemma *global-depth-set-eq-coset'*:

*global-depth-set 0 = (w powi (-n)) *o (global-depth-set n)* **if** $\forall p. p\text{-depth } p \ w = 1$
<proof>

end

3.6.2 Embedding of base type constants

locale *global-p-equality-w-consts = global-p-equality p-equal p-restrict*

for *p-equal* **::**

'a ⇒ 'b::comm-ring-1 ⇒ 'b ⇒ bool

and *p-restrict* **::** *'b ⇒ ('a ⇒ bool) ⇒ 'b*

+

fixes *func-of-consts* **::** *('a ⇒ 'c::ring-1) ⇒ 'b*

assumes *consts-1* *[simp]: func-of-consts 1 = 1*

and *consts-diff* *[simp]: func-of-consts (f - g) = func-of-consts f - func-of-consts g*

and *consts-mult* *[simp]:*

*func-of-consts (f * g) = func-of-consts f * func-of-consts g*

begin

abbreviation *consts* \equiv *func-of-consts*

abbreviation *const a* \equiv *consts* ($\lambda p. a$)

lemma *consts-0* *[simp]: consts 0 = 0*

<proof>

lemma *const-0* *[simp]: const 0 = 0*

<proof>

lemma *const-1* *[simp]: const 1 = 1*

<proof>

lemma *consts-uminus* [*simp*]: $\text{consts } (- f) = - \text{consts } f$
<proof>

lemma *const-uminus* [*simp*]: $\text{const } (- a) = - \text{const } a$
<proof>

lemma *const-diff* [*simp*]: $\text{const } (a - b) = \text{const } a - \text{const } b$
<proof>

lemma *consts-add* [*simp*]: $\text{consts } (f + g) = \text{consts } f + \text{consts } g$
<proof>

lemma *const-add* [*simp*]: $\text{const } (a + b) = \text{const } a + \text{const } b$
<proof>

lemma *const-mult* [*simp*]: $\text{const } (a * b) = \text{const } a * \text{const } b$
<proof>

lemma *const-of-nat*: $\text{const } (\text{of-nat } n) = \text{of-nat } n$
<proof>

lemma *const-of-int*: $\text{const } (\text{of-int } z) = \text{of-int } z$
<proof>

end

locale *global-p-equal-w-consts* =
 global-p-equal p-equal p-restrict
+ *global-p-equality-w-consts p-equal p-restrict func-of-consts*
for *p-equal* ::
 '*a* \Rightarrow '*b*::*comm-ring-1* \Rightarrow '*b* \Rightarrow *bool*
and *p-restrict* :: '*b* \Rightarrow ('*a* \Rightarrow *bool*) \Rightarrow '*b*
and *func-of-consts* :: ('*a* \Rightarrow '*c*::*ring-1*) \Rightarrow '*b*

locale *global-p-equality-w-inj-consts* = *global-p-equality-w-consts*
+
assumes *p-equal-func-of-consts-0*: $p\text{-equal } p (\text{consts } f) 0 \iff f p = 0$
begin

lemmas *p-equal-func-of-const-0* = *p-equal-func-of-consts-0*[*of - λp. -*]

lemma *p-equal-func-of-consts*:
 $p\text{-equal } p (\text{consts } f) (\text{consts } g) \iff f p = g p$
<proof>

lemma *globally-p-equal-func-of-consts*:
 $\text{globally-p-equal } (\text{func-of-consts } f) (\text{func-of-consts } g) \iff$

$(\forall p. f p = g p)$
<proof>

lemma *const-p-equal*: $p\text{-equal } p \text{ (const } a \text{) (const } b \text{)} \implies a = b$
<proof>

end

locale *global-p-equality-w-inj-consts-char-0* =
 global-p-equality-w-inj-consts p-equal p-restrict func-of-consts
 for *p-equal* ::
 'a \Rightarrow *'b::comm-ring-1* \Rightarrow *'b* \Rightarrow *bool*
 and *p-restrict* :: *'b* \Rightarrow (*'a* \Rightarrow *bool*) \Rightarrow *'b*
 and *func-of-consts* :: (*'a* \Rightarrow *'c::ring-char-0*) \Rightarrow *'b*
begin

lemma *const-of-nat-p-equal-0-iff*: $p\text{-equal } p \text{ (of-nat } n \text{) } 0 \longleftrightarrow n = 0$
<proof>

lemma *const-of-int-p-equal-0-iff*: $p\text{-equal } p \text{ (of-int } z \text{) } 0 \longleftrightarrow z = 0$
<proof>

end

locale *global-p-equality-div-inv-w-inj-consts* =
 p-equality-div-inv + global-p-equality-w-inj-consts
begin

lemma *const-divide-self-equiv*: $p\text{-equal } p \text{ ((const } f \text{) / (const } f \text{)) } 1 \text{ if } f p \neq 0$
<proof>

lemma *const-divide-self-equiv*: $p\text{-equal } p \text{ ((const } c \text{) / (const } c \text{)) } 1 \text{ if } c \neq 0$
<proof>

end

locale *global-p-equal-w-inj-consts* =
 global-p-equal p-equal p-restrict
+ *global-p-equality-w-inj-consts p-equal p-restrict*
 for *p-equal* ::
 'a \Rightarrow *'b::comm-ring-1* \Rightarrow *'b* \Rightarrow *bool*
 and *p-restrict* :: *'b* \Rightarrow (*'a* \Rightarrow *bool*) \Rightarrow *'b*
begin

lemma *const-eq-iff*: $\text{const } f = \text{const } g \longleftrightarrow f = g$
<proof>

lemma *consts-eq-0-iff*: $\text{consts } f = 0 \longleftrightarrow f = 0$
<proof>

lemma *inj-consts*: $\text{inj } \text{consts}$
<proof>

lemma *const-eq-iff*: $\text{const } a = \text{const } b \longleftrightarrow a = b$
<proof>

lemma *const-eq-0-iff*: $\text{const } a = 0 \longleftrightarrow a = 0$
<proof>

lemma *inj-const*: $\text{inj } \text{const}$
<proof>

end

locale *global-p-equal-div-inv-w-inj-consts* =
global-p-equal-div-inv + *global-p-equal-w-inj-consts*
begin

sublocale *global-p-equality-div-inv-w-inj-consts* *<proof>*

lemma *consts-divide-self*:
 $(\text{consts } f) / (\text{consts } f) = p\text{-restrict } 1 \ (\lambda p::'a. f \neq 0)$
<proof>

lemma *const-right-inverse* [*simp*]: $\text{const } a * \text{inverse } (\text{const } a) = 1 \ \text{if } a \neq 0$
<proof>

lemma *const-left-inverse* [*simp*]: $\text{inverse } (\text{const } a) * \text{const } a = 1 \ \text{if } a \neq 0$
<proof>

lemma *const-divide-self*: $(\text{const } c) / (\text{const } c) = 1 \ \text{if } c \neq 0$
<proof>

lemma *mult-const-divide-mult-const-cancel-right*:
 $(y * \text{const } a) / (z * \text{const } a) = y / z \ \text{if } a \neq 0$
<proof>

lemma *mult-const-divide-mult-const-cancel-left*:
 $(\text{const } a * y) / (\text{const } a * z) = y / z \ \text{if } a \neq 0$
<proof>

lemma *mult-const-divide-mult-const-cancel-left-right*:
 $(\text{const } a * y) / (z * \text{const } a) = y / z \ \text{if } a \neq 0$
<proof>

lemma *mult-const-divide-mult-const-cancel-right-left*:

$(y * \text{const } a) / (\text{const } a * z) = y / z$ **if** $a \neq 0$
<proof>

end

locale *global-p-equal-div-inv-w-inj-consts-char-0* =

global-p-equality-w-inj-consts-char-0 p-equal p-restrict func-of-consts
+ *global-p-equal-div-inv-w-inj-consts p-equal p-restrict func-of-consts*
for *p-equal* ::
 $'a \Rightarrow 'b :: \{\text{comm-ring-1, ring-char-0, inverse, divide-trivial}\} \Rightarrow$
 $'b \Rightarrow \text{bool}$
and *p-restrict* :: $'b \Rightarrow ('a \Rightarrow \text{bool}) \Rightarrow 'b$
and *func-of-consts* :: $('a \Rightarrow 'c :: \text{ring-char-0}) \Rightarrow 'b$
begin

context

fixes $n :: \text{nat}$
assumes $n \neq 0$

begin

lemma *left-inverse-const-of-nat[simp]*: $\text{inverse } (\text{of-nat } n :: 'b) * \text{of-nat } n = 1$
<proof>

lemma *right-inverse-const-of-nat[simp]*: $\text{of-nat } n * \text{inverse } (\text{of-nat } n :: 'b) = 1$
<proof>

lemma *divide-self-const-of-nat[simp]*: $(\text{of-nat } n :: 'b) / \text{of-nat } n = 1$
<proof>

lemma *mult-const-of-nat-divide-const-mult-of-nat-cancel-right*:

$(a * \text{of-nat } n) / (b * \text{of-nat } n) = a / b$ **for** $a \ b :: 'b$
<proof>

lemma *mult-const-of-nat-divide-const-mult-of-nat-cancel-left*:

$(\text{of-nat } n * a) / (\text{of-nat } n * b) = a / b$ **for** $a \ b :: 'b$
<proof>

end

context

fixes $z :: \text{int}$
assumes $z \neq 0$

begin

lemma *left-inverse-const-of-int[simp]*: $\text{inverse } (\text{of-int } z :: 'b) * \text{of-int } z = 1$
<proof>

lemma *right-inverse-const-of-int*[simp]: $\text{of-int } z * \text{inverse } (\text{of-int } z :: 'b) = 1$
⟨proof⟩

lemma *divide-self-const-of-int*[simp]: $(\text{of-int } z :: 'b) / \text{of-int } z = 1$
⟨proof⟩

lemma *mult-const-of-int-divide-const-mult-of-int-cancel-right*:
 $(a * \text{of-int } z) / (b * \text{of-int } z) = a / b$ for $a \ b :: 'b$
⟨proof⟩

lemma *mult-const-of-int-divide-const-mult-of-int-cancel-left*:
 $(\text{of-int } z * a) / (\text{of-int } z * b) = a / b$ for $a \ b :: 'b$
⟨proof⟩

end

context

includes *rat.lifting*

begin

lift-definition *const-of-rat* :: $\text{rat} \Rightarrow 'b$
is $\lambda x. \text{of-int } (\text{fst } x) / \text{of-int } (\text{snd } x)$
⟨proof⟩

lemma *inj-const-of-rat*: $\text{inj } \text{const-of-rat}$
⟨proof⟩

lemma *const-of-rat-rat*: $\text{const-of-rat } (\text{Rat.Fract } a \ b) = (\text{of-int } a :: 'b) / \text{of-int } b$
⟨proof⟩

lemma *const-of-rat-0* [simp]: $\text{const-of-rat } 0 = 0$
⟨proof⟩

lemma *const-of-rat-1* [simp]: $\text{const-of-rat } 1 = 1$
⟨proof⟩

lemma *const-of-rat-minus*: $\text{const-of-rat } (- a) = - \text{const-of-rat } a$
⟨proof⟩

lemma *const-of-rat-add*: $\text{const-of-rat } (a + b) = \text{const-of-rat } a + \text{const-of-rat } b$
⟨proof⟩

lemma *const-of-rat-mult*: $\text{const-of-rat } (a * b) = \text{const-of-rat } a * \text{const-of-rat } b$
⟨proof⟩

lemma *const-of-rat-p-equal-0-iff*: $p\text{-equal } p (\text{const-of-rat } a) 0 \iff a = 0$
⟨proof⟩

end

lemma *const-of-rat-eq-0-iff* [simp]: $\text{const-of-rat } a = 0 \longleftrightarrow a = 0$
(proof)

lemma *const-of-rat-neg-one* [simp]: $\text{const-of-rat } (- 1) = -1$
(proof)

lemma *const-of-rat-diff*: $\text{const-of-rat } (a - b) = \text{const-of-rat } a - \text{const-of-rat } b$
(proof)

lemma *const-of-rat-sum*: $\text{const-of-rat } (\sum a \in A. f a) = (\sum a \in A. \text{const-of-rat } (f a))$
(proof)

lemma *const-of-rat-prod*: $\text{const-of-rat } (\prod a \in A. f a) = (\prod a \in A. \text{const-of-rat } (f a))$
(proof)

lemma *const-of-rat-inverse*: $\text{const-of-rat } (\text{inverse } a) = \text{inverse } (\text{const-of-rat } a)$
(proof)

lemma *left-inverse-const-of-rat*[simp]:
 $\text{inverse } (\text{const-of-rat } a) * \text{const-of-rat } a = 1$ if $a \neq 0$
(proof)

lemma *right-inverse-const-of-rat*[simp]:
 $\text{const-of-rat } a * \text{inverse } (\text{const-of-rat } a) = 1$ if $a \neq 0$
(proof)

lemma *const-of-rat-divide*: $\text{const-of-rat } (a / b) = \text{const-of-rat } a / \text{const-of-rat } b$
(proof)

lemma *divide-self-const-of-rat*[simp]: $(\text{const-of-rat } a) / (\text{const-of-rat } a) = 1$ if $a \neq 0$
(proof)

lemma *const-of-rat-power*: $\text{const-of-rat } (a \wedge n) = (\text{const-of-rat } a) \wedge n$
(proof)

lemma *const-of-rat-eq-iff* [simp]: $\text{const-of-rat } a = \text{const-of-rat } b \longleftrightarrow a = b$
(proof)

lemma *zero-eq-const-of-rat-iff* [simp]: $0 = \text{const-of-rat } a \longleftrightarrow 0 = a$
(proof)

lemma *const-of-rat-eq-1-iff* [simp]: $\text{const-of-rat } a = 1 \longleftrightarrow a = 1$
(proof)

lemma *one-eq-const-of-rat-iff* [simp]: $1 = \text{const-of-rat } a \longleftrightarrow 1 = a$
(proof)

Collapse nested embeddings.

lemma *const-of-rat-of-nat-eq* [simp]: *const-of-rat* (of-nat n) = of-nat n
<proof>

lemma *const-of-rat-of-int-eq* [simp]: *const-of-rat* (of-int z) = of-int z
<proof>

Product of all p-adic embeddings of the field of rationals.

definition *range-const-of-rat* :: 'b set ($\mathbb{Q}_{\forall p}$)
where $\mathbb{Q}_{\forall p} = \text{range } \text{const-of-rat}$

lemma *range-const-of-rat-cases* [cases set: *range-const-of-rat*]:
assumes $q \in \mathbb{Q}_{\forall p}$
obtains (const-of-rat) r where $q = \text{const-of-rat } r$
<proof>

lemma *range-const-of-rat-cases'*:
assumes $x \in \mathbb{Q}_{\forall p}$
obtains a b where $b > 0$ coprime a b $x = \text{of-int } a / \text{of-int } b$
<proof>

lemma *range-const-of-rat-of-rat* [simp]: *const-of-rat* $r \in \mathbb{Q}_{\forall p}$
<proof>

lemma *range-const-of-rat-of-int* [simp]: of-int $z \in \mathbb{Q}_{\forall p}$
<proof>

lemma *Ints-subset-range-const-of-rat*: $\mathbb{Z} \subseteq \mathbb{Q}_{\forall p}$
<proof>

lemma *range-const-of-rat-of-nat* [simp]: of-nat $n \in \mathbb{Q}_{\forall p}$
<proof>

lemma *Nats-subset-range-const-of-rat*: $\mathbb{N} \subseteq \mathbb{Q}_{\forall p}$
<proof>

lemma *range-const-of-rat-0* [simp]: $0 \in \mathbb{Q}_{\forall p}$
<proof>

lemma *range-const-of-rat-1* [simp]: $1 \in \mathbb{Q}_{\forall p}$
<proof>

lemma *range-const-of-rat-add* [simp]:
 $a \in \mathbb{Q}_{\forall p} \implies$
 $b \in \mathbb{Q}_{\forall p} \implies$
 $a + b \in \mathbb{Q}_{\forall p}$
<proof>

lemma *range-const-of-rat-minus-iff* [simp]:

$- a \in \mathbb{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathbb{Q}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *range-const-of-rat-diff* [*simp*]:

$a \in \mathbb{Q}_{\forall p} \implies$
 $b \in \mathbb{Q}_{\forall p} \implies$
 $a - b \in \mathbb{Q}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *range-const-of-rat-mult* [*simp*]:

$a \in \mathbb{Q}_{\forall p} \implies$
 $b \in \mathbb{Q}_{\forall p} \implies$
 $a * b \in \mathbb{Q}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *range-const-of-rat-inverse* [*simp*]:

$a \in \mathbb{Q}_{\forall p} \implies$
 $\text{inverse } a \in \mathbb{Q}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *range-const-of-rat-divide* [*simp*]:

$a \in \mathbb{Q}_{\forall p} \implies$
 $b \in \mathbb{Q}_{\forall p} \implies$
 $a / b \in \mathbb{Q}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *range-const-of-rat-power* [*simp*]:

$a \in \mathbb{Q}_{\forall p} \implies$
 $a \wedge n \in \mathbb{Q}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *range-const-of-rat-sum* [*intro*]:

$(\bigwedge x. x \in A \implies f x \in \mathbb{Q}_{\forall p}) \implies$
 $\text{sum } f A \in \mathbb{Q}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *range-const-of-rat-prod* [*intro*]:

$(\bigwedge x. x \in A \implies f x \in \mathbb{Q}_{\forall p})$
 $\implies \text{prod } f A \in \mathbb{Q}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *range-const-of-rat-induct* [*case-names const-of-rat, induct set: range-const-of-rat*]:

$q \in \mathbb{Q}_{\forall p} \implies$
 $(\bigwedge r. P (\text{const-of-rat } r)) \implies P q$
 $\langle \text{proof} \rangle$

lemma *p-adic-Rats-p-equal-0-iff*:

$a \in \mathbb{Q}_{\forall p} \implies$

$p\text{-equal } p \ a \ 0 \longleftrightarrow a = 0$
 ⟨proof⟩

lemma *range-const-of-rat-infinite*: $\neg \text{finite } \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *left-inverse-range-const-of-rat*:

$a \in \mathbb{Q}_{\forall p} \implies a \neq 0 \implies$
 $\text{inverse } a * a = 1$
 ⟨proof⟩

lemma *right-inverse-range-const-of-rat*:

$a \in \mathbb{Q}_{\forall p} \implies a \neq 0 \implies$
 $a * \text{inverse } a = 1$
 ⟨proof⟩

lemma *divide-self-range-const-of-rat*:

$a \in \mathbb{Q}_{\forall p} \implies a \neq 0 \implies$
 $a / a = 1$
 ⟨proof⟩

lemma *range-const-of-rat-add-iff*:

$a \in \mathbb{Q}_{\forall p} \vee$
 $b \in \mathbb{Q}_{\forall p} \implies$
 $a + b \in \mathbb{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathbb{Q}_{\forall p} \wedge b \in \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *range-const-of-rat-diff-iff*:

$a \in \mathbb{Q}_{\forall p} \vee$
 $b \in \mathbb{Q}_{\forall p} \implies$
 $a - b \in \mathbb{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathbb{Q}_{\forall p} \wedge b \in \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *range-const-of-rat-mult-iff*:

$a * b \in \mathbb{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathbb{Q}_{\forall p} \wedge b \in \mathbb{Q}_{\forall p}$
if
 $a \in \mathbb{Q}_{\forall p} - \{0\} \vee$
 $b \in \mathbb{Q}_{\forall p} - \{0\}$
 ⟨proof⟩

lemma *range-const-of-rat-inverse-iff* [simp]:

$\text{inverse } a \in \mathbb{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *range-const-of-rat-divide-iff*:

$a / b \in \mathbb{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathbb{Q}_{\forall p} \wedge b \in \mathbb{Q}_{\forall p}$
if
 $a \in \mathbb{Q}_{\forall p} - \{0\} \vee$
 $b \in \mathbb{Q}_{\forall p} - \{0\}$
 ⟨proof⟩

end

lemma *Fract-eq0-iff*: *Fraction-Field.Fract* $a b = 0 \longleftrightarrow a = 0 \vee b = 0$
 ⟨proof⟩

locale *global-p-equal-div-inv-w-inj-idom-consts* =
global-p-equal-div-inv-w-inj-consts *p-equal* *p-restrict* *func-of-consts*
 + *global-p-equality-w-inj-consts* *p-equal* *p-restrict* *func-of-consts*
for *p-equal* ::
 $'a \Rightarrow 'b :: \{comm-ring-1, inverse, divide-trivial\} \Rightarrow$
 $'b \Rightarrow bool$
and *p-restrict* :: $'b \Rightarrow ('a \Rightarrow bool) \Rightarrow 'b$
and *func-of-consts* :: $('a \Rightarrow 'c :: idom) \Rightarrow 'b$
begin

lift-definition *const-of-fract* :: $'c \text{ fract} \Rightarrow 'b$
is $\lambda x :: 'c \times 'c. \text{const} (\text{fst } x) / \text{const} (\text{snd } x)$
 ⟨proof⟩

lemma *inj-const-of-fract*: *inj* *const-of-fract*
 ⟨proof⟩

lemma *const-of-fract-0* [*simp*]: *const-of-fract* $0 = 0$
 ⟨proof⟩

lemma *const-of-fract-1* [*simp*]: *const-of-fract* $1 = 1$
 ⟨proof⟩

lemma *const-of-fract-Fract*: *const-of-fract* (*Fraction-Field.Fract* $a b$) = *const* $a /$
const b
 ⟨proof⟩

lemma *const-of-fract-p-eq-0-iff* [*simp*]: *p-equal* p (*const-of-fract* x) $0 \longleftrightarrow x = 0$
 ⟨proof⟩

lemma *const-of-fract-equal-0-iff* [*simp*]: *const-of-fract* $x = 0 \longleftrightarrow x = 0$
 ⟨proof⟩

lemma *const-of-fract-minus*: *const-of-fract* $(- a) = - \text{const-of-fract } a$
 ⟨proof⟩

lemma *const-of-fract-add*: *const-of-fract* $(a + b) = \text{const-of-fract } a + \text{const-of-fract } b$

b
 $\langle proof \rangle$

lemma *const-of-fract-mult*: $const-of-fract (a * b) = const-of-fract a * const-of-fract b$
 $\langle proof \rangle$

lemma *const-of-fract-neg-one* [*simp*]: $const-of-fract (- 1) = -1$
 $\langle proof \rangle$

lemma *const-of-fract-diff*: $const-of-fract (a - b) = const-of-fract a - const-of-fract b$
 $\langle proof \rangle$

lemma *const-of-fract-sum*:
 $const-of-fract (\sum a \in A. f a) = (\sum a \in A. const-of-fract (f a))$
 $\langle proof \rangle$

lemma *const-of-fract-prod*:
 $const-of-fract (\prod a \in A. f a) = (\prod a \in A. const-of-fract (f a))$
 $\langle proof \rangle$

lemma *const-of-fract-inverse*: $const-of-fract (inverse a) = inverse (const-of-fract a)$
 $\langle proof \rangle$

lemma *left-inverse-const-of-fract*[*simp*]:
 $inverse (const-of-fract a) * const-of-fract a = 1$ **if** $a \neq 0$
 $\langle proof \rangle$

lemma *right-inverse-const-of-fract*[*simp*]:
 $(const-of-fract a) * inverse (const-of-fract a) = 1$ **if** $a \neq 0$
 $\langle proof \rangle$

lemma *const-of-fract-divide*: $const-of-fract (a / b) = const-of-fract a / const-of-fract b$
 $\langle proof \rangle$

lemma *p-adic-prod-divide-self-of-fract*[*simp*]:
 $(const-of-fract a) / (const-of-fract a) = 1$ **if** $a \neq 0$
 $\langle proof \rangle$

lemma *const-of-fract-power*: $const-of-fract (a ^ n) = (const-of-fract a) ^ n$
 $\langle proof \rangle$

lemma *const-of-fract-eq-iff* [*simp*]:
 $const-of-fract a = const-of-fract b \iff a = b$
 $\langle proof \rangle$

lemma *zero-eq-const-of-fract-iff* [simp]: $0 = \text{const-of-fract } a \longleftrightarrow 0 = a$
 ⟨proof⟩

lemma *const-of-fract-eq-1-iff* [simp]: $\text{const-of-fract } a = 1 \longleftrightarrow a = 1$
 ⟨proof⟩

lemma *one-eq-const-of-fract-iff* [simp]: $1 = \text{const-of-fract } a \longleftrightarrow 1 = a$
 ⟨proof⟩

Collapse nested embeddings.

lemma *const-of-fract-numer-eq-const* [simp]: $\text{const-of-fract } (\text{Fraction-Field.Fract } a \ 1) = \text{const } a$
 ⟨proof⟩

lemma *const-of-fract-of-nat-eq* [simp]: $\text{const-of-fract } (\text{of-nat } n) = \text{of-nat } n$
 ⟨proof⟩

lemma *const-of-fract-of-int-eq* [simp]: $\text{const-of-fract } (\text{of-int } z) = \text{of-int } z$
 ⟨proof⟩

Product of all p-adic embeddings of the field of fractions of the base ring.

definition *range-const-of-fract* :: 'b set ($\mathbb{Q}_{\forall p}$)
 where $\mathbb{Q}_{\forall p} = \text{range } \text{const-of-fract}$

lemma *range-const-of-fract-cases* [cases set: *range-const-of-fract*]:
 assumes $q \in \mathbb{Q}_{\forall p}$
 obtains $(\text{const-of-fract}) \ r$ where $q = \text{const-of-fract } r$
 ⟨proof⟩

lemma *range-const-of-fract-cases'*:
 assumes $x \in \mathbb{Q}_{\forall p}$
 obtains $a \ b$ where $b \neq 0 \ x = \text{const } a / \text{const } b$
 ⟨proof⟩

lemma *range-const-of-fract-of-fract* [simp]:
 $\text{const-of-fract } r \in \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *range-const-of-fract-const* [simp]: $\text{const } a \in \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *range-const-of-fract-of-int* [simp]: $\text{of-int } z \in \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *Ints-subset-range-const-of-fract*: $\mathbb{Z} \subseteq \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *range-const-of-fract-of-nat* [simp]: $\text{of-nat } n \in \mathbb{Q}_{\forall p}$
 ⟨proof⟩

lemma *Nats-subset-range-const-of-fract*: $\mathbf{N} \subseteq \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-0* [*simp*]: $0 \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-1* [*simp*]: $1 \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-add* [*simp*]:

$a \in \mathcal{Q}_{\forall p} \implies$
 $b \in \mathcal{Q}_{\forall p} \implies$
 $a + b \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-minus-iff* [*simp*]:

$- a \in \mathcal{Q}_{\forall p} \iff$
 $a \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-diff* [*simp*]:

$a \in \mathcal{Q}_{\forall p} \implies$
 $b \in \mathcal{Q}_{\forall p} \implies$
 $a - b \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-mult* [*simp*]:

$a \in \mathcal{Q}_{\forall p} \implies$
 $b \in \mathcal{Q}_{\forall p} \implies$
 $a * b \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-inverse* [*simp*]:

$a \in \mathcal{Q}_{\forall p} \implies$
inverse $a \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-divide* [*simp*]:

$a \in \mathcal{Q}_{\forall p} \implies$
 $b \in \mathcal{Q}_{\forall p} \implies$
 $a / b \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-power* [*simp*]:

$a \in \mathcal{Q}_{\forall p} \implies$
 $a ^ n \in \mathcal{Q}_{\forall p}$
<proof>

lemma *range-const-of-fract-sum* [intro]:

$(\bigwedge x. x \in A \implies$
 $f x \in \mathcal{Q}_{\forall p}) \implies$
 $\text{sum } f A \in \mathcal{Q}_{\forall p}$
⟨proof⟩

lemma *range-const-of-fract* [intro]:

$(\bigwedge x. x \in A \implies f x \in \mathcal{Q}_{\forall p})$
 $\implies \text{prod } f A \in \mathcal{Q}_{\forall p}$
⟨proof⟩

lemma *range-const-of-fract-induct* [case-names const-of-fract, induct set: range-const-of-fract]:

$q \in \mathcal{Q}_{\forall p} \implies$
 $(\bigwedge r. P (\text{const-of-fract } r)) \implies P q$
⟨proof⟩

lemma *p-adic-Fracts-p-equal-0-iff*:

$a \in \mathcal{Q}_{\forall p} \implies$
 $p\text{-equal } p a 0 \iff a = 0$
⟨proof⟩

lemma *range-const-of-fract-infinite*:

infinite $\mathcal{Q}_{\forall p}$ **if** *infinite* (UNIV :: 'c set)
⟨proof⟩

lemma *left-inverse-range-const-of-fract*:

$a \in \mathcal{Q}_{\forall p} \implies a \neq 0 \implies$
 $\text{inverse } a * a = 1$
⟨proof⟩

lemma *right-inverse-range-const-of-fract*:

$a \in \mathcal{Q}_{\forall p} \implies a \neq 0 \implies$
 $a * \text{inverse } a = 1$
⟨proof⟩

lemma *divide-self-range-const-of-fract*:

$a \in \mathcal{Q}_{\forall p} \implies a \neq 0 \implies$
 $a / a = 1$
⟨proof⟩

lemma *range-const-of-fract-add-iff*:

$a \in \mathcal{Q}_{\forall p} \vee$
 $b \in \mathcal{Q}_{\forall p} \implies$
 $a + b \in \mathcal{Q}_{\forall p} \iff$
 $a \in \mathcal{Q}_{\forall p} \wedge b \in \mathcal{Q}_{\forall p}$
⟨proof⟩

lemma *range-const-of-fract-diff-iff*:

$a \in \mathcal{Q}_{\forall p} \vee$

$b \in \mathcal{Q}_{\forall p} \implies$
 $a - b \in \mathcal{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathcal{Q}_{\forall p} \wedge b \in \mathcal{Q}_{\forall p}$
 ⟨proof⟩

lemma *range-const-of-fract-mult-iff*:

$a * b \in \mathcal{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathcal{Q}_{\forall p} \wedge b \in \mathcal{Q}_{\forall p}$
if
 $a \in \mathcal{Q}_{\forall p} - \{0\} \vee$
 $b \in \mathcal{Q}_{\forall p} - \{0\}$
 ⟨proof⟩

lemma *range-const-of-fract-inverse-iff* [simp]:

$\text{inverse } a \in \mathcal{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathcal{Q}_{\forall p}$
 ⟨proof⟩

lemma *range-const-of-fract-divide-iff*:

$a / b \in \mathcal{Q}_{\forall p} \longleftrightarrow$
 $a \in \mathcal{Q}_{\forall p} \wedge b \in \mathcal{Q}_{\forall p}$
if
 $a \in \mathcal{Q}_{\forall p} - \{0\} \vee$
 $b \in \mathcal{Q}_{\forall p} - \{0\}$
 ⟨proof⟩

end

locale *global-p-equality-w-inj-index-consts* = *global-p-equality-w-inj-consts*

+
fixes *index-inj* :: 'a ⇒ 'c
assumes *index-inj* : *inj index-inj*
and *index-inj-nzero*: *index-inj p ≠ 0*
begin

abbreviation *global-uniformizer* ≡ *consts index-inj*

abbreviation *index-const p* ≡ *const (index-inj p)*

lemma *global-uniformizer-locally-uniformizes*: *p-equal p global-uniformizer (index-const p)*

⟨proof⟩

lemma *index-const-globally-nequiv-0*: \neg *p-equal q (index-const p) 0*

⟨proof⟩

lemma *global-uniformizer-globally-nequiv-0*: \neg *p-equal p global-uniformizer 0*

⟨proof⟩

end

locale *global-p-equality-no-zero-divisors-1-w-inj-index-consts* =
global-p-equality-w-inj-index-consts + *p-equality-no-zero-divisors-1*
begin

lemma *index-const-pow-globally-nequiv-0*: $\neg p\text{-equal } q \text{ (index-const } p \wedge n) 0$
<proof>

lemma *global-uniformizer-pow-globally-nequiv-0*: $\neg p\text{-equal } q \text{ (global-uniformizer } \wedge n) 0$
<proof>

end

locale *global-p-equality-depth-w-inj-index-consts* =
p-equality-depth p-equal p-depth
+ *global-p-equality-w-inj-index-consts p-equal p-restrict func-of-consts index-inj*
for *p-equal* :: 'a \Rightarrow 'b::*comm-ring-1* \Rightarrow 'b \Rightarrow *bool*
and *p-restrict* :: 'b \Rightarrow ('a \Rightarrow *bool*) \Rightarrow 'b
and *p-depth* :: 'a \Rightarrow 'b \Rightarrow *int*
and *func-of-consts* :: ('a \Rightarrow 'c::{*factorial-semiring,ring-1*}) \Rightarrow 'b
and *index-inj* :: 'a \Rightarrow 'c
+
assumes *p-depth-func-of-consts*:
p-depth p (func-of-consts f) = int (multiplicity (index-inj p) (f p))
and *index-inj-nunit* : $\neg \text{is-unit (index-inj p)}$
and *index-inj-coprime*:
p \neq q \implies multiplicity (index-inj p) (index-inj q) = 0
begin

lemma *p-depth-set-consts*: *func-of-consts f \in p-depth-set p 0*
<proof>

lemma *global-depth-set-consts*: *func-of-consts f \in global-depth-set 0*
<proof>

lemma *p-depth-1[simp]*: *p-depth p (1::'b) = 0*
<proof>

lemma *p-depth-index-const*: *p-depth p (index-const p) = 1*
<proof>

lemma *p-depth-index-const'*: *p-depth q (index-const p) = of-bool (q = p)*
<proof>

lemma *p-depth-global-uniformizer*: *p-depth p (global-uniformizer::'b) = 1*
<proof>

lemma *p-depth-consts-func-ge0*: *p-depth p (consts f) \geq 0*

```

    <proof>

end

locale global-p-equal-depth-no-zero-divisors-w-inj-index-consts =
  global-p-equal p-equal p-restrict
+ global-p-equality-depth-no-zero-divisors-1 p-equal p-restrict p-depth
+ global-p-equality-depth-w-inj-index-consts p-equal p-restrict p-depth func-of-consts
index-inj
  for p-equal      :: 'a ⇒ 'b::comm-ring-1 ⇒ 'b ⇒ bool
  and p-restrict  :: 'b ⇒ ('a ⇒ bool) ⇒ 'b
  and p-depth     :: 'a ⇒ 'b ⇒ int
  and func-of-consts :: ('a ⇒ 'c::{factorial-semiring,ring-1}) ⇒ 'b
  and index-inj   :: 'a ⇒ 'c
begin

sublocale global-p-equal-w-inj-consts <proof>
sublocale global-p-equal-depth-no-zero-divisors <proof>

lemma p-depth-index-const-pow: p-depth p ((index-const p) ^ n) = int n
  <proof>

lemma p-depth-index-const-pow': p-depth q ((index-const p) ^ n) = int n * of-bool
(q = p)
  <proof>

lemma p-depth-global-uniformizer-pow: p-depth p (global-uniformizer ^ n) = int n
  <proof>

end

locale global-p-equal-depth-div-inv-w-inj-index-consts =
  global-p-equal-depth-div-inv p-equal p-depth p-restrict
+ global-p-equal-depth-no-zero-divisors-w-inj-index-consts
  p-equal p-restrict p-depth func-of-consts index-inj
  for p-equal      ::
    'a ⇒ 'b::{comm-ring-1, inverse, divide-trivial} ⇒
    'b ⇒ bool
  and p-depth     :: 'a ⇒ 'b ⇒ int
  and p-restrict  :: 'b ⇒ ('a ⇒ bool) ⇒ 'b
  and func-of-consts :: ('a ⇒ 'c::{factorial-semiring,ring-1}) ⇒ 'b
  and index-inj   :: 'a ⇒ 'c
begin

lemma index-const-powi-globally-nequiv-0: ¬ p-equal q (index-const p powi n) 0
  <proof>

lemma global-uniformizer-powi-globally-nequiv-0: ¬ p-equal q (global-uniformizer

```

$\text{powi } n) 0$
 $\langle \text{proof} \rangle$

lemma *index-const-powi-add*:

$(\text{index-const } p) \text{ powi } (m + n) = (\text{index-const } p) \text{ powi } m * (\text{index-const } p) \text{ powi } n$
 $\langle \text{proof} \rangle$

lemma *global-uniformizer-powi-add*:

$\text{global-uniformizer powi } (m + n) = \text{global-uniformizer powi } m * \text{global-uniformizer powi } n$
 $\langle \text{proof} \rangle$

lemma *p-depth-index-const-powi*: $p\text{-depth } p (\text{index-const } p \text{ powi } n) = n$
 $\langle \text{proof} \rangle$

lemma *p-depth-index-const-powi'*: $p\text{-depth } q (\text{index-const } p \text{ powi } n) = n * \text{of-bool } (q = p)$
 $\langle \text{proof} \rangle$

lemma *p-depth-global-uniformizer-powi*: $p\text{-depth } p (\text{global-uniformizer powi } n) = n$
 $\langle \text{proof} \rangle$

lemma *local-uniformizer-locally-uniformizes*:

$p\text{-depth } p ((\text{index-const } p) \text{ powi } (-n) * x) = 0 \text{ if } p\text{-depth } p x = n$
 $\langle \text{proof} \rangle$

lemma *global-uniformizer-locally-uniformizes*:

$p\text{-depth } p (\text{global-uniformizer powi } (-n) * x) = 0 \text{ if } p\text{-depth } p x = n$
 $\langle \text{proof} \rangle$

lemma *p-depth-set-eq-index-const-coset*:

$p\text{-depth-set } p n = ((\text{index-const } p) \text{ powi } n) * o (p\text{-depth-set } p 0)$
 $\langle \text{proof} \rangle$

lemma *p-depth-set-eq-index-const-coset'*:

$p\text{-depth-set } p 0 = ((\text{index-const } p) \text{ powi } (-n)) * o (p\text{-depth-set } p n)$
 $\langle \text{proof} \rangle$

lemma *p-depth-set-eq-global-uniformizer-coset*:

$p\text{-depth-set } p n = (\text{global-uniformizer powi } n) * o (p\text{-depth-set } p 0)$
 $\langle \text{proof} \rangle$

lemma *p-depth-set-eq-global-uniformizer-coset'*:

$p\text{-depth-set } p 0 = (\text{global-uniformizer powi } (-n)) * o (p\text{-depth-set } p n)$
 $\langle \text{proof} \rangle$

lemma *global-depth-set-eq-global-uniformizer-coset*:

$\text{global-depth-set } n = (\text{global-uniformizer powi } n) * o (\text{global-depth-set } 0)$

<proof>

lemma *global-depth-set-eq-global-uniformizer-coset'*:

*global-depth-set 0 = (global-uniformizer powi (-n)) *o (global-depth-set n)*

<proof>

definition *shift-p-depth* :: 'a \Rightarrow int \Rightarrow 'b \Rightarrow 'b

where *shift-p-depth p n x = x * (consts (λq . if q = p then index-inj p else 1))*
powi n

lemma *shift-p-depth-0[simp]*: *shift-p-depth p n 0 = 0*

<proof>

lemma *shift-p-depth-by-0[simp]*: *shift-p-depth p 0 x = x*

<proof>

lemma *shift-p-depth-add[simp]*:

shift-p-depth p n (x + y) = shift-p-depth p n x + shift-p-depth p n y

<proof>

lemma *shift-p-depth-uminus[simp]*:

shift-p-depth p n (- x) = - shift-p-depth p n x

<proof>

lemma *shift-p-depth-minus[simp]*:

shift-p-depth p n (x - y) = shift-p-depth p n x - shift-p-depth p n y

<proof>

lemma *shift-p-depth-equiv-at-place*: *p-equal p (shift-p-depth p n x) (x * (index-const p powi n))*

<proof>

lemma *shift-p-depth-equiv-at-other-places*:

p-equal q (shift-p-depth p n x) x **if** $q \neq p$

<proof>

lemma *shift-p-depth-equiv0-iff*:

p-equal q (shift-p-depth p n x) 0 \longleftrightarrow p-equal q x 0

<proof>

lemma *shift-p-depth-equiv-iff*:

p-equal q (shift-p-depth p n x) (shift-p-depth p n y) = p-equal q x y

<proof>

lemma *shift-p-depth-trivial-iff*:

shift-p-depth p n x = x \longleftrightarrow n = 0 \vee p-equal p x 0

<proof>

lemma *shift-p-depth-times-right*: *shift-p-depth p n (x * y) = x * shift-p-depth p n*

y
(proof)

lemma *shift-p-depth-times-left*: $\text{shift-p-depth } p \ n \ (x * y) = \text{shift-p-depth } p \ n \ x * y$
(proof)

lemma *shift-shift-p-depth*: $\text{shift-p-depth } p \ n \ (\text{shift-p-depth } p \ m \ x) = \text{shift-p-depth } p \ (m + n) \ x$
(proof)

lemma *shift-shift-p-depth-image*:
 $\text{shift-p-depth } p \ n \ ` \text{shift-p-depth } p \ m \ ` \ A = \text{shift-p-depth } p \ (m + n) \ ` \ A$
(proof)

lemma *shift-p-depth-at-place*:
 $p\text{-depth } p \ (\text{shift-p-depth } p \ n \ x) = p\text{-depth } p \ x + n$ **if** $n = 0 \vee \neg p\text{-equal } p \ x \ 0$
(proof)

lemma *shift-p-depth-at-other-places*:
 $p\text{-depth } q \ (\text{shift-p-depth } p \ n \ x) = p\text{-depth } q \ x$ **if** $q \neq p$
(proof)

lemma *p-depth-set-shift-p-depth-closed*:
 $\text{shift-p-depth } p \ n \ x \in p\text{-depth-set } p \ m$ **if** $p\text{-depth } p \ x \geq m - n$
(proof)

lemma *global-depth-set-shift-p-depth-closed*:
 $\text{shift-p-depth } p \ n \ x \in \text{global-depth-set } m$
if $x \in \text{global-depth-set } m$
and $\neg p\text{-equal } p \ x \ 0 \longrightarrow p\text{-depth } p \ x \geq m - n$
(proof)

lemma *shift-p-depth-mem-p-depth-set*:
 $p\text{-depth } p \ x \geq m - n$
if $\neg p\text{-equal } p \ x \ 0$ **and** $\text{shift-p-depth } p \ n \ x \in p\text{-depth-set } p \ m$
(proof)

lemma *shift-p-depth-mem-global-depth-set*:
 $p\text{-depth } p \ x \geq m - n$
if $\neg p\text{-equal } p \ x \ 0$ **and** $\text{shift-p-depth } p \ n \ x \in \text{global-depth-set } m$
(proof)

lemma *shift-p-depth-p-depth-set*: $\text{shift-p-depth } p \ n \ ` \ p\text{-depth-set } p \ m = p\text{-depth-set } p \ (m + n)$
(proof)

lemma *shift-p-depth-cong*:
 $p\text{-equal } q \ (\text{shift-p-depth } p \ n \ x) \ (\text{shift-p-depth } p \ n \ y)$ **if** $p\text{-equal } q \ x \ y$
(proof)

lemma *shift-p-depth-p-restrict*:
 $\text{shift-p-depth } p \ n \ (p\text{-restrict } x \ P) = p\text{-restrict } (\text{shift-p-depth } p \ n \ x) \ P$
 ⟨proof⟩

lemma *shift-p-depth-p-restrict-global-depth-set-memI*:
 $\text{shift-p-depth } p \ n \ (p\text{-restrict } x \ ((=) \ p)) \in \text{global-depth-set } m$
if $\neg p\text{-equal } p \ x \ 0 \longrightarrow p\text{-depth } p \ x \geq m - n$
 ⟨proof⟩

lemma *shift-p-depth-p-restrict-global-depth-set-memI'*:
 $\text{shift-p-depth } p \ n \ (p\text{-restrict } x \ ((=) \ p)) \in \text{global-depth-set } (m + n)$
if $x \in p\text{-depth-set } p \ m$
 ⟨proof⟩

lemma *shift-p-depth-p-restrict-global-depth-set-image*:
 $\text{shift-p-depth } p \ n \ (\lambda x. p\text{-restrict } x \ ((=) \ p)) \text{ 'global-depth-set } m$
 $= (\lambda x. p\text{-restrict } x \ ((=) \ p)) \text{ 'global-depth-set } (m + n)$
 ⟨proof⟩

end

3.7 Topological patterns

3.7.1 By place

Convergence of sequences as measured by depth

context *p-equality-depth*
begin

definition *p-limseq-condition* ::
 $'a \Rightarrow (\text{nat} \Rightarrow 'b) \Rightarrow 'b \Rightarrow \text{int} \Rightarrow$
 $\text{nat} \Rightarrow \text{bool}$
where
 $p\text{-limseq-condition } p \ X \ x \ n \ K =$
 $(\forall k \geq K. \neg p\text{-equal } p \ (X \ k) \ x \longrightarrow p\text{-depth } p \ (X \ k - x) > n)$

lemma *p-limseq-conditionI* [intro]:
 $p\text{-limseq-condition } p \ X \ x \ n \ K$
if
 $\bigwedge k. k \geq K \implies \neg p\text{-equal } p \ (X \ k) \ x \implies$
 $p\text{-depth } p \ (X \ k - x) > n$
 ⟨proof⟩

lemma *p-limseq-conditionD*:
 $p\text{-depth } p \ (X \ k - x) > n$
if $p\text{-limseq-condition } p \ X \ x \ n \ K$ **and** $k \geq K$ **and** $\neg p\text{-equal } p \ (X \ k) \ x$
 ⟨proof⟩

abbreviation $p\text{-limseq}$::

$'a \Rightarrow (\text{nat} \Rightarrow 'b) \Rightarrow 'b \Rightarrow \text{bool}$

where $p\text{-limseq } p \ X \ x \equiv (\forall n. \exists K. p\text{-limseq-condition } p \ X \ x \ n \ K)$

lemma $p\text{-limseq-p-cong}$:

$p\text{-limseq } p \ X \ x$

if $\forall n \geq N. p\text{-equal } p \ (X \ n) \ (Y \ n)$ **and** $p\text{-equal } p \ x \ y$ **and** $p\text{-limseq } p \ Y \ y$
 $\langle \text{proof} \rangle$

lemma $p\text{-limseq-p-constant}$: $p\text{-limseq } p \ X \ x$ **if** $\forall n. p\text{-equal } p \ (X \ n) \ x$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-constant}$: $p\text{-limseq } p \ (\lambda n. x) \ x$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-p-eventually-constant}$:

$p\text{-limseq } p \ X \ x$ **if** $\forall_F k$ *in sequentially*. $p\text{-equal } p \ (X \ k) \ x$
 $\langle \text{proof} \rangle$

lemma $p\text{-limseq-0 [simp]}$: $p\text{-limseq } p \ 0 \ 0$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-0-iff}$: $p\text{-limseq } p \ 0 \ x \longleftrightarrow p\text{-equal } p \ x \ 0$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-add}$: $p\text{-limseq } p \ (X + Y) \ (x + y)$ **if** $p\text{-limseq } p \ X \ x$ **and** $p\text{-limseq } p \ Y \ y$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-uminus}$: $p\text{-limseq } p \ (-X) \ (-x)$ **if** $p\text{-limseq } p \ X \ x$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-diff}$: $p\text{-limseq } p \ (X - Y) \ (x - y)$ **if** $p\text{-limseq } p \ X \ x$ **and** $p\text{-limseq } p \ Y \ y$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-conv-0}$: $p\text{-limseq } p \ X \ x \longleftrightarrow p\text{-limseq } p \ (\lambda n. X \ n - x) \ 0$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-unique}$: $p\text{-equal } p \ x \ y$ **if** $p\text{-limseq } p \ X \ x$ **and** $p\text{-limseq } p \ X \ y$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-eventually-nequiv-0}$:

$\forall_F k$ *in sequentially*. $\neg p\text{-equal } p \ (X \ k) \ 0$

if $\neg p\text{-equal } p \ x \ 0$ **and** $p\text{-limseq } p \ X \ x$
 $\langle \text{proof} \rangle$

lemma $p\text{-limseq-bdd-depth}$:

assumes $\neg p\text{-equal } p \ x \ 0$ **and** $p\text{-limseq } p \ X \ x$

obtains d **where** $\forall k. \neg p\text{-equal } p (X k) 0 \longrightarrow p\text{-depth } p (X k) \leq d$
 ⟨proof⟩

lemma *p-limseq-eventually-bdd-depth*:
 $\forall_F k$ *in sequentially.* $p\text{-depth } p (X k) \leq p\text{-depth } p x$
if $\neg p\text{-equal } p x 0$ **and** $p\text{-limseq } p X x$
 ⟨proof⟩

lemma *p-limseq-delayed-iff*:
 $p\text{-limseq } p X x \longleftrightarrow p\text{-limseq } p (\lambda n. X (N + n)) x$
 ⟨proof⟩

lemma *p-depth-set-p-limseq-closed*:
 $x \in p\text{-depth-set } p n$
if $p\text{-limseq } p X x$ **and** $\forall_F k$ *in sequentially.* $X k \in p\text{-depth-set } p n$
 ⟨proof⟩

Cauchy sequences by place

definition *p-cauchy-condition* ::
 $'a \Rightarrow (\text{nat} \Rightarrow 'b) \Rightarrow \text{int} \Rightarrow \text{nat} \Rightarrow \text{bool}$
where
 $p\text{-cauchy-condition } p X n K =$
 $(\forall k1 \geq K. \forall k2 \geq K.$
 $\neg p\text{-equal } p (X k1) (X k2) \longrightarrow p\text{-depth } p (X k1 - X k2) > n$
 $)$

lemma *p-cauchy-conditionI*:
 $p\text{-cauchy-condition } p X n K$
if
 $\bigwedge k k'. k \geq K \implies k' \geq K \implies$
 $\neg p\text{-equal } p (X k) (X k') \implies p\text{-depth } p (X k - X k') > n$
 ⟨proof⟩

lemma *p-cauchy-conditionD*:
 $p\text{-depth } p (X k - X k') > n$
if $p\text{-cauchy-condition } p X n K$ **and** $k \geq K$ **and** $k' \geq K$
and $\neg p\text{-equal } p (X k) (X k')$
 ⟨proof⟩

lemma *p-cauchy-condition-mono-seq-tail*:
 $p\text{-cauchy-condition } p X n K \implies p\text{-cauchy-condition } p X n K'$
if $K' \geq K$
 ⟨proof⟩

lemma *p-cauchy-condition-mono-depth*:
 $p\text{-cauchy-condition } p X n K \implies p\text{-cauchy-condition } p X m K$
if $m \leq n$
 ⟨proof⟩

abbreviation $p\text{-cauchy } p X \equiv (\forall n. \exists K. p\text{-cauchy-condition } p X n K)$

lemma $p\text{-cauchy-condition-LEAST}$:

$p\text{-cauchy-condition } p X n \text{ (LEAST } K. p\text{-cauchy-condition } p X n K)$

if $p\text{-cauchy } p X$

$\langle \text{proof} \rangle$

lemma $p\text{-cauchy-condition-LEAST-mono}$:

$(\text{LEAST } K. p\text{-cauchy-condition } p X m K) \leq (\text{LEAST } K. p\text{-cauchy-condition } p X n K)$

if $p\text{-cauchy } p X$ **and** $m \leq n$

$\langle \text{proof} \rangle$

definition $p\text{-consec-cauchy-condition} ::$

$'a \Rightarrow (\text{nat} \Rightarrow 'b) \Rightarrow \text{int} \Rightarrow \text{nat} \Rightarrow \text{bool}$

where

$p\text{-consec-cauchy-condition } p X n K =$

$(\forall k \geq K.$

$\neg p\text{-equal } p (X (\text{Suc } k)) (X k) \longrightarrow p\text{-depth } p (X (\text{Suc } k) - X k) > n$

$)$

lemma $p\text{-consec-cauchy-conditionI}$:

$p\text{-consec-cauchy-condition } p X n K$

if

$\bigwedge k. k \geq K \implies \neg p\text{-equal } p (X (\text{Suc } k)) (X k) \implies$

$p\text{-depth } p (X (\text{Suc } k) - X k) > n$

$\langle \text{proof} \rangle$

lemma $p\text{-consec-cauchy}$:

$p\text{-cauchy } p X = (\forall n. \exists K. p\text{-consec-cauchy-condition } p X n K)$

$\langle \text{proof} \rangle$

end

context $p\text{-equality-depth-no-zero-divisors}$

begin

lemma $p\text{-limseq-mult-both-0}$: $p\text{-limseq } p (X * Y) 0$ **if** $p\text{-limseq } p X 0$ **and** $p\text{-limseq } p Y 0$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-constant-mult-0}$: $p\text{-limseq } p (\lambda n. x * Y n) 0$ **if** $p\text{-limseq } p Y 0$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-mult-0-right}$: $p\text{-limseq } p (X * Y) 0$ **if** $p\text{-limseq } p X x$ **and** $p\text{-limseq } p Y 0$

$\langle \text{proof} \rangle$

lemma $p\text{-limseq-mult}$: $p\text{-limseq } p (X * Y) (x * y)$ **if** $p\text{-limseq } p X x$ **and** $p\text{-limseq } p Y y$

$p \ Y \ y$
 $\langle \text{proof} \rangle$

lemma *poly-continuous-p-open-nbhds*:
 $p\text{-limseq } p \ (\lambda n. \text{poly } f \ (X \ n)) \ (\text{poly } f \ x) \ \mathbf{if} \ p\text{-limseq } p \ X \ x$
 $\langle \text{proof} \rangle$

end

context *p-equality-depth-no-zero-divisors-1*
begin

lemma *p-limseq-pow*: $p\text{-limseq } p \ (\lambda k. (X \ k)^\wedge n) \ (x^\wedge n) \ \mathbf{if} \ p\text{-limseq } p \ X \ x$
 $\langle \text{proof} \rangle$

lemma *p-limseq-poly*: $p\text{-limseq } p \ (\lambda n. \text{poly } f \ (X \ n)) \ (\text{poly } f \ x) \ \mathbf{if} \ p\text{-limseq } p \ X \ x$
 $\langle \text{proof} \rangle$

end

context *p-equality-depth-div-inv*
begin

lemma *p-limseq-inverse*:
 $p\text{-limseq } p \ (\lambda n. \text{inverse } (X \ n)) \ (\text{inverse } x)$
 $\mathbf{if} \ \neg \ p\text{-equal } p \ x \ 0 \ \mathbf{and} \ p\text{-limseq } p \ X \ x$
 $\langle \text{proof} \rangle$

lemma *p-limseq-divide*:
 $p\text{-limseq } p \ (\lambda n. X \ n \ / \ Y \ n) \ (x \ / \ y)$
 $\mathbf{if} \ \neg \ p\text{-equal } p \ y \ 0 \ \mathbf{and} \ p\text{-limseq } p \ X \ x \ \mathbf{and} \ p\text{-limseq } p \ Y \ y$
 $\langle \text{proof} \rangle$

end

context *global-p-equality-depth*
begin

lemma *p-limseq-p-restrict-iff*:
 $p\text{-limseq } p \ (\lambda n. \text{p-restrict } (X \ n) \ P) \ x \ \longleftrightarrow \ p\text{-limseq } p \ X \ x \ \mathbf{if} \ P \ p$
 $\langle \text{proof} \rangle$

lemma *p-limseq-p-restricted*: $p\text{-limseq } p \ (\lambda n. \text{p-restrict } (X \ n) \ P) \ 0 \ \mathbf{if} \ \neg \ P \ p$
 $\langle \text{proof} \rangle$

end

Base of open sets at local places

context *global-p-equal-depth*

begin

abbreviation $p\text{-open-nbhd} :: 'a \Rightarrow \text{int} \Rightarrow 'b \Rightarrow 'b \text{ set}$
where $p\text{-open-nbhd } p \ n \ x \equiv x + o \ p\text{-depth-set } p \ n$

abbreviation $local\text{-}p\text{-open-nbhds } p \equiv \{ p\text{-open-nbhd } p \ n \ x \mid n \ x. \ \text{True} \}$
abbreviation $p\text{-open-nbhds} \equiv (\bigcup p. \ local\text{-}p\text{-open-nbhds } p)$

lemma $p\text{-open-nbhd}$: $x \in p\text{-open-nbhd } p \ n \ x$
<proof>

lemma $local\text{-}p\text{-open-nbhd-is-open}$: $generate\text{-topology } (local\text{-}p\text{-open-nbhds } p) \ (p\text{-open-nbhd } p \ n \ x)$
<proof>

lemma $p\text{-open-nbhd-is-open}$: $generate\text{-topology } p\text{-open-nbhds} \ (p\text{-open-nbhd } p \ n \ x)$
<proof>

lemma $local\text{-}p\text{-open-nbhds-are-coarser}$:
 $generate\text{-topology } (local\text{-}p\text{-open-nbhds } p) \ S \implies generate\text{-topology } p\text{-open-nbhds } S$
<proof>

lemma $p\text{-open-nbhd-diff-depth}$:
 $p\text{-depth } p \ (y - x) \geq n \ \text{if } \neg p\text{-equal } p \ y \ x \ \text{and } y \in p\text{-open-nbhd } p \ n \ x$
<proof>

lemma $p\text{-open-nbhd-eq-circle}$:
 $p\text{-open-nbhd } p \ n \ x = \{y. \ \neg p\text{-equal } p \ y \ x \longrightarrow p\text{-depth } p \ (y - x) \geq n\}$
<proof>

lemma $p\text{-open-nbhd-circle-multicentre}$:
 $p\text{-open-nbhd } p \ n \ y = p\text{-open-nbhd } p \ n \ x \ \text{if } y \in p\text{-open-nbhd } p \ n \ x$
<proof>

lemma $p\text{-open-nbhd-circle-multicentre}'$:
 $x \in p\text{-open-nbhd } p \ n \ y \longleftrightarrow y \in p\text{-open-nbhd } p \ n \ x$
<proof>

lemma $p\text{-open-nbhd-p-restrict}$:
 $p\text{-open-nbhd } p \ n \ (p\text{-restrict } x \ P) = p\text{-open-nbhd } p \ n \ x \ \text{if } P \ p$
<proof>

lemma $p\text{-restrict-p-open-nbhd-mem-iff}$:
 $y \in p\text{-open-nbhd } p \ n \ x \longleftrightarrow p\text{-restrict } y \ P \in p\text{-open-nbhd } p \ n \ x \ \text{if } P \ p$
<proof>

lemma $p\text{-open-nbhd-p-restrict-add-mixed-drop-right}$:
 $p\text{-open-nbhd } p \ n \ (p\text{-restrict } x \ P + p\text{-restrict } y \ Q) = p\text{-open-nbhd } p \ n \ x$
if $P \ p$ **and** $\neg Q \ p$

<proof>

lemma *p-open-nbhd-p-restrict-add-mixed-drop-left:*

p-open-nbhd p n (p-restrict x P + p-restrict y Q) = p-open-nbhd p n y

if $\neg P p$ **and** $Q p$

<proof>

lemma *p-open-nbhd-antimono:*

p-open-nbhd p n x \subseteq p-open-nbhd p m x **if** $m \leq n$

<proof>

lemma *p-open-nbhds-open-subopen:*

generate-topology (local-p-open-nbhds p) A =

($\forall a \in A. \exists n. p\text{-open-nbhd } p n a \subseteq A$)

<proof>

lemma *p-restrict-p-open-set-mem-iff:*

a \in A \longleftrightarrow p-restrict a P \in A

if *generate-topology (local-p-open-nbhds p) A* **and** $P p$

<proof>

lemma *hausdorff:*

$\exists p n. (p\text{-open-nbhd } p n x) \cap (p\text{-open-nbhd } p n y) = \{\}$ **if** $x \neq y$

<proof>

sublocale *t2-p-open-nbhds: t2-space generate-topology p-open-nbhds*

<proof>

abbreviation *p-open-nbhds-tendsto* \equiv *t2-p-open-nbhds.tendsto*

abbreviation *p-open-nbhds-convergent* \equiv *t2-p-open-nbhds.convergent*

abbreviation *p-open-nbhds-LIMSEQ* \equiv *t2-p-open-nbhds.LIMSEQ*

lemma *locally-limD: $\exists K. p\text{-limseq-condition } p X x n K$* **if** *p-open-nbhds-LIMSEQ*

X x

<proof>

lemma *globally-limseq-imp-locally-limseq:*

p-limseq p X x **if** *p-open-nbhds-LIMSEQ X x*

<proof>

lemma *globally-limseq-iff-locally-limseq: p-open-nbhds-LIMSEQ X x = ($\forall p. p\text{-limseq}$*

p X x)

<proof>

lemma *p-open-nbhds-LIMSEQ-eventually-p-constant:*

p-equal p x c

if *p-open-nbhds-LIMSEQ X x* **and** $\forall_F k$ *in sequentially. p-equal p (X k) c*

<proof>

lemma *p-open-nbhds-LIMSEQ-p-restrict*:
p-equal p x 0 **if** *p-open-nbhds-LIMSEQ* ($\lambda n. p\text{-restrict } (X\ n)\ P$) *x* **and** $\neg P\ p$
 ⟨*proof*⟩

lemma *global-depth-set-p-open-nbhds-LIMSEQ-closed*:
x \in *global-depth-set n*
if *p-open-nbhds-LIMSEQ* *X x*
and $\forall_F k$ *in sequentially*. *X k* \in *global-depth-set n*
 ⟨*proof*⟩

end

context *global-p-equal-depth-div-inv-w-inj-index-consts*
begin

lemma *shift-p-depth-p-open-nbhd*:
shift-p-depth p n ‘ *p-open-nbhd p m x* = *p-open-nbhd p (m + n)* (*shift-p-depth p n x*)
 ⟨*proof*⟩

lemma *shift-p-depth-p-open-set*:
generate-topology (local-p-open-nbhds p) (*shift-p-depth p n* ‘ *A*)
if *generate-topology (local-p-open-nbhds p)* *A*
 ⟨*proof*⟩

end

locale *p-complete-global-p-equal-depth* = *global-p-equal-depth* +
assumes *complete*:
p-cauchy p X \implies *p-open-nbhds-convergent* ($\lambda n. p\text{-restrict } (X\ n)\ ((=)\ p)$)
begin

lemma *p-cauchyE*:
assumes *p-cauchy p X*
obtains *x* **where** *p-open-nbhds-LIMSEQ* ($\lambda n. p\text{-restrict } (X\ n)\ ((=)\ p)$) *x*
 ⟨*proof*⟩

end

Hensel’s Lemma.

context *p-equality-div-inv*
begin

— A sequence to approach a root from within the ring of integers.

primrec *hensel-seq* :: ‘*a* \Rightarrow ‘*b* *poly* \Rightarrow ‘*b* \Rightarrow *nat* \Rightarrow ‘*b*
where *hensel-seq p f x 0* = *x* |
hensel-seq p f x (Suc n) = (
if p-equal p (poly f (hensel-seq p f x n)) 0 *then hensel-seq p f x n*
else

$\text{hensel-seq } p \text{ f } x \text{ n} - (\text{poly } f (\text{hensel-seq } p \text{ f } x \text{ n})) / (\text{poly } (\text{polyderiv } f) (\text{hensel-seq } p \text{ f } x \text{ n}))$

lemma *constant-hensel-seq-case*:

$k \geq n \implies \text{hensel-seq } p \text{ f } x \text{ k} = \text{hensel-seq } p \text{ f } x \text{ n}$
if $p\text{-equal } p (\text{poly } f (\text{hensel-seq } p \text{ f } x \text{ n})) \text{ } 0$
 <proof>

end

context *p-equality-depth-div-inv*
begin

— Context for inductive steps leading to Hensel’s Lemma.

context

fixes $p :: 'a$ **and** $f :: 'b$ *poly* **and** $x :: 'b$ **and** $n :: \text{nat}$
assumes $f\text{-deg} : \text{degree } f > 0$
and $f\text{-depth} : \text{set } (\text{coeffs } f) \subseteq p\text{-depth-set } p \text{ } 0$
and $d\text{-hensel} : \text{hensel-seq } p \text{ f } x \text{ n} \in p\text{-depth-set } p \text{ } 0$
and $f : \neg p\text{-equal } p (\text{poly } f (\text{hensel-seq } p \text{ f } x \text{ n})) \text{ } 0$
and $\text{deriv-f} : \neg p\text{-equal } p (\text{poly } (\text{polyderiv } f) (\text{hensel-seq } p \text{ f } x \text{ n})) \text{ } 0$
and $df :$
 $p\text{-depth } p (\text{poly } f (\text{hensel-seq } p \text{ f } x \text{ n})) >$
 $2 * p\text{-depth } p (\text{poly } (\text{polyderiv } f) (\text{hensel-seq } p \text{ f } x \text{ n}))$

begin

lemma *hensel-seq-step-1*:

$p\text{-depth } p (\text{poly } f (\text{hensel-seq } p \text{ f } x (\text{Suc } n))) >$
 $p\text{-depth } p (\text{poly } f ((\text{hensel-seq } p \text{ f } x \text{ n})))$
if $\text{next-f} : \neg p\text{-equal } p (\text{poly } f (\text{hensel-seq } p \text{ f } x (\text{Suc } n))) \text{ } 0$
 <proof>

lemma *hensel-seq-step-2*:

$\neg p\text{-equal } p (\text{poly } (\text{polyderiv } f) (\text{hensel-seq } p \text{ f } x (\text{Suc } n))) \text{ } 0$
 $p\text{-depth } p (\text{poly } (\text{polyderiv } f) (\text{hensel-seq } p \text{ f } x (\text{Suc } n))) =$
 $p\text{-depth } p (\text{poly } (\text{polyderiv } f) (\text{hensel-seq } p \text{ f } x \text{ n}))$
 <proof>

end

— Context for Hensel’s Lemma.

context

fixes $p :: 'a$ **and** $f :: 'b$ *poly* **and** $x :: 'b$
assumes $f\text{-deg} : \text{degree } f > 0$
and $f\text{-depth} : \text{set } (\text{coeffs } f) \subseteq p\text{-depth-set } p \text{ } 0$
and $d\text{-init} : x \in p\text{-depth-set } p \text{ } 0$
and $\text{init-deriv} :$
 $\neg p\text{-equal } p (\text{poly } f \text{ } x) \text{ } 0 \longrightarrow \neg p\text{-equal } p (\text{poly } (\text{polyderiv } f) \text{ } x) \text{ } 0$

and *d-init-deriv*:
 $\neg p\text{-equal } p \text{ (poly } f \ x) \ 0 \longrightarrow$
 $p\text{-depth } p \text{ (poly } f \ x) > 2 * p\text{-depth } p \text{ (poly (polyderiv } f) \ x)$

begin

lemma
hensel-seq-adelic-int : $\text{hensel-seq } p \ f \ x \ n \in p\text{-depth-set } p \ 0$ (**is** ?A)
and *hensel-seq-poly-depth* :
 $\neg p\text{-equal } p \text{ (poly } f \ (\text{hensel-seq } p \ f \ x \ n)) \ 0 \longrightarrow$
 $p\text{-depth } p \text{ (poly } f \ (\text{hensel-seq } p \ f \ x \ n)) \geq p\text{-depth } p \text{ (poly } f \ x) + \text{int } n$
(**is** ?B)
and *hensel-seq-deriv* :
 $\neg p\text{-equal } p \text{ (poly } f \ (\text{hensel-seq } p \ f \ x \ n)) \ 0 \longrightarrow$
 $\neg p\text{-equal } p \text{ (poly (polyderiv } f) \ (\text{hensel-seq } p \ f \ x \ n)) \ 0$
(**is** ?C)
and *hensel-seq-deriv-depth*:
 $\neg p\text{-equal } p \text{ (poly } f \ (\text{hensel-seq } p \ f \ x \ n)) \ 0 \longrightarrow$
 $p\text{-depth } p \text{ (poly (polyderiv } f) \ (\text{hensel-seq } p \ f \ x \ n)) = p\text{-depth } p \text{ (poly (polyderiv$
 $f) \ x)$
(**is** ?D)
and *hensel-seq-depth-ineq*:
 $\neg p\text{-equal } p \text{ (poly } f \ (\text{hensel-seq } p \ f \ x \ n)) \ 0 \longrightarrow$
 $p\text{-depth } p \text{ (poly } f \ (\text{hensel-seq } p \ f \ x \ n)) >$
 $2 * p\text{-depth } p \text{ (poly (polyderiv } f) \ (\text{hensel-seq } p \ f \ x \ n))$
(**is** ?E)
⟨*proof*⟩

lemma *hensel-seq-cauchy*: $p\text{-cauchy } p \text{ (hensel-seq } p \ f \ x)$
⟨*proof*⟩

lemma *p-limseq-poly-hensel-seq*: $p\text{-limseq } p \ (\lambda n. \text{poly } f \ (\text{hensel-seq } p \ f \ x \ n)) \ 0$
⟨*proof*⟩

end

end

locale *p-complete-global-p-equal-depth-div-inv* =
 $\text{global-p-equal-depth-div-inv} + p\text{-complete-global-p-equal-depth}$
begin

lemma *hensel*:
fixes $p :: 'a$ **and** $f :: 'b$ **poly** **and** $x :: 'b$
assumes $\text{degree } f > 0$
and $\text{set (coeffs } f) \subseteq p\text{-depth-set } p \ 0$
and $x \in p\text{-depth-set } p \ 0$
and
 $\neg p\text{-equal } p \text{ (poly } f \ x) \ 0 \longrightarrow \neg p\text{-equal } p \text{ (poly (polyderiv } f) \ x) \ 0$

and
 $\neg p\text{-equal } p \text{ (poly } f \ x) \ 0 \longrightarrow$
 $p\text{-depth } p \text{ (poly } f \ x) > 2 * p\text{-depth } p \text{ (poly (polyderiv } f) \ x)$
obtains z **where** $z \in p\text{-depth-set } p \ 0$ **and** $p\text{-equal } p \text{ (poly } f \ z) \ 0$
 $\langle \text{proof} \rangle$

end

3.7.2 Globally

We use bounded depth to create a global metric.

context $p\text{-equality-depth}$
begin

definition $bdd\text{-global-depth} :: 'b \Rightarrow nat$

where
 $bdd\text{-global-depth } x =$
(if globally-p-equal $x \ 0$ then 0 else $\text{Inf } \{nat \ (p\text{-depth } p \ x + 1) \mid p. \neg p\text{-equal } p \ x \ 0\}$)

— The plus-one is to distinguish depth-0 from negative depth.

lemma $bdd\text{-global-depth-0[simp]: globally-p-equal } x \ 0 \Longrightarrow bdd\text{-global-depth } x = 0$
 $\langle \text{proof} \rangle$

lemma $bdd\text{-global-depthD}:$

$bdd\text{-global-depth } x = \text{Inf } \{nat \ (p\text{-depth } p \ x + 1) \mid p. \neg p\text{-equal } p \ x \ 0\}$
if $\neg \text{globally-p-equal } x \ 0$
 $\langle \text{proof} \rangle$

lemma $bdd\text{-global-depthD-as-image}:$

$bdd\text{-global-depth } x = (\text{INF } p \in \{p. \neg p\text{-equal } p \ x \ 0\}. nat \ (p\text{-depth } p \ x + 1))$
if $\neg \text{globally-p-equal } x \ 0$
 $\langle \text{proof} \rangle$

lemma $bdd\text{-global-depth-cong}:$

$bdd\text{-global-depth } x = bdd\text{-global-depth } y$ **if** $\text{globally-p-equal } x \ y$
 $\langle \text{proof} \rangle$

lemma $bdd\text{-global-depth-le}:$

$bdd\text{-global-depth } x \leq nat \ (p\text{-depth } p \ x + 1)$ **if** $\neg p\text{-equal } p \ x \ 0$
 $\langle \text{proof} \rangle$

lemma $bdd\text{-global-depth-greatest}:$

$bdd\text{-global-depth } x \geq B$
if $\neg p\text{-equal } p \ x \ 0$
and $\forall q. \neg p\text{-equal } q \ x \ 0 \longrightarrow nat \ (p\text{-depth } q \ x + 1) \geq B$
 $\langle \text{proof} \rangle$

lemma $bdd\text{-global-depth-witness}:$

assumes \neg globally-p-equal x 0
obtains p **where** \neg p-equal p x 0 **and** $\text{bdd-global-depth } x = \text{nat } (p\text{-depth } p$ $x + 1)$
 <proof>

lemma *bdd-global-depth-eq-0*:
 $\text{bdd-global-depth } x = 0$ **if** $p\text{-depth } p$ $x < 0$
 <proof>

lemma *bdd-global-depth-eq-0-iff*:
 $\text{bdd-global-depth } x = 0 \iff$
 $(\exists p. p\text{-depth } p$ $x < 0) \vee (\text{globally-p-equal } x$ 0)
 (**is** ?L = ?R)
 <proof>

lemma *bdd-global-depth-diff*: $\text{bdd-global-depth } (x - y) = \text{bdd-global-depth } (y - x)$
 <proof>

lemma *bdd-global-depth-uminus*: $\text{bdd-global-depth } (- x) = \text{bdd-global-depth } x$
 <proof>

lemma *bdd-global-depth-nonarch*:
 $\text{bdd-global-depth } (x + y) \geq \min (\text{bdd-global-depth } x) (\text{bdd-global-depth } y)$
if \neg globally-p-equal $(x + y)$ 0
 <proof>

definition *bdd-global-dist* :: 'b \Rightarrow 'b \Rightarrow real **where**
 $\text{bdd-global-dist } x$ $y =$
 (if globally-p-equal x y then 0 else inverse $(2 \wedge \text{bdd-global-depth } (x - y))$)

lemma *bdd-global-dist-eqD* [simp]: $\text{bdd-global-dist } x$ $y = 0$ **if** globally-p-equal x y
 <proof>

lemma *bdd-global-distD*:
 $\text{bdd-global-dist } x$ $y = \text{inverse } (2 \wedge \text{bdd-global-depth } (x - y))$ **if** \neg globally-p-equal
 x y
 <proof>

lemma *bdd-global-dist-cong*:
 $\text{bdd-global-dist } w$ $x = \text{bdd-global-dist } y$ z
if globally-p-equal w y **and** globally-p-equal x z
 <proof>

lemma *bdd-global-dist-nonneg*: $\text{bdd-global-dist } x$ $y \geq 0$
 <proof>

lemma *bdd-global-dist-bdd*: $\text{bdd-global-dist } x$ $y \leq 1$
 <proof>

lemma *bdd-global-dist-lessI*:

bdd-global-dist $x\ y < e$

if *pos*: $e > 0$

and *depth*:

$\bigwedge p. \neg p\text{-equal}\ p\ x\ y \implies$

$p\text{-depth}\ p\ (x - y) \geq \lfloor \log 2\ (\text{inverse}\ e) \rfloor$

<proof>

lemma *bdd-global-dist-less-imp*:

$p\text{-depth}\ p\ (x - y) \geq \lfloor \log 2\ (\text{inverse}\ e) \rfloor$

if $e > 0$ **and** $e \leq 1$ **and** $\neg p\text{-equal}\ p\ x\ y$ **and** *bdd-global-dist* $x\ y < e$

<proof>

lemma *bdd-global-dist-less-pow2-iff*:

bdd-global-dist $x\ y < \text{inverse}\ (2 \wedge n) \iff$

$(\forall p. \neg p\text{-equal}\ p\ x\ y \implies p\text{-depth}\ p\ (x - y) \geq \text{int}\ n)$

<proof>

lemma *bdd-global-dist-sym*: *bdd-global-dist* $x\ y = \text{bdd-global-dist}\ y\ x$

<proof>

lemma *bdd-global-dist-conv0*: *bdd-global-dist* $x\ y = \text{bdd-global-dist}\ (x - y)\ 0$

<proof>

lemma *bdd-global-dist-conv0'*: *bdd-global-dist* $x\ y = \text{bdd-global-dist}\ 0\ (x - y)$

<proof>

lemma *bdd-global-dist-nonarch*:

bdd-global-dist $x\ y \leq \max\ (\text{bdd-global-dist}\ x\ z)\ (\text{bdd-global-dist}\ y\ z)$

<proof>

lemma *bdd-global-dist-ball-multicentre*:

$\{z. \text{bdd-global-dist}\ y\ z < e\} = \{z. \text{bdd-global-dist}\ x\ z < e\}$

if *bdd-global-dist* $x\ y < e$

<proof>

lemma *bdd-global-dist-ball-at-0*:

$\{z. \text{bdd-global-dist}\ 0\ z < \text{inverse}\ (2 \wedge n)\} = \text{global-depth-set}\ (\text{int}\ n)$

<proof>

lemma *bdd-global-dist-ball-UNIV*:

$\{z. \text{bdd-global-dist}\ x\ z < e\} = \text{UNIV}$ **if** $e > 1$

<proof>

lemma *bdd-global-dist-ball-at-0-normalize*:

$\{z. \text{bdd-global-dist}\ 0\ z < e\} = \text{global-depth-set}\ (\lfloor \log 2\ (\text{inverse}\ e) \rfloor)$

if $e > 0$ **and** $e \leq 1$

<proof>

lemma *bdd-global-dist-ball-translate*:

$\{z. \text{bdd-global-dist } x \ z < e\} = x + o \{z. \text{bdd-global-dist } 0 \ z < e\}$
 $\langle \text{proof} \rangle$

lemma *bdd-global-dist-left-translate-continuous*:

$\text{bdd-global-dist } (x + y) \ (x + z) < e$ **if** $\text{bdd-global-dist } y \ z < e$
 $\langle \text{proof} \rangle$

lemma *bdd-global-dist-right-translate-continuous*:

$\text{bdd-global-dist } (y + x) \ (z + x) < e$ **if** $\text{bdd-global-dist } y \ z < e$
 $\langle \text{proof} \rangle$

definition *global-cauchy-condition* ::

$(\text{nat} \Rightarrow 'b) \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool}$

where

$\text{global-cauchy-condition } X \ n \ K =$
 $(\forall k1 \geq K. \forall k2 \geq K. \forall p.$
 $\neg \text{p-equal } p \ (X \ k1) \ (X \ k2) \longrightarrow \text{nat } (p\text{-depth } p \ (X \ k1 - X \ k2)) > n$
 $)$

lemma *global-cauchy-conditionI*:

$\text{global-cauchy-condition } X \ n \ K$

if

$\bigwedge p \ k \ k'. k \geq K \Longrightarrow k' \geq K \Longrightarrow$
 $\neg \text{p-equal } p \ (X \ k) \ (X \ k') \Longrightarrow \text{nat } (p\text{-depth } p \ (X \ k - X \ k')) > n$

$\langle \text{proof} \rangle$

lemma *global-cauchy-conditionD*:

$\text{nat } (p\text{-depth } p \ (X \ k - X \ k')) > n$

if $\text{global-cauchy-condition } X \ n \ K$ **and** $k \geq K$ **and** $k' \geq K$

and $\neg \text{p-equal } p \ (X \ k) \ (X \ k')$

$\langle \text{proof} \rangle$

lemma *global-cauchy-condition-mono-seq-tail*:

$\text{global-cauchy-condition } X \ n \ K \Longrightarrow \text{global-cauchy-condition } X \ n \ K'$

if $K' \geq K$

$\langle \text{proof} \rangle$

lemma *global-cauchy-condition-mono-depth*:

$\text{global-cauchy-condition } X \ n \ K \Longrightarrow \text{global-cauchy-condition } X \ m \ K$

if $m \leq n$

$\langle \text{proof} \rangle$

abbreviation *globally-cauchy* $X \equiv (\forall n. \exists K. \text{global-cauchy-condition } X \ n \ K)$

lemma *global-cauchy-condition-LEAST*:

$\text{global-cauchy-condition } X \ n$ (*LEAST* $K. \text{global-cauchy-condition } X \ n \ K$) **if** *globally-cauchy* X

$\langle \text{proof} \rangle$

lemma *global-cauchy-condition-LEAST-mono*:
 $(LEAST\ K.\ global\ cauchy\ condition\ X\ m\ K) \leq (LEAST\ K.\ global\ cauchy\ condition\ X\ n\ K)$
if *globally-cauchy X* **and** $m \leq n$
 ⟨*proof*⟩

definition *bdd-global-uniformity* :: ('b × 'b) filter
where *bdd-global-uniformity* = (INF e∈{0 <..<}. principal {(x, y). *bdd-global-dist* x y < e})

end

context *global-p-equal-depth*
begin

sublocale *metric-space-by-bdd-depth*:
metric-space *bdd-global-dist* *bdd-global-uniformity*
 $\lambda U. \forall x \in U.$
eventually ($\lambda(x', y). x' = x \longrightarrow y \in U$) *bdd-global-uniformity*
 ⟨*proof*⟩

lemma *nonneg-global-depth-set-LIMSEQ-closed*:
 $x \in global\ depth\ set\ (int\ n)$
if *lim*: *metric-space-by-bdd-depth.LIMSEQ* X x
and *seq*: $\forall_F k$ in *sequentially*. $X\ k \in global\ depth\ set\ (int\ n)$
 ⟨*proof*⟩

lemma *globally-cauchy-vs-bdd-metric-Cauchy*:
globally-cauchy X = *metric-space-by-bdd-depth.Cauchy* X **for** X :: nat \Rightarrow 'b
 ⟨*proof*⟩

end

context *global-p-equal-depth-no-zero-divisors*
begin

lemma *bdd-global-dist-bdd-mult-continuous*:
bdd-global-dist (w * x) (w * y) < e
if *pos* : e > 0
and *bdd* : $\forall p. \neg p\ equal\ p\ w\ 0 \longrightarrow p\ depth\ p\ w \geq n$
and *input*: *bdd-global-dist* x y < min (e * 2 powi (n - 1)) 1
 ⟨*proof*⟩

lemma *bdd-global-dist-limseq-bdd-mult*:
metric-space-by-bdd-depth.LIMSEQ ($\lambda k. w * X\ k$) (w * x)
if *bdd*: $\forall p. \neg p\ equal\ p\ w\ 0 \longrightarrow p\ depth\ p\ w \geq n$
and *seq*: *metric-space-by-bdd-depth.LIMSEQ* X x
 ⟨*proof*⟩

end

context *global-p-equal-depth-div-inv-w-inj-index-consts*
begin

lemma *bdd-global-dist-limseq-global-uniformizer-powi-mult*:
metric-space-by-bdd-depth.LIMSEQ
($\lambda k. \text{global-uniformizer powi } n * X k$) (*global-uniformizer powi } n * x*)
if *metric-space-by-bdd-depth.LIMSEQ* *X x*
(*proof*)

lemma *global-depth-set-LIMSEQ-closed*:
x \in *global-depth-set* *n*
if *lim* : *metric-space-by-bdd-depth.LIMSEQ* *X x*
and *depth*: $\forall_F k$ *in sequentially. X k* \in *global-depth-set* *n*
(*proof*)

end

3.8 Notation

consts

p-equal :: 'a \Rightarrow 'b \Rightarrow 'b \Rightarrow bool
globally-p-equal ::
'b \Rightarrow 'b \Rightarrow bool
($(-/ \simeq_{\forall p} / -)$ [51, 51] 50)
p-restrict :: 'b \Rightarrow ('a \Rightarrow bool) \Rightarrow 'b (**infixl** *prestrict* 70)
p-depth :: 'a \Rightarrow 'b \Rightarrow int
p-depth-set :: 'a \Rightarrow int \Rightarrow 'b set ($\mathcal{P}_{@-}$)
global-depth-set :: int \Rightarrow 'b set ($\mathcal{P}_{\forall p-}$)
global-unfrmzr-pows :: ('a \Rightarrow 'c) \Rightarrow 'b

abbreviation *p-equal-notation* :: 'b \Rightarrow 'a \Rightarrow 'b \Rightarrow bool
($(-/ \simeq / -)$ [51, 51, 51] 50)
where *p-equal-notation* *x p y* \equiv *p-equal* *p x y*

abbreviation *p-nequal-notation* :: 'b \Rightarrow 'a \Rightarrow 'b \Rightarrow bool
($(-/ \neg \simeq / -)$ [51, 51, 51] 50)
where *p-nequal-notation* *x p y* \equiv \neg *p-equal* *p x y*

abbreviation *p-depth-notation* :: 'b \Rightarrow 'a \Rightarrow int
($(-^{\circ-})$ [51, 51] 50)
where *p-depth-notation* *x p* \equiv *p-depth* *p x*

abbreviation *p-depth-set-0-notation* :: 'a \Rightarrow 'b set ($\mathcal{O}_{@-}$)
where *p-depth-set-0-notation* *p* \equiv *p-depth-set* *p 0*

abbreviation *p-depth-set-1-notation* :: 'a \Rightarrow 'b set ($\mathcal{P}_{@-}$)

where *p*-depth-set-1-notation $p \equiv p$ -depth-set p 1
abbreviation *global-depth-set-0-notation* :: 'b set ($\mathcal{O}_{\forall p}$)
where *global-depth-set-0-notation* \equiv *global-depth-set* 0
abbreviation *global-depth-set-1-notation* :: 'b set ($\mathcal{P}_{\forall p}$)
where *global-depth-set-1-notation* \equiv *global-depth-set* 1
abbreviation $\mathfrak{p} \equiv$ *global-unfrmzr-pows*
end

theory *FLS-Prime-Equiv-Depth*

imports

Parametric-Equiv-Depth

HOL.Equiv-Relations

HOL-Computational-Algebra.Formal-Laurent-Series

begin

unbundle *fps-syntax*

4 Equivalence of sequences of formal Laurent series relative to divisibility of partial sums by a fixed prime

4.1 Preliminaries

4.1.1 Lift/transfer of equivalence relations.

lemma *reflp-transfer*:

reflp $r \implies \text{reflp } (\lambda x y. r (f x) (f y))$
<proof>

lemma *symp-transfer*:

symp $r \implies \text{symp } (\lambda x y. r (f x) (f y))$
<proof>

lemma *transp-transfer*:

transp $r \implies \text{transp } (\lambda x y. r (f x) (f y))$
<proof>

lemma *equivp-transfer*:

equivp $r \implies \text{equivp } (\lambda x y. r (f x) (f y))$
<proof>

4.1.2 An additional fact about products/sums

lemma (in *comm-monoid-set*) *atLeastAtMost-int-shift-bounds*:
 $F\ g\ \{m..n\} = F\ (g \circ \text{plus } m \circ \text{int})\ \{..nat\ (n - m)\}$
if $m \leq n$
for $m\ n :: \text{int}$
<proof>

4.1.3 An additional fact about algebraic powers of functions

lemma *pow-fun-apply*: $(f \wedge n)\ x = (f\ x) \wedge n$
<proof>

4.1.4 Some additional facts about formal power and Laurent series

lemma *fps-shift-fps-mult-by-fps-const*:
fixes $c :: 'a::\{\text{comm-monoid-add,mult-zero}\}$
shows $\text{fps-shift } n\ (\text{fps-const } c * f) = \text{fps-const } c * \text{fps-shift } n\ f$
and $\text{fps-shift } n\ (f * \text{fps-const } c) = \text{fps-shift } n\ f * \text{fps-const } c$
<proof>

lemma *fps-shift-mult-Suc*:
fixes $f\ g :: 'a::\{\text{comm-monoid-add,mult-zero}\}\ \text{fps}$
shows $\text{fps-shift } (\text{Suc } n)\ (f * g)$
 $= \text{fps-const } (f\$0) * \text{fps-shift } (\text{Suc } n)\ g + \text{fps-shift } n\ (\text{fps-shift } 1\ f * g)$
and $\text{fps-shift } (\text{Suc } n)\ (f * g)$
 $= \text{fps-shift } n\ (f * \text{fps-shift } 1\ g) + \text{fps-shift } (\text{Suc } n)\ f * \text{fps-const } (g\$0)$
<proof>

lemma *fls-subdegree-minus'*:
 $\text{fls-subdegree } (f - g) = \text{fls-subdegree } f$
if $f \neq 0$ **and** $\text{fls-subdegree } g > \text{fls-subdegree } f$
<proof>

lemma *fls-regpart-times-fps-X*:
fixes $f :: 'a::\{\text{comm-monoid-add,mult-zero,monoid-mult}\}\ \text{fls}$
assumes $\text{fls-subdegree } f \geq 0$
shows $\text{fls-regpart } f * \text{fps-X} = \text{fls-regpart } (\text{fls-shift } (-1)\ f)$
and $\text{fps-X} * \text{fls-regpart } f = \text{fls-regpart } (\text{fls-shift } (-1)\ f)$
<proof>

lemma *fls-regpart-times-fps-X-power*:
fixes $f :: 'a::\text{semiring-1}\ \text{fls}$
assumes $\text{fls-subdegree } f \geq 0$
shows $\text{fls-regpart } f * \text{fps-X} \wedge n = \text{fls-regpart } (\text{fls-shift } (-n)\ f)$
and $\text{fps-X} \wedge n * \text{fls-regpart } f = \text{fls-regpart } (\text{fls-shift } (-n)\ f)$
<proof>

lemma *fls-base-factor-uminus*: $\text{fls-base-factor } (-f) = -\ \text{fls-base-factor } f$

<proof>

4.2 Partial evaluation of formal power series

Make a degree- n polynomial out of a formal power series.

definition *poly-of-fps*

where $\text{poly-of-fps } f \ n = \text{Abs-poly } (\lambda i. \text{if } i \leq n \text{ then } f \$ i \text{ else } 0)$

lemma *if-then-MOST-zero*:

$\forall_{\infty} x. (\text{if } P \ x \ \text{then } f \ x \ \text{else } 0) = 0$

if $\forall_{\infty} x. \neg P \ x$

<proof>

lemma *truncated-fps-eventually-zero*:

$(\lambda i. \text{if } i \leq n \text{ then } f \$ i \text{ else } 0) \in \{g. \forall_{\infty} n. g \ n = 0\}$

<proof>

lemma *coeff-poly-of-fps[simp]*:

$\text{coeff } (\text{poly-of-fps } f \ n) \ i = (\text{if } i \leq n \text{ then } f \$ i \text{ else } 0)$

<proof>

lemma *poly-of-fps-0[simp]*: $\text{poly-of-fps } 0 \ n = 0$

<proof>

lemma *poly-of-fps-0th-conv-pCons[simp]*: $\text{poly-of-fps } f \ 0 = [:f \$ 0 :]$

<proof>

lemma *poly-of-fps-degree*: $\text{degree } (\text{poly-of-fps } f \ n) \leq n$

<proof>

lemma *poly-of-fps-Suc*:

$\text{poly-of-fps } f \ (\text{Suc } n) = \text{poly-of-fps } f \ n + \text{monom } (f \ \$ \ \text{Suc } n) \ (\text{Suc } n)$

<proof>

lemma *poly-of-fps-Suc-nth-eq0*:

$\text{poly-of-fps } f \ (\text{Suc } n) = \text{poly-of-fps } f \ n \ \text{if } f \ \$ \ \text{Suc } n = 0$

<proof>

lemma *poly-of-fps-Suc-conv-pCons-shift*:

$\text{poly-of-fps } f \ (\text{Suc } n) = \text{pCons } (f \$ 0) \ (\text{poly-of-fps } (\text{fps-shift } 1 \ f) \ n)$

<proof>

abbreviation *fps-parteval* $f \ x \ n \equiv \text{poly } (\text{poly-of-fps } f \ n) \ x$

abbreviation *fls-base-parteval* $f \equiv \text{fps-parteval } (\text{fls-base-factor-to-fps } f)$

lemma *fps-parteval-0[simp]*: $\text{fps-parteval } 0 \ x \ n = 0$

<proof>

lemma *fps-parteval-0th[simp]*: $\text{fps-parteval } f \ x \ 0 = f \ \$ \ 0$

<proof>

lemma *fps-parteval-Suc*:

$fps\text{-parteval } f \ x \ (Suc \ n) = fps\text{-parteval } f \ x \ n + (f \ \$ \ Suc \ n) * x \ ^ \ Suc \ n$
for $f :: 'a::comm\text{-semiring-1} \ fps$
<proof>

lemma *fps-parteval-Suc-cons*:

$fps\text{-parteval } f \ x \ (Suc \ n) = f \ \$ \ 0 + x * fps\text{-parteval } (fps\text{-shift } 1 \ f) \ x \ n$
<proof>

lemma *fps-parteval-at-0[simp]*: $fps\text{-parteval } f \ 0 \ n = f \ \$ \ 0$

<proof>

lemma *fps-parteval-conv-sum*:

$fps\text{-parteval } f \ x \ n = (\sum k \leq n. f \ \$ \ k * x \ ^ \ k)$ **for** $x :: 'a::comm\text{-semiring-1}$
<proof>

lemma *fls-base-parteval-conv-sum*:

$fls\text{-base-parteval } f \ x \ n = (\sum k \leq n. f \ \$ \ (fls\text{-subdegree } f + int \ k) * x \ ^ \ k)$
for $f :: 'a::comm\text{-semiring-1} \ fls$ **and** $x :: 'a$ **and** $n :: nat$
<proof>

lemma *fps-parteval-fps-const[simp]*:

$fps\text{-parteval } (fps\text{-const } c) \ x \ n = c$
<proof>

lemma *fps-parteval-1[simp]*: $fps\text{-parteval } 1 \ x \ n = 1$

<proof>

lemma *fps-parteval-mult-by-fps-const*:

$fps\text{-parteval } (fps\text{-const } c * f) \ x \ n = c * fps\text{-parteval } f \ x \ n$
<proof>

lemma *fps-parteval-plus[simp]*:

$fps\text{-parteval } (f + g) \ x \ n = fps\text{-parteval } f \ x \ n + fps\text{-parteval } g \ x \ n$
<proof>

lemma *fps-parteval-uminus[simp]*:

$fps\text{-parteval } (-f) \ x \ n = - fps\text{-parteval } f \ x \ n$ **for** $f :: 'a::comm\text{-ring} \ fps$
<proof>

lemma *fps-parteval-minus[simp]*:

$fps\text{-parteval } (f - g) \ x \ n = fps\text{-parteval } f \ x \ n - fps\text{-parteval } g \ x \ n$
for $f :: 'a::comm\text{-ring} \ fps$
<proof>

lemma *fps-parteval-mult-fps-X*:

fixes $f :: 'a::comm\text{-semiring-1} \ fps$

shows $\text{fps-parteval } (f * \text{fps-X}) x (\text{Suc } n) = x * \text{fps-parteval } f x n$
and $\text{fps-parteval } (\text{fps-X} * f) x (\text{Suc } n) = x * \text{fps-parteval } f x n$
 $\langle \text{proof} \rangle$

lemma *fps-parteval-mult-fps-X-power*:

fixes $f :: 'a::\text{comm-semiring-1 } \text{fps}$

assumes $n \geq m$

shows $\text{fps-parteval } (f * \text{fps-X} \hat{\ } m) x n = x \hat{\ } m * \text{fps-parteval } f x (n - m)$

and $\text{fps-parteval } (\text{fps-X} \hat{\ } m * f) x n = x \hat{\ } m * \text{fps-parteval } f x (n - m)$

$\langle \text{proof} \rangle$

lemma *fps-parteval-mult-right*:

$\text{fps-parteval } (f * g) x n =$

$\text{fps-parteval } (\text{Abs-fps } (\lambda i. f \$ i * \text{fps-parteval } g x (n - i))) x n$

$\langle \text{proof} \rangle$

lemma *fps-parteval-mult-left*:

$\text{fps-parteval } (f * g) x n =$

$\text{fps-parteval } (\text{Abs-fps } (\lambda i. \text{fps-parteval } f x (n - i) * g \$ i)) x n$

$\langle \text{proof} \rangle$

lemma *fps-parteval-mult-conv-sum*:

fixes $f g :: 'a::\text{comm-semiring-1 } \text{fps}$

shows $\text{fps-parteval } (f * g) x n = (\sum k \leq n. f \$ k * \text{fps-parteval } g x (n - k) * x \hat{\ } k)$

and $\text{fps-parteval } (f * g) x n = (\sum k \leq n. \text{fps-parteval } f x (n - k) * g \$ k * x \hat{\ } k)$

$\langle \text{proof} \rangle$

lemma *fps-parteval-fls-base-factor-to-fps-diff*:

$\text{fls-base-parteval } (f - g) x n = \text{fls-base-parteval } f x n$

if $f \neq 0$ **and** $\text{fls-subdegree } g > \text{fls-subdegree } f + (\text{int } n)$

for $f g :: 'a::\text{comm-ring-1 } \text{fls}$

$\langle \text{proof} \rangle$

4.3 Vanishing of formal power series relative to divisibility of all partial sums

definition *fps-pvanishes* $:: 'a::\text{comm-semiring-1} \Rightarrow 'a \text{ fps} \Rightarrow \text{bool}$

where $\text{fps-pvanishes } p f \equiv (\forall n::\text{nat. } p \hat{\ } \text{Suc } n \text{ dvd } (\text{fps-parteval } f p n))$

lemma *fps-pvanishesD*: $\text{fps-pvanishes } p f \implies p \hat{\ } \text{Suc } n \text{ dvd } (\text{fps-parteval } f p n)$

$\langle \text{proof} \rangle$

lemma *fps-pvanishesI*:

$\text{fps-pvanishes } p f$ **if** $\bigwedge n::\text{nat. } p \hat{\ } \text{Suc } n \text{ dvd } (\text{fps-parteval } f p n)$

$\langle \text{proof} \rangle$

lemma *fps-pvanishes-0[simp]*: $\text{fps-pvanishes } p 0$

$\langle \text{proof} \rangle$

lemma *fps-pvanishes-at-0*[simp]: *fps-pvanishes* 0 *f* = (*f*\$0 = 0)
 ⟨proof⟩

lemma *fps-pvanishes-at-unit*: *fps-pvanishes* *p f* **if** *is-unit* *p*
 ⟨proof⟩

lemma *fps-pvanishes-one-iff*:
fps-pvanishes *p 1* = *is-unit* *p*
 ⟨proof⟩

lemma *fps-pvanishes-uminus*[simp]:
fps-pvanishes *p* (*-f*) **if** *fps-pvanishes* *p f* **for** *f* :: '*a*::*comm-ring-1* *fps*
 ⟨proof⟩

lemma *fps-pvanishes-add*[simp]:
fps-pvanishes *p* (*f + g*) **if** *fps-pvanishes* *p f* **and** *fps-pvanishes* *p g*
for *f* :: '*a*::*comm-semiring-1* *fps*
 ⟨proof⟩

lemma *fps-pvanishes-minus*[simp]:
fps-pvanishes *p* (*f - g*) **if** *fps-pvanishes* *p f* **and** *fps-pvanishes* *p g*
for *f* :: '*a*::*comm-ring-1* *fps*
 ⟨proof⟩

lemma *fps-pvanishes-mult*: *fps-pvanishes* *p* (*f * g*) **if** *fps-pvanishes* *p g*
 ⟨proof⟩

lemma *fps-pvanishes-mult-fps-X-iff*:
assumes (*p*::'*a*::*idom*) ≠ 0
shows *fps-pvanishes* *p f* ↔ *fps-pvanishes* *p* (*f * fps-X*)
and *fps-pvanishes* *p f* ↔ *fps-pvanishes* *p* (*fps-X * f*)
 ⟨proof⟩

lemma *fps-pvanishes-mult-fps-X-power-iff*:
assumes (*p*::'*a*::*idom*) ≠ 0
shows *fps-pvanishes* *p f* ↔ *fps-pvanishes* *p* (*f * fps-X* ^ *n*)
and *fps-pvanishes* *p f* ↔ *fps-pvanishes* *p* (*fps-X* ^ *n * f*)
 ⟨proof⟩

4.4 Equivalence and depth of formal Laurent series relative to a prime

4.4.1 Definition of equivalence and basic facts

abbreviation *fps-primevanishes* *p* ≡ *fps-pvanishes* (*Rep-prime* *p*)

overloading

p-equal-fls ≡
p-equal :: '*a*::*nontrivial-factorial-comm-ring* *prime* ⇒
 '*a* *fls* ⇒ '*a* *fls* ⇒ *bool*

begin

definition *p-equal-fls* ::

'a::nontrivial-factorial-comm-ring prime \Rightarrow

'a fls \Rightarrow *'a fls* \Rightarrow *bool*

where

p-equal-fls p f g \equiv *fps-primevanishes p (fls-base-factor-to-fps (f - g))*

end

context

fixes p :: 'a :: nontrivial-factorial-comm-ring prime

begin

lemma *p-equal-flsD*:

f \simeq_p *g* \Longrightarrow *fps-primevanishes p (fls-base-factor-to-fps (f - g))*

for *f g* :: *'a fls*

<proof>

lemma *p-equal-fls-imp-p-dvd*: *p pdvd ((f - g) \$\$ (fls-subdegree (f - g)))*

if *f* \simeq_p *g*

for *f g* :: *'a fls*

<proof>

lemma *p-equal-flsI*:

fps-primevanishes p (fls-base-factor-to-fps (f - g)) \Longrightarrow *f* \simeq_p *g*

for *f g* :: *'a fls*

<proof>

lemma *p-equal-fls-conv-p-equal-0*:

f \simeq_p *g* \longleftrightarrow *(f - g)* \simeq_p *0*

for *f g* :: *'a fls*

<proof>

lemma *p-equal-fls-refl*: *f* \simeq_p *f* **for** *f*:: *'a fls*

<proof>

lemma *p-equal-fls-sym*:

f \simeq_p *g* \Longrightarrow *g* \simeq_p *f* **for** *f g* :: *'a fls*

<proof>

lemma *p-equal-fls-shift-iff*:

f \simeq_p *g* \longleftrightarrow *(fls-shift n f)* \simeq_p *(fls-shift n g)*

for *f g* :: *'a fls*

<proof>

lemma *p-equal-fls-extended-sum-iff*:

f \simeq_p *g* \longleftrightarrow

$(\forall n. (\text{Rep-prime } p) \wedge \text{Suc } n \text{ dvd}$

$(\sum_{k \leq n}. (f - g) \$(N + \text{int } k) * (\text{Rep-prime } p) \wedge k))$
if $N \leq \min (\text{fls-subdegree } f) (\text{fls-subdegree } g)$
 <proof>

lemma *p-equal-fls-extended-intsum-iff*:

$f \simeq_p g \iff$
 $(\forall n \geq N. (\text{Rep-prime } p) \wedge \text{Suc } (\text{nat } (n - N)) \text{ dvd}$
 $(\sum_{k=N..n}. (f - g) \$(k) * (\text{Rep-prime } p) \wedge \text{nat } (k - N)))$
if $N \leq \min (\text{fls-subdegree } f) (\text{fls-subdegree } g)$
 <proof>

lemma *fls-p-equal0-extended-sum-iff*:

$f \simeq_p 0 \iff$
 $(\forall n. (\text{Rep-prime } p) \wedge \text{Suc } n \text{ dvd}$
 $(\sum_{k \leq n}. f \$(N + \text{int } k) * (\text{Rep-prime } p) \wedge k))$
if $N \leq \text{fls-subdegree } f$
 <proof>

lemma *fls-p-equal0-extended-intsum-iff*:

$f \simeq_p 0 \iff$
 $(\forall n \geq N. (\text{Rep-prime } p) \wedge \text{Suc } (\text{nat } (n - N)) \text{ dvd}$
 $(\sum_{k=N..n}. f \$(k) * (\text{Rep-prime } p) \wedge \text{nat } (k - N)))$
if $N \leq \text{fls-subdegree } f$
 <proof>

end

context

fixes $p :: 'a :: \text{nontrivial-factorial-idom prime}$
begin

lemma *fls-1-not-p-equal-0*: $(1 :: 'a \text{ fls}) \neg \simeq_p 0$
 <proof>

lemma *p-equal-fls0-nonneg-subdegree*:

$f \simeq_p 0 \iff \text{fps-primevanishes } p (\text{fls-regpart } f)$
if $\text{fls-subdegree } f \geq 0$
 <proof>

lemma *p-equal-fls-nonneg-subdegree*:

$f \simeq_p g \iff \text{fps-primevanishes } p (\text{fls-regpart } (f - g))$
if $\text{fls-subdegree } f \geq 0$ **and** $\text{fls-subdegree } g \geq 0$
 <proof>

lemma *p-equal-fls-trans-to-0*:

$f \simeq_p g$ **if** $f \simeq_p 0$ **and** $g \simeq_p 0$ **for** $f g :: 'a \text{ fls}$
 <proof>

lemma *p-equal-fls-trans*:

```

   $f \simeq_p g \implies g \simeq_p h \implies$ 
   $f \simeq_p h$ 
  for  $f g h :: 'a \text{ fls}$ 
  <proof>

end

global-interpretation p-equality-fls:
  p-equality-1
  p-equal ::  $'a :: \text{nontrivial-factorial-idom prime} \implies$ 
   $'a \text{ fls} \implies 'a \text{ fls} \implies \text{bool}$ 
  <proof>

overloading
  globally-p-equal-fls  $\equiv$ 
  globally-p-equal ::
   $'a :: \text{nontrivial-factorial-idom fls} \implies 'a \text{ fls} \implies \text{bool}$ 
begin

definition globally-p-equal-fls ::
   $'a :: \text{nontrivial-factorial-idom fls} \implies 'a \text{ fls} \implies \text{bool}$ 
  where globally-p-equal-fls[simp]:
  globally-p-equal-fls = p-equality-fls.globally-p-equal

end

context
  fixes  $p :: 'a :: \text{nontrivial-factorial-idom prime}$ 
begin

lemma p-equal-fls-const-iff:
   $(\text{fls-const } c) \simeq_p (\text{fls-const } d) \iff c = d$  for  $c d :: 'a$ 
  <proof>

lemma p-equal-fls-const-0-iff:
   $(\text{fls-const } c) \simeq_p 0 \iff c = 0$  for  $c :: 'a$ 
  <proof>

lemma p-equal-fls-X-intpow-iff:
   $((\text{fls-X-intpow } m :: 'a \text{ fls}) \simeq_p (\text{fls-X-intpow } n)) = (m = n)$ 
  <proof>

lemma fls-X-intpow-nequiv0:  $(\text{fls-X-intpow } m :: 'a \text{ fls}) \not\simeq_p 0$ 
  <proof>

end

```

4.4.2 Depth as maximum subdegree of equivalent series

Each nonzero equivalence class will have a maximum subdegree.

lemma *no-greatest-p-equal-subdegree*:

$f \simeq_p 0$
if $\forall n :: \text{int. } (\exists g. \text{fls-subdegree } g > n \wedge f \simeq_p g)$
for $p :: 'a::\text{nontrivial-factorial-comm-ring prime}$ **and** $f :: 'a \text{ fls}$
<proof>

abbreviation

p-equal-fls-simplified-set $p f \equiv$
 $\{g. f \simeq_p g \wedge \text{fls-subdegree } g \geq \text{fls-subdegree } f\}$
— We restrict to those equivalent series that represent a simplification of the given series so that the image of the collection under taking subdegrees is finite.

context

fixes $p :: 'a::\text{nontrivial-factorial-comm-ring prime}$
and $f :: 'a \text{ fls}$
begin

lemma *p-equal-fls-simplified-nempty*:

$\text{fls-subdegree } '(p\text{-equal-fls-simplified-set } p f) \neq \{\}$
<proof>

lemma *p-equal-fls-simplified-finite*:

$\text{finite } (\text{fls-subdegree } '(p\text{-equal-fls-simplified-set } p f))$ **if** $f \not\simeq_p 0$
<proof>

end

overloading

p-depth-fls \equiv
 $p\text{-depth} :: 'a::\text{nontrivial-factorial-comm-ring prime} \Rightarrow 'a \text{ fls} \Rightarrow \text{int}$
p-depth-const \equiv
 $p\text{-depth} :: 'a::\text{nontrivial-factorial-comm-ring prime} \Rightarrow 'a \Rightarrow \text{int}$

begin

definition

p-depth-fls $:: 'a::\text{nontrivial-factorial-comm-ring prime} \Rightarrow 'a \text{ fls} \Rightarrow \text{int}$
where
p-depth-fls $p f \equiv$
 $(\text{if } f \simeq_p 0 \text{ then } 0 \text{ else } \text{Max } (\text{fls-subdegree } '(p\text{-equal-fls-simplified-set } p f)))$

definition

p-depth-const $:: 'a::\text{nontrivial-factorial-comm-ring prime} \Rightarrow 'a \Rightarrow \text{int}$
where
p-depth-const $p c \equiv p\text{-depth } p (\text{fls-const } c)$

end

context

fixes $p :: 'a :: \text{nontrivial-factorial-comm-ring prime}$
begin

lemma $p\text{-depth-fls-eqI}$:

$f^{\circ p} = \text{fls-subdegree } g$
if $f \not\approx_p 0$ **and** $f \approx_p g$
and $\bigwedge h. f \approx_p h \implies \text{fls-subdegree } h \leq \text{fls-subdegree } g$
for $f g :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma $p\text{-depth-fls-p-equal0[simp]}$:

$(f^{\circ p}) = 0$ **if** $f \approx_p 0$ **for** $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma $p\text{-depth-fls0[simp]}$: $(0 :: 'a \text{ fls})^{\circ p} = 0$

$\langle \text{proof} \rangle$

lemma $p\text{-depth-fls-n0D}$:

$f^{\circ p} = \text{Max } (\text{fls-subdegree } ` (p\text{-equal-fls-simplified-set } p f))$
if $f \not\approx_p 0$
for $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma $p\text{-depth-flsE}$:

fixes $f :: 'a \text{ fls}$
assumes $f \not\approx_p 0$
obtains g **where** $f \approx_p g$ **and** $\text{fls-subdegree } g = (f^{\circ p})$
and $\forall h. f \approx_p h \implies \text{fls-subdegree } h \leq \text{fls-subdegree } g$
 $\langle \text{proof} \rangle$

lemma $p\text{-depth-fls-ge-p-equal-subdegree}$:

$(f^{\circ p}) \geq \text{fls-subdegree } g$
if $f \not\approx_p 0$ **and** $f \approx_p g$
for $f g :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma $p\text{-depth-fls-ge-p-equal-subdegree}'$:

$(f^{\circ p}) \geq \text{fls-subdegree } f$ **if** $f \not\approx_p 0$ **for** $f g :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma $p\text{-equal-fls-simplified-set-shift}$:

$p\text{-equal-fls-simplified-set } p (\text{fls-shift } n f) = \text{fls-shift } n ` p\text{-equal-fls-simplified-set } p f$
if $f \not\approx_p 0$
for $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *p-equal-fls-simplified-set-shift-subdegrees*:
 $\text{fls-subdegree } 'p\text{-equal-fls-simplified-set } p \text{ (fls-shift } n \text{ } f) =$
 $(\lambda x. x - n) ' \text{fls-subdegree } 'p\text{-equal-fls-simplified-set } p \text{ } f$
if $f \not\sim_p 0$
for $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *p-depth-fls-shift*:
 $(\text{fls-shift } n \text{ } f)^{\circ p} = (f^{\circ p}) - n$ **if** $f \not\sim_p 0$
for $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

end

context
fixes $p :: 'a :: \text{nontrivial-factorial-idom prime}$
and $f \ g :: 'a \text{ fls}$
begin

lemma *p-depth-fls-p-equal*: $(f^{\circ p}) = (g^{\circ p})$ **if** $f \simeq_p g$
 $\langle \text{proof} \rangle$

lemma *p-depth-fls-can-simplify-iff*:
 $p \text{ pdvd } g \ \S\ \text{fls-subdegree } g = \text{fls-subdegree } g < (f^{\circ p})$
if $f\text{-nequiv0} : f \not\sim_p 0$
and $fg\text{-equiv} : f \simeq_p g$
 $\langle \text{proof} \rangle$

lemma *p-depth-fls-rep-iff*:
 $(f^{\circ p}) = \text{fls-subdegree } g \iff$
 $\neg p \text{ pdvd } g \ \S\ \text{fls-subdegree } g$
if $f \not\sim_p 0$ **and** $f \simeq_p g$
 $\langle \text{proof} \rangle$

end

global-interpretation *p-depth-fls*:
 $p\text{-equality-depth-no-zero-divisors-1}$
 $p\text{-equal} :: 'a :: \text{nontrivial-factorial-idom prime} \Rightarrow$
 $'a \text{ fls} \Rightarrow 'a \text{ fls} \Rightarrow \text{bool}$
 $p\text{-depth} :: 'a \text{ prime} \Rightarrow 'a \text{ fls} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

context
fixes $p :: 'a :: \text{nontrivial-factorial-idom prime}$
begin

lemma *p-depth-fls1[simp]*: $(1 :: 'a \text{ fls})^{\circ p} = 0$
 $\langle \text{proof} \rangle$

lemma *p-depth-fls-X*: $(fls-X::'a\ fls)^{op} = 1$
 ⟨proof⟩

lemma *p-depth-fls-X-inv*: $(fls-X-inv::'a\ fls)^{op} = -1$
 ⟨proof⟩

lemma *p-depth-fls-X-intpow*: $((fls-X-intpow\ n)::'a\ fls)^{op} = n$
 ⟨proof⟩

lemma *p-depth-const*:
 $c^{op} = pmultiplicity\ p\ c$ for $c :: 'a$
 ⟨proof⟩

lemma *p-depth-prime-const*: $(Rep-prime\ p)^{op} = 1$
 ⟨proof⟩

end

4.5 Reduction relative to a prime

4.5.1 Of scalars

class *nontrivial-euclidean-ring-cancel* = *nontrivial-factorial-comm-ring* + *euclidean-semiring-cancel*
begin
subclass *nontrivial-factorial-idom* ⟨proof⟩
subclass *euclidean-ring-cancel* ⟨proof⟩
end

instance *int* :: *nontrivial-euclidean-ring-cancel* ⟨proof⟩

abbreviation *pdiv* ::
 $'a \Rightarrow 'a::nontrivial-euclidean-ring-cancel\ prime \Rightarrow 'a$ (**infix** *pdiv* 70)
where $a\ pdiv\ p \equiv a\ div\ (Rep-prime\ p)$

consts *p-modulo* :: $'a \Rightarrow 'b\ prime \Rightarrow 'a$ (**infixl** *pmod* 70)

overloading *p-modulo-scalar* $\equiv p-modulo$:: $'a \Rightarrow 'a\ prime \Rightarrow 'a$
begin
definition *p-modulo-scalar* ::
 $'a::nontrivial-euclidean-ring-cancel \Rightarrow 'a\ prime \Rightarrow 'a$
where *p-modulo-scalar-def[simp]*: $p-modulo-scalar\ a\ p \equiv a\ mod\ (Rep-prime\ p)$
end

class *nontrivial-unique-euclidean-ring* = *nontrivial-factorial-comm-ring* + *unique-euclidean-semiring*
 + **assumes** *division-segment-prime*: $prime\ p \Longrightarrow division-segment\ p = 1$
begin
subclass *nontrivial-euclidean-ring-cancel* ⟨proof⟩
end

instance *int* :: *nontrivial-unique-euclidean-ring*
 ⟨*proof*⟩

lemma *division-segment-1*: *division-segment* (1::'a::*unique-euclidean-semiring*) =
 1
 ⟨*proof*⟩

lemma *one-pdiv*: (1::'a) *pdiv* p = 0
and *one-pmod*: (1::'a) *pmod* p = 1
for p :: 'a::*nontrivial-unique-euclidean-ring prime*
 ⟨*proof*⟩

4.5.2 Inductive partial reduction of formal Laurent series

This inductive function reduces one term at a time. The *nat* argument specifies how far beyond the subdegree to reduce, so that all terms to *one less* than the *nat* argument beyond the subdegree are reduced, and the term at the *nat* argument beyond the subdegree is updated to include the carry.

primrec *fls-partially-p-modulo-rep* ::
 'a::*nontrivial-euclidean-ring-cancel fls* ⇒
 nat ⇒ 'a *prime* ⇒ (int ⇒ 'a)
where *fls-partially-p-modulo-rep* f 0 p = (\$\$) f |
fls-partially-p-modulo-rep f (Suc N) p =
 (*fls-partially-p-modulo-rep* f N p)(
 fls-subdegree f + int N :=
 (*fls-partially-p-modulo-rep* f N p (*fls-subdegree* f + int N)) *pmod* p,
 fls-subdegree f + int N + 1 :=
 f \$\$ (*fls-subdegree* f + int N + 1) +
 (*fls-partially-p-modulo-rep* f N p (*fls-subdegree* f + int N)) *pdiv* p
)

lemma *fls-partially-p-modulo-rep-zero[simp]*:
fls-partially-p-modulo-rep 0 N p n = 0
 ⟨*proof*⟩

lemma *fls-partially-p-modulo-rep-zeros-below*:
fls-partially-p-modulo-rep f N p n = 0 **if** n < *fls-subdegree* f
 ⟨*proof*⟩

lemma *fls-partially-p-modulo-rep-func-lower-bound*:
 ∀ n < *fls-subdegree* f. *fls-partially-p-modulo-rep* f (nat (n + 1 - *fls-subdegree* f))
 p n = 0
 ⟨*proof*⟩

lemma *fls-partially-p-modulo-rep-agrees*:
fls-partially-p-modulo-rep f N p n = *fls-partially-p-modulo-rep* f M p n
if n ≤ *fls-subdegree* f + int N - 1 **and** M ≥ N
 ⟨*proof*⟩

lemma *fls-partially-p-modulo-rep-func-partially*:

$n > \text{fls-subdegree } f + \text{int } N \implies \text{fls-partially-p-modulo-rep } f \ N \ p \ n = f \ \$\$ \ n$
 ⟨proof⟩

lemma *fls-partially-p-modulo-rep-cong*:

$\text{fls-partially-p-modulo-rep } f \ N \ p \ n = \text{fls-partially-p-modulo-rep } g \ N \ p \ n$
if $\text{fls-subdegree } f = \text{fls-subdegree } g$ **and** $\forall k \leq n. f \ \$\$ \ k = g \ \$\$ \ k$
 ⟨proof⟩

lemma *fls-partially-reduced-nth-vs-rep*:

$N \leq n + 1 - \text{fls-subdegree } f \implies \text{fls-partially-p-modulo-rep } f \ N \ p = (\$\$) \ f$
if $\forall k \leq n. (f \ \$\$ \ k) \ \text{pdiv } p = 0$
 ⟨proof⟩

lemma *fls-partially-p-modulo-rep-carry*:

$(\sum_{k=\text{fls-subdegree } f..N} (f \ \$\$ \ k - \text{fls-partially-p-modulo-rep } f \ (\text{nat } (N - \text{fls-subdegree } f)) \ p \ k) * (\text{Rep-prime } p) \wedge \text{nat } (k - \text{fls-subdegree } f)) = 0$
 ⟨proof⟩

4.5.3 Full reduction of formal Laurent series

Now we collect the reduced terms into a formal Laurent series.

overloading $p\text{-modulo-fls} \equiv p\text{-modulo} :: 'a \ \text{fls} \Rightarrow 'a \ \text{prime} \Rightarrow 'a \ \text{fls}$
begin

definition $p\text{-modulo-fls} ::$

$'a :: \text{nontrivial-euclidean-ring-cancel } \text{fls} \Rightarrow 'a \ \text{prime} \Rightarrow 'a \ \text{fls}$

where

$p\text{-modulo-fls } f \ p \equiv$

$\text{Abs-fls } (\lambda n. \text{fls-partially-p-modulo-rep } f \ (\text{nat } (n + 1 - \text{fls-subdegree } f)) \ p \ n)$

end

lemma $p\text{-modulo-fls-nth}$:

$(f \ \text{pmod } p) \ \$\$ \ n = \text{fls-partially-p-modulo-rep } f \ (\text{nat } (n + 1 - \text{fls-subdegree } f)) \ p \ n$
 ⟨proof⟩

lemma $p\text{-modulo-fls-nth}'$:

$\text{int } N \geq n + 1 - \text{fls-subdegree } f \implies$
 $(f \ \text{pmod } p) \ \$\$ \ n = \text{fls-partially-p-modulo-rep } f \ N \ p \ n$
 ⟨proof⟩

context

fixes $p :: 'a :: \text{nontrivial-euclidean-ring-cancel } \text{prime}$

begin

lemma $\text{fls-pmod-reduced[simp]}$: $((f \ \text{pmod } p) \ \$\$ \ n) \ \text{pdiv } p = 0$
 ⟨proof⟩

lemma *fls-pmod-pmod-nth[simp]*: $((f \text{ pmod } p) \text{ \#\# } n) \text{ pmod } p = (f \text{ pmod } p) \text{ \#\# } n$ **for**
 $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pmod-carry*:
 $(\sum_{k=\text{fls-subdegree } f..N}. (f - (f \text{ pmod } p)) \text{ \#\# } k * (\text{Rep-prime } p) \wedge \text{nat } (k - \text{fls-subdegree } f)) =$
 $(\text{fls-partially-p-modulo-rep } f (\text{nat } (N + 1 - \text{fls-subdegree } f)) p (N + 1) - f \text{ \#\# } (N + 1)) *$
 $(\text{Rep-prime } p) \wedge \text{nat } (N + 1 - \text{fls-subdegree } f)$
 $\langle \text{proof} \rangle$

lemma *fls-pmod-carry'*:
 $(\sum_{k=\text{fls-subdegree } f..N}. (f - (f \text{ pmod } p)) \text{ \#\# } k * (\text{Rep-prime } p) \wedge \text{nat } (k - \text{fls-subdegree } f)) =$
 $(\text{fls-partially-p-modulo-rep } f (\text{nat } (N - \text{fls-subdegree } f)) p N \text{ pdiv } p) *$
 $(\text{Rep-prime } p) \wedge \text{Suc } (\text{nat } (N - \text{fls-subdegree } f))$
if $N \geq \text{fls-subdegree } f$
 $\langle \text{proof} \rangle$

lemma *fls-p-reduced-is-zero-iff*:
 $f = 0 \iff f \simeq_p 0 \wedge (\forall n. f \text{ \#\# } n \text{ pdiv } p = 0)$
for $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *p-modulo-fls-eq0[simp]*:
 $f \text{ pmod } p = 0$ **if** $f \simeq_p 0$ **for** $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pmod-eq0-iff*:
 $f \text{ pmod } p = 0 \iff f \simeq_p 0$ **for** $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pmod-equiv*: $f \simeq_p f \text{ pmod } p$ **for** $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *fls-p-reduced-subdegree-unique*:
 $\text{fls-subdegree } g = (f^{\circ p})$
if $f \simeq_p g$ **and** $\forall n. (g \text{ \#\# } n) \text{ pdiv } p = 0$
for $f g :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pmod-subdegree*:
 $\text{fls-subdegree } (f \text{ pmod } p) = (f^{\circ p})$ **for** $f :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pmod-depth*:
 $(f \text{ pmod } p)^{\circ p} = (f^{\circ p})$ **for** $f :: 'a \text{ fls}$

<proof>

lemma *fls-pmod-subdegree-ge*:

fls-subdegree (f pmod p) ≥ fls-subdegree f **if** $f \not\sim_p 0$

for $f :: 'a \text{ fls}$

<proof>

lemma *fls-pmod-nth-below-subdegree*:

$(f \text{ pmod } p) \text{ \#\# } n = 0$ **if** $n < \text{fls-subdegree } f$ **for** $f :: 'a \text{ fls}$

<proof>

lemma *fls-pmod-nth-cong*:

$(f \text{ pmod } p) \text{ \#\# } n = (g \text{ pmod } p) \text{ \#\# } n$ **if** $\forall k \leq n. f \text{ \#\# } k = g \text{ \#\# } k$ **for** $f g :: 'a \text{ fls}$

<proof>

lemma *fls-partially-reduced-nth-vs-pmod*:

$(f \text{ pmod } p) \text{ \#\# } n = f \text{ \#\# } n$ **if** $\forall k \leq n. (f \text{ \#\# } k) \text{ pdiv } p = 0$

<proof>

lemma *fls-pmod-eqI*:

$f \text{ pmod } p = g$ **if** $f \sim_p g$ **and** $\forall n. (g \text{ \#\# } n) \text{ pdiv } p = 0$

<proof>

lemma *fls-pmod-eq-pmod-iff*:

$f \text{ pmod } p = g \text{ pmod } p \iff f \sim_p g$ **for** $f g :: 'a \text{ fls}$

<proof>

lemma *fls-pmod-equiv-pmod-iff*:

$(f \text{ pmod } p) \sim_p (g \text{ pmod } p) \iff f \sim_p g$

for $f g :: 'a \text{ fls}$

<proof>

lemma *fls-p-modulo-iff*:

$f \text{ pmod } p = g \iff f \sim_p g \wedge (\forall n. (g \text{ \#\# } n) \text{ pdiv } p = 0)$

<proof>

lemma *fls-pmod-pmod[simp]*: $(f \text{ pmod } p) \text{ pmod } p = f \text{ pmod } p$ **for** $f :: 'a \text{ fls}$

<proof>

lemma *fls-pmod-cong*:

$f \text{ pmod } p = g \text{ pmod } p$ **if** $f \sim_p g$ **for** $f g :: 'a \text{ fls}$

<proof>

end

4.5.4 Algebraic properties of reduction

lemma *fls-pmod-1[simp]*:

$(1 :: 'a \text{ fls}) \text{ pmod } p = 1$ **for** $p :: 'a :: \text{nontrivial-unique-euclidean-ring prime}$

$\langle proof \rangle$

context

fixes $p :: 'a::\text{nontrivial-euclidean-ring-cancel prime}$

begin

lemma *fls-pmod-0*: $(0::'a \text{ fls}) \text{ pmod } p = 0$

$\langle proof \rangle$

lemma *fls-pmod-uminus-equiv*: $(-f) \text{ pmod } p \simeq_p -(f \text{ pmod } p)$ **for** $f :: 'a \text{ fls}$

$\langle proof \rangle$

lemma *fls-pmod-equiv-add*:

$(f + g) \text{ pmod } p = (f' + g') \text{ pmod } p$ **if** $f \simeq_p f'$ **and** $g \simeq_p g'$

for $f f' g g' :: 'a \text{ fls}$

$\langle proof \rangle$

lemma *fls-pmod-const-add*:

$(\text{fls-const } c + f) \text{ pmod } p = \text{fls-const } c + f \text{ pmod } p$ **if** $\text{fls-subdegree } f > 0$ **and** $c \text{ pdiv } p = 0$

for $c :: 'a$ **and** $f :: 'a \text{ fls}$

$\langle proof \rangle$

lemma *fls-pmod-add-equiv*:

$(f + g) \text{ pmod } p \simeq_p (f \text{ pmod } p) + (g \text{ pmod } p)$ **for** $f g :: 'a \text{ fls}$

$\langle proof \rangle$

lemma *fls-pmod-mismatched-add-nth*:

$n < \text{fls-subdegree } g \implies ((f + g) \text{ pmod } p) \text{ \#\# } n = (f \text{ pmod } p) \text{ \#\# } n$

$((f + g) \text{ pmod } p) \text{ \#\# } (\text{fls-subdegree } g) =$

$((f \text{ pmod } p) \text{ \#\# } (\text{fls-subdegree } g) + g \text{ \#\# } (\text{fls-subdegree } g)) \text{ pmod } p$

if $\text{fls-subdegree } g > \text{fls-subdegree } f$

for $f :: 'a \text{ fls}$

$\langle proof \rangle$

lemma *fls-pmod-equiv-minus*:

$(f - g) \text{ pmod } p = (f' - g') \text{ pmod } p$ **if** $f \simeq_p f'$ **and** $g \simeq_p g'$

for $f f' g g' :: 'a \text{ fls}$

$\langle proof \rangle$

lemma *fls-pmod-minus-equiv*:

$(f - g) \text{ pmod } p \simeq_p ((f \text{ pmod } p) - (g \text{ pmod } p))$ **for** $f g :: 'a \text{ fls}$

$\langle proof \rangle$

lemma *fls-pmod-equiv-times*:

$(f * g) \text{ pmod } p = (f' * g') \text{ pmod } p$ **if** $f \simeq_p f'$ **and** $g \simeq_p g'$

for $f f' g g' :: 'a \text{ fls}$

$\langle proof \rangle$

lemma *fls-pmod-times-equiv*:
 $(f * g) \text{ pmod } p \simeq_p (f \text{ pmod } p) * (g \text{ pmod } p)$ **for** $f g :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pmod-diff-cancel*:
 $(f \text{ pmod } p) \text{ \&\& } n = (g \text{ pmod } p) \text{ \&\& } n$ **if** $n < ((f - g)^{\circ p})$
for $f g :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pmod-diff-cancel-iff*:
 $((f - g)^{\circ p}) > n \iff$
 $(\forall k \leq n. (f \text{ pmod } p) \text{ \&\& } k = (g \text{ pmod } p) \text{ \&\& } k)$
if $f \not\simeq_p g$
for $f g :: 'a \text{ fls}$
 $\langle \text{proof} \rangle$

end

4.6 Inversion relative to a prime

4.6.1 Of scalars

class *bezout-semiring* = *semiring-gcd* +
assumes *bezout*: $\exists u v. u * x + v * y = \text{gcd } x y$

class *bezout-ring* = *ring-gcd* +
assumes *bezout*: $\exists u v. u * x + v * y = \text{gcd } x y$
begin
subclass *bezout-semiring* $\langle \text{proof} \rangle$
subclass *idom* $\langle \text{proof} \rangle$
end

instance *int* :: *bezout-ring* $\langle \text{proof} \rangle$

class *nontrivial-factorial-euclidean-bezout* = *nontrivial-euclidean-ring-cancel* + *bezout-semiring*

begin
subclass *bezout-ring* $\langle \text{proof} \rangle$
end

class *nontrivial-factorial-unique-euclidean-bezout* =
nontrivial-unique-euclidean-ring + *bezout-semiring*
begin
subclass *nontrivial-factorial-euclidean-bezout* $\langle \text{proof} \rangle$
end

instance *int* :: *nontrivial-factorial-unique-euclidean-bezout* $\langle \text{proof} \rangle$

lemma *prime-residue-inverse-ex1*:
 $\exists! x. x \text{ pdiv } p = 0 \wedge (x * a) \text{ pmod } p = 1$ **if** $\neg p \text{ pdiv } a$

for $a :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout}$ **and** $p :: 'a \text{ prime}$
 $\langle \text{proof} \rangle$

consts $\text{pinverse} :: 'a \Rightarrow 'b \text{ prime} \Rightarrow 'a$
 $((-^{-1}) [51, 51] 50)$

overloading $\text{pinverse-scalar} \equiv \text{pinverse} :: 'a \Rightarrow 'a \text{ prime} \Rightarrow 'a$
begin

definition $\text{pinverse-scalar} ::$
 $'a::\text{nontrivial-factorial-unique-euclidean-bezout} \Rightarrow 'a \text{ prime} \Rightarrow 'a$
where
 $\text{pinverse-scalar } a \ p \equiv$
 $(\text{if } p \text{ pdvd } a \text{ then } 0 \text{ else}$
 $(\text{THE } x. x \text{ pdiv } p = 0 \wedge (x * a) \text{ pmod } p = 1))$
end

context
fixes $p :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime}$
begin

lemma $\text{pinverse-scalar}: ((a^{-1p}) * a) \text{ pmod } p = 1$ **if** $\neg p \text{ pdvd } a$ **for** $a :: 'a$
 $\langle \text{proof} \rangle$

lemma $\text{pinverse-scalar-reduced}: (a^{-1p}) \text{ pdiv } p = 0$ **for** $a :: 'a$
 $\langle \text{proof} \rangle$

lemma $\text{pinverse-scalar-nzero}: (a^{-1p}) \neq 0$ **if** $\neg p \text{ pdvd } a$ **for** $a :: 'a$
 $\langle \text{proof} \rangle$

lemma $\text{pinverse-scalar-zero[simp]}: ((0::'a)^{-1p}) = 0$
 $\langle \text{proof} \rangle$

lemma $\text{pinverse-scalar-one[simp]}: ((1::'a)^{-1p}) = 1$
 $\langle \text{proof} \rangle$

end

4.6.2 Inductive partial inversion of formal Laurent series

Inductively define a depth-zero partial inverse representative function. It is effectively a polynomial of degree equal to the *nat* argument.

primrec $\text{fls-partially-pinverse-rep} ::$
 $'a::\text{nontrivial-factorial-unique-euclidean-bezout fls} \Rightarrow \text{nat} \Rightarrow$
 $'a \text{ prime} \Rightarrow (\text{int} \Rightarrow 'a)$
where
 $\text{fls-partially-pinverse-rep } f \ 0 \ p \ n =$
 $(\text{if } n = 0$
 $\text{then } ((f \text{ pmod } p) \ \$\$ \ (f^{op}))^{-1p}$
 $\text{else } 0)$

```

| fls-partially-pinverse-rep f (Suc N) p =
  (fls-partially-pinverse-rep f N p)(
    int (Suc N) := -(
      (
        (
          fls-shift (fop) (Abs-fls (fls-partially-pinverse-rep f N p)) *
          (f p mod p)
        ) p mod p
      ) $$ (int (Suc N)) * (((f p mod p) $$ (fop))-1p)
    ) p mod p
  )

```

lemma *fls-partially-pinverse-rep-func-lower-bound*:
 $\forall n < 0. \text{fls-partially-pinverse-rep } f \text{ (nat } n) \text{ p } n = 0$
<proof>

lemma *fls-partially-pinverse-rep-at-0*:
 $\text{fls-partially-pinverse-rep } f \text{ N p } 0 = (((f \text{ p mod } p) \text{ $$ } (f^{\circ p}))^{-1}p)$
<proof>

lemma *fls-partially-pinverse-rep-func-lower-bound'*:
 $\forall n < 0. \text{fls-partially-pinverse-rep } f \text{ N p } n = 0$
<proof>

lemmas *Abs-fls-partially-pinverse-rep-nth =*
nth-Abs-fls-lower-bound[OF fls-partially-pinverse-rep-func-lower-bound']

lemma *fls-partially-pinverse-rep-func-truncates*:
 $n > \text{int } N \implies \text{fls-partially-pinverse-rep } f \text{ N p } n = 0$
<proof>

context
fixes $p :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout-prime}$
begin

lemma *pinverse-scalar-fls-pmod-base*:
 (
 (((f p mod p) \$\$ (f^op))⁻¹p) * (f p mod p) \$\$ (f^op)
) p mod p = 1
if $f \not\approx_p 0$
for $f :: 'a \text{ fls}$
<proof>

lemma *fls-partially-pinverse-rep-eq-zero*:
 $\text{fls-partially-pinverse-rep } f \text{ N p } = 0$ **if** $f \approx_p 0$
for $f :: 'a \text{ fls}$
<proof>

lemma *fls-partially-pinverse-rep-nzero-at-0*:

fls-partially-pinverse-rep $f N p 0 \neq 0$ **if** $f \not\sim_p 0$ **for** $f :: 'a \text{ fls}$
 ⟨proof⟩

lemma *fls-partially-pinverse-rep-eq-zero-iff*:
fls-partially-pinverse-rep $f N p = 0 \iff f \sim_p 0$
for $f :: 'a \text{ fls}$
 ⟨proof⟩

lemma *Abs-fls-partially-pinverse-rep-nzero*:
Abs-fls (fls-partially-pinverse-rep $f N p) \neq 0$ **if** $f \not\sim_p 0$
for $f :: 'a \text{ fls}$
 ⟨proof⟩

lemma *Abs-fls-partially-pinverse-rep-eq-zero-iff*:
Abs-fls (fls-partially-pinverse-rep $f N p) = 0 \iff f \sim_p 0$
for $f :: 'a \text{ fls}$
 ⟨proof⟩

lemma *Abs-fls-partially-pinverse-rep-subdegree*:
fls-subdegree (Abs-fls (fls-partially-pinverse-rep $f N p)) = 0$ **if** $f \not\sim_p 0$
for $f :: 'a \text{ fls}$
 ⟨proof⟩

lemma *fls-shift-pinv-rep-subdegree*:
fls-subdegree (fls-shift $m (Abs-fls (fls-partially-pinverse-rep$ $f N p))) = -m$
if $f \not\sim_p 0$
for $f :: 'a \text{ fls}$
 ⟨proof⟩

lemma *fls-partially-pinverse-rep-cong*:
fls-partially-pinverse-rep $f N p (n - K) = \text{fls-partially-pinverse-rep}$ $g N p (n - K)$
if $f^{op} = K$ **and** $g^{op} = K$ **and** $\forall k \leq n. f \text{ $$$ } k = g \text{ $$$ } k$
for $f g :: 'a \text{ fls}$
 ⟨proof⟩

lemma *fls-pinv-rep-times-f-subdegree*:
fls-subdegree (
fls-shift $(f^{op}) (Abs-fls (fls-partially-pinverse-rep$ $f N p)) * (f \text{ pmod } p)$
 $) = 0$
if $f \not\sim_p 0$
for $f :: 'a \text{ fls}$
 ⟨proof⟩

lemma *Abs-fls-pinv-rep-Suc*:
fixes $f :: 'a \text{ fls}$ **and** $N :: \text{nat}$
defines
 $c \equiv$

```

- (
  (
    (
      fls-shift (fop) (Abs-fls (fls-partially-pinverse-rep f N p)) *
      (f p mod p)
    ) p mod p
  ) $$ (int (Suc N)) * (((f p mod p) $$ (fop))-1p)
) p mod p
assumes f: f  $\neg \simeq_p 0$ 
shows
  Abs-fls (fls-partially-pinverse-rep f (Suc N) p) =
  Abs-fls (fls-partially-pinverse-rep f N p) + fls-shift (- int (Suc N)) (fls-const
c)
<proof>

```

lemma *fls-pinv-rep-times-f-nth*:

```

((
  fls-shift (fop) (Abs-fls (fls-partially-pinverse-rep f N p)) *
  (f p mod p)
) p mod p) $$ n = 0
if f: f  $\neg \simeq_p 0$  and n: n ≥ 1 n ≤ int N
for f :: 'a fls and n :: int
<proof>

```

end

4.6.3 Full inversion of formal Laurent series

Now we collect the terms of the infinite sequence of partial inversion polynomials into a formal Laurent series, shifted to the opposite depth as the original.

overloading *pinverse-fls* \equiv *pinverse* :: 'a fls \Rightarrow 'a prime \Rightarrow 'a fls

begin

definition *pinverse-fls* ::

'a::nontrivial-factorial-unique-euclidean-bezout fls \Rightarrow 'a prime \Rightarrow 'a fls

where

```

pinverse-fls f p  $\equiv$ 
  (if f  $\simeq_p 0$  then 0 else
   fls-shift (fop)
   (Abs-fls ( $\lambda n$ . fls-partially-pinverse-rep f (nat n) p n)))

```

end

context

fixes p :: 'a::nontrivial-factorial-unique-euclidean-bezout prime

begin

lemma *pinverse-fls-nth*:

```

(f-1p) $$ n =
  fls-partially-pinverse-rep f (nat (n + (fop))) p (n + (fop))

```

if $f \not\sim_p 0$
 $\langle \text{proof} \rangle$

lemma *pinverse-fls-nth'*:
 $\text{int } N \geq n + (f^{\circ p}) \implies$
 $(f^{-1p}) \text{ \#\# } n = \text{fls-partially-pinverse-rep } f \ N \ p \ (n + (f^{\circ p}))$
if $f \not\sim_p 0$
 $\langle \text{proof} \rangle$

lemma *pinverse-fls-eq0[simp]*:
 $(f^{-1p}) = 0 \text{ if } f \simeq_p 0 \text{ for } f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pinv-eq0-iff*:
 $(f^{-1p}) = 0 \iff f \simeq_p 0 \text{ for } f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pinv-subdegree*:
 $\text{fls-subdegree } (f^{-1p}) = -(f^{\circ p}) \text{ for } f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pinv-reduced[simp]*: $((f^{-1p}) \text{ \#\# } n) \text{ pdiv } p = 0$
 $\langle \text{proof} \rangle$

lemma *pinverse-times-p-modulo-fls*:
 $((f^{-1p}) * (f \text{ pmod } p)) \text{ pmod } p = 1 \text{ if } f \not\sim_p 0$
for $f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *pinverse-fls*:
 $((f^{-1p}) * f) \text{ pmod } p = 1 \text{ if } f \not\sim_p 0 \text{ for } f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *pinverse-fls-mult-equiv1*:
 $((f^{-1p}) * f) \simeq_p 1 \text{ if } f \not\sim_p 0 \text{ for } f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *pinverse-fls-mult-equiv1'*:
 $(f * (f^{-1p})) \simeq_p 1 \text{ if } f \not\sim_p 0 \text{ for } f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *p-modulo-pinverse-fls[simp]*:
 $(f^{-1p}) \text{ pmod } p = (f^{-1p}) \text{ for } f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pinv-equiv0-iff*:
 $(f^{-1p}) \simeq_p 0 \iff f \simeq_p 0$
for $f :: 'a \ \text{fls}$
 $\langle \text{proof} \rangle$

lemma *fls-pinv-depth*:

$(f^{-1p})^{\circ p} = -(f^{\circ p})$ **for** $f :: 'a \text{ fls}$
<proof>

lemma *fls-pinv-eqI*:

$(f^{-1p}) = g$ **if** $g * f \simeq_p 1$ **and** $\forall n. (g \text{ $$$ } n) \text{ pdiv } p = 0$
for $f g :: 'a \text{ fls}$
<proof>

lemma *fls-pinv-cong*:

$(f^{-1p}) = (g^{-1p})$ **if** $f \simeq_p g$ **for** $f :: 'a \text{ fls}$
<proof>

lemma *pinverse-p-modulo-fls*:

$(f \text{ pmod } p)^{-1p} = (f^{-1p})$ **for** $f :: 'a \text{ fls}$
<proof>

lemma *fls-pinv-X-intpow*: $(\text{fls-X-intpow } n :: 'a \text{ fls})^{-1p} = \text{fls-X-intpow } (-n)$
<proof>

lemma *fls-pinv-uminus*: $(-f)^{-1p} = -(f^{-1p}) \text{ pmod } p$ **for** $f :: 'a \text{ fls}$
<proof>

end

4.6.4 Algebraic properties of inversion

context

fixes $p :: 'a :: \text{nontrivial-factorial-unique-euclidean-bezout prime}$

begin

lemma *fls-pinv0*: $(0 :: 'a \text{ fls})^{-1p} = 0$
<proof>

lemma *fls-pinv1[simp]*: $(1 :: 'a \text{ fls})^{-1p} = 1$
<proof>

lemma *fls-pinv-pinv*:

$((f^{-1p})^{-1p}) = f \text{ pmod } p$ **for** $f :: 'a \text{ fls}$
<proof>

lemma *fls-pinv-pinv-equiv*:

$((f^{-1p})^{-1p}) \simeq_p f$ **for** $f :: 'a \text{ fls}$
<proof>

lemma *fls-pinv-mult*:

$(f * g)^{-1p} = ((f^{-1p}) * (g^{-1p})) \text{ pmod } p$
for $f g :: 'a \text{ fls}$

<proof>

lemma *fls-pinv-mult-equiv*:
 $((f * g)^{-1p}) \simeq_p (f^{-1p}) * (g^{-1p})$
for $f g :: 'a \text{ fls}$
<proof>

lemma *fls-pinv-pow*:
 $(f \wedge n)^{-1p} = ((f^{-1p}) \wedge n) \text{ pmod } p$
for $f g :: 'a \text{ fls}$
<proof>

lemma *fls-pinv-pow-equiv*:
 $((f \wedge n)^{-1p}) \simeq_p (f^{-1p}) \wedge n$ **for** $f :: 'a \text{ fls}$
<proof>

lemma *fls-pinv-diff-cancel-lead-coeff*:
 $((f^{-1p}) - (g^{-1p}))^{op} > -n$
if $f : f \dashv\sim_p 0 \ f^{op} = n$
and $g : g \dashv\sim_p 0 \ g^{op} = n$
and $fg : f \dashv\sim_p g \ ((f - g)^{op}) > n$
for $f g :: 'a \text{ fls}$
<proof>

end

4.7 Topology by place relative to indexed depth for formal Laurent series

4.7.1 General pattern for constructing sequence limits

definition *fls-condition-lim* ::
 $'a :: \text{nontrivial-euclidean-ring-cancel prime} \Rightarrow (\text{nat} \Rightarrow 'a \text{ fls}) \Rightarrow$
 $(\text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool}) \Rightarrow 'a \text{ fls}$
where
 $\text{fls-condition-lim } p \ F \ P =$
 $\text{Abs-fls } (\lambda n. (F \ (\text{LEAST } K. P \ (\text{nat } n) \ K) \ \text{pmod } p) \ \text{\$\$ } n)$

lemma *fls-condition-lim-fun-ex-lower-bound*:
fixes $p :: 'a :: \text{nontrivial-euclidean-ring-cancel prime}$
and $F :: \text{nat} \Rightarrow 'a \text{ fls}$
and $P :: \text{int} \Rightarrow \text{nat} \Rightarrow \text{bool}$
defines
 $f \equiv (\lambda n. (F \ (\text{LEAST } K. P \ (\text{nat } n) \ K) \ \text{pmod } p) \ \text{\$\$ } n)$
and
 $N \equiv \min 0 \ (\text{fls-subdegree } ((F \ (\text{LEAST } K. P \ 0 \ K)) \ \text{pmod } p))$
shows $\forall n < N. f \ n = 0$
<proof>

lemma *fls-condition-lim-nth*:

fls-condition-lim $p F P$ $\$ \$ n = ((F (LEAST K. P (nat n) K)) pmod p) \$ \$ n$
 ⟨proof⟩

lemma *fls-pmod-condition-lim*: (*fls-condition-lim* $p F P$) $pmod p = fls-condition-lim$
 $p F P$
 ⟨proof⟩

4.7.2 Completeness

abbreviation *fls-p-limseq-condition* $\equiv p\text{-depth-fls.p-limseq-condition}$
abbreviation *fls-p-limseq* $\equiv p\text{-depth-fls.p-limseq}$
abbreviation *fls-p-cauchy-condition* $\equiv p\text{-depth-fls.p-cauchy-condition}$
abbreviation *fls-p-cauchy* $\equiv p\text{-depth-fls.p-cauchy}$

lemma *fls-p-cauchy-condition-pmod-uniformity*:
 $((F k) pmod p) \$ \$ m = ((F k') pmod p) \$ \$ m$
if *fls-p-cauchy-condition* $p F n K$ **and** $k \geq K$ **and** $k' \geq K$ **and** $m \leq n$
for $p :: 'a::\text{nontrivial-euclidean-ring-cancel prime}$ **and** $F :: \text{nat} \Rightarrow 'a \text{ fls}$
 ⟨proof⟩

abbreviation
fls-p-cauchy-lim $p X \equiv$
fls-condition-lim $p X (\lambda n. fls-p-cauchy-condition p X (int n))$

lemma *fls-p-cauchy-limseq*: *fls-p-limseq* $p F (fls-p-cauchy-lim p F)$ **if** *fls-p-cauchy*
 $p F$
 ⟨proof⟩

lemma *fls-p-cauchy-lim-unique*:
fls-p-cauchy-lim $p F = fls-p-cauchy-lim p G$
if *fls-p-cauchy* $p G$ **and** $\forall n \geq N. (F n) \simeq_p (G n)$
 ⟨proof⟩

4.8 Transfer to prime-indexed sequences of formal Laurent series

4.8.1 Equivalence and depth

The equivalence is by divisibility of the difference of each pair of corresponding terms by the index for those terms.

type-synonym $'a \text{ fls-pseq} = 'a \text{ prime} \Rightarrow 'a \text{ fls}$

overloading
p-equal-fls-pseq \equiv
p-equal $:: 'a::\text{nontrivial-factorial-semiring prime} \Rightarrow$
 $'a \text{ fls-pseq} \Rightarrow 'a \text{ fls-pseq} \Rightarrow \text{bool}$
p-restrict-fls-pseq \equiv
p-restrict $::$
 $'a \text{ fls-pseq} \Rightarrow ('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ fls-pseq}$

$p\text{-depth-fls-pseq} \equiv$
 $p\text{-depth} :: 'a \text{ prime} \Rightarrow 'a \text{ fls-pseq} \Rightarrow \text{int}$
 $global\text{-unfrmzr-pows-fls-pseq} \equiv$
 $global\text{-unfrmzr-pows} :: ('a \text{ prime} \Rightarrow \text{int}) \Rightarrow 'a \text{ fls-pseq}$

begin

definition $p\text{-equal-fls-pseq} ::$
 $'a::\text{nontrivial-factorial-semiring prime} \Rightarrow$
 $'a \text{ fls-pseq} \Rightarrow 'a \text{ fls-pseq} \Rightarrow \text{bool}$
where $p\text{-equal-fls-pseq-def[simp]}$: $p\text{-equal-fls-pseq } p \ X \ Y \equiv (X \ p) \simeq_p (Y \ p)$

definition $p\text{-restrict-fls-pseq} ::$
 $'a::\text{nontrivial-factorial-semiring fls-pseq} \Rightarrow$
 $('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ fls-pseq}$
where
 $p\text{-restrict-fls-pseq } X \ P \equiv (\lambda p::'a \text{ prime. if } P \ p \ \text{then } X \ p \ \text{else } 0)$

definition $p\text{-depth-fls-pseq} ::$
 $'a::\text{nontrivial-factorial-semiring prime} \Rightarrow 'a \text{ fls-pseq} \Rightarrow \text{int}$
where $p\text{-depth-fls-pseq-def[simp]}$: $p\text{-depth-fls-pseq } p \ X \equiv (X \ p)^{\circ p}$

definition $global\text{-unfrmzr-pows-fls-pseq} ::$
 $('a::\text{nontrivial-factorial-semiring prime} \Rightarrow \text{int}) \Rightarrow 'a \text{ fls-pseq}$
where $global\text{-unfrmzr-pows-fls-pseq } f \equiv (\lambda p::'a \text{ prime. fls-X-intpow } (f \ p))$

end

global-interpretation $p\text{-equality-depth-fls-pseq}$:
 $global\text{-p-equality-depth-no-zero-divisors-1}$
 $p\text{-equal} :: 'a::\text{nontrivial-factorial-idom prime} \Rightarrow$
 $'a \text{ fls-pseq} \Rightarrow 'a \text{ fls-pseq} \Rightarrow \text{bool}$
 $p\text{-restrict} ::$
 $'a \text{ fls-pseq} \Rightarrow ('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ fls-pseq}$
 $p\text{-depth} :: 'a \text{ prime} \Rightarrow 'a \text{ fls-pseq} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

overloading
 $globally\text{-p-equal-fls-pseq} \equiv$
 $globally\text{-p-equal} :: 'a::\text{nontrivial-factorial-idom fls-pseq} \Rightarrow$
 $'a \text{ fls-pseq} \Rightarrow \text{bool}$
 $p\text{-depth-set-fls-pseq} \equiv$
 $p\text{-depth-set} ::$
 $'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{int} \Rightarrow 'a \text{ fls-pseq set}$
 $global\text{-depth-set-fls-pseq} \equiv$
 $global\text{-depth-set} :: \text{int} \Rightarrow 'a \text{ fls-pseq set}$

begin

definition $globally\text{-p-equal-fls-pseq} ::$
 $'a::\text{nontrivial-factorial-idom fls-pseq} \Rightarrow 'a \text{ fls-pseq} \Rightarrow \text{bool}$

where *globally-p-equal-fls-pseq-def*[simp]:
globally-p-equal-fls-pseq = *p-equality-depth-fls-pseq.globally-p-equal*

definition *p-depth-set-fls-pseq* ::
'a::nontrivial-factorial-idom prime \Rightarrow *int* \Rightarrow *'a fls-pseq set*
where *p-depth-set-fls-pseq-def*[simp]:
p-depth-set-fls-pseq = *p-equality-depth-fls-pseq.p-depth-set*

definition *global-depth-set-fls-pseq* ::
int \Rightarrow *'a::nontrivial-factorial-idom fls-pseq set*
where *global-depth-set-fls-pseq-def*[simp]:
global-depth-set-fls-pseq = *p-equality-depth-fls-pseq.global-depth-set*

end

lemma *fls-pseq-globally-reduced*:
 $X \simeq_{\forall p} (\lambda p. (X p) \text{ pmod } p)$
for *X* :: *'a::nontrivial-euclidean-ring-cancel fls-pseq*
<proof>

lemma *global-unfrmzr-pows0-fls-pseq*:
 $(\mathfrak{p} (0 :: 'a::nontrivial-factorial-idom prime \Rightarrow int) :: 'a fls-pseq) = 1$
<proof>

context
fixes *p* :: *'a::nontrivial-factorial-idom prime*
begin

lemma *global-unfrmzr-pows-fls-pseq-nequiv0*:
 $(\mathfrak{p} f :: 'a fls-pseq) \not\simeq_p 0$ **for** *f* :: *'a prime* \Rightarrow *int*
<proof>

lemma *global-unfrmzr-pows-fls-pseq*:
 $(\mathfrak{p} f :: 'a fls-pseq)^{\circ p} = f p$ **for** *f* :: *'a prime* \Rightarrow *int*
<proof>

lemma *global-unfrmzr-pows-fls-pseq-pequiv-iff*:
 $(\mathfrak{p} f :: 'a fls-pseq) \simeq_p (\mathfrak{p} g) \iff f p = g p$
for *f g* :: *'a prime* \Rightarrow *int*
<proof>

end

lemma *global-unfrmzr-pows-prod-fls-pseq*:
 $(\mathfrak{p} f :: 'a fls-pseq) * (\mathfrak{p} g) = \mathfrak{p} (f + g)$
for *f g* :: *'a::nontrivial-factorial-idom prime* \Rightarrow *int*
<proof>

context

fixes $p :: 'a::\text{nontrivial-factorial-idom prime}$
begin

lemma *global-unfrmrzr-pows-fls-pseq-nzero*:
 $(\mathfrak{p} f :: 'a \text{ fls-pseq}) \neg \simeq_p 0$ **for** $f :: 'a \text{ prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *prod-w-global-unfrmrzr-pows-fls-pseq*:
 $(X * \mathfrak{p} f)^{\circ p} = (X^{\circ p}) + f p$ **if** $X \neg \simeq_p 0$
for $X :: 'a \text{ fls-pseq}$ **and** $f :: 'a \text{ prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *normalize-depth-fls-pseq*:
 $(X * \mathfrak{p} (\lambda p :: 'a \text{ prime. } \neg(X^{\circ p})))^{\circ p} = 0$
for $X :: 'a \text{ fls-pseq}$
 $\langle \text{proof} \rangle$

end

lemma *normalized-depth-fls-pseq-product-additive*:
 $(X * \mathfrak{p} (\lambda p :: 'a \text{ prime. } \neg(X^{\circ p}))) * (Y * \mathfrak{p} (\lambda p :: 'a \text{ prime. } \neg(Y^{\circ p}))) = ((X * Y) * \mathfrak{p} (\lambda p :: 'a \text{ prime. } \neg((X^{\circ p}) + (Y^{\circ p}))))$
for $X Y :: 'a::\text{nontrivial-factorial-idom fls-pseq}$
 $\langle \text{proof} \rangle$

context

fixes $p :: 'a::\text{nontrivial-factorial-idom prime}$
begin

lemma *normalized-depth-fls-pseq-product-equiv*:
 $(X * \mathfrak{p} (\lambda p :: 'a \text{ prime. } \neg(X^{\circ p}))) * (Y * \mathfrak{p} (\lambda p :: 'a \text{ prime. } \neg(Y^{\circ p}))) \simeq_p ((X * Y) * \mathfrak{p} (\lambda p :: 'a \text{ prime. } \neg((X * Y)^{\circ p}))$
for $X Y :: 'a \text{ fls-pseq}$
 $\langle \text{proof} \rangle$

lemma *trivial-global-unfrmrzr-pows-fls-pseq*:
 $(\mathfrak{p} f :: 'a \text{ fls-pseq}) \simeq_p 1$ **if** $f p = 0$ **for** $f :: 'a \text{ prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *prod-w-trivial-global-unfrmrzr-pows-fls-pseq*:
 $X * \mathfrak{p} f \simeq_p X$ **if** $f p = 0$
for $f :: 'a \text{ prime} \Rightarrow \text{int}$ **and** $X :: 'a \text{ fls-pseq}$
 $\langle \text{proof} \rangle$

end

lemma *pow-global-unfrmrzr-pows-fls-pseq*:

$(\mathfrak{p} f :: 'a \text{ fls-pseq}) \wedge n = (\mathfrak{p} (\lambda p :: 'a \text{ prime. int } n * f p))$
for $f :: 'a :: \text{nontrivial-factorial-idom prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *global-unfrmzr-pows-fls-pseq-inv*:
 $(\mathfrak{p} (-f) :: 'a \text{ fls-pseq}) * (\mathfrak{p} f) = 1$
for $f :: 'a :: \text{nontrivial-factorial-idom prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *global-unfrmzr-pows-fls-pseq-decomp*:
 $X = ($
 $(X * \mathfrak{p} (\lambda p :: 'a \text{ prime. } -(X^{op}))) *$
 $\mathfrak{p} (\lambda p :: 'a \text{ prime. } X^{op})$
 $)$
for $X :: 'a :: \text{nontrivial-factorial-idom fls-pseq}$
 $\langle \text{proof} \rangle$

context
fixes $p :: 'a :: \text{nontrivial-factorial-idom prime}$
begin

lemma *global-unfrmzr-ptl-fls-pseq*:
 $(\mathfrak{p} (1 :: ('a \text{ prime} \Rightarrow \text{int})) p :: 'a \text{ fls}) \simeq_p$
 $(\text{fls-const } (\text{Rep-prime } p))$
 $\langle \text{proof} \rangle$

lemma *global-unfrmzr-fls-pseq*:
 $(\mathfrak{p} (1 :: ('a \text{ prime} \Rightarrow \text{int})) :: 'a \text{ fls-pseq}) \simeq_p$
 $(\lambda p :: 'a \text{ prime. fls-const } (\text{Rep-prime } p))$
 $\langle \text{proof} \rangle$

end

lemma *normalize-depth-fls-pseqE*:
fixes $X :: 'a :: \text{nontrivial-factorial-idom fls-pseq}$
obtains $X-0$
where $\forall p :: 'a \text{ prime. } X-0^{op} = 0$
and $X = X-0 * \mathfrak{p} (\lambda p :: 'a \text{ prime. } X^{op})$
 $\langle \text{proof} \rangle$

4.8.2 Inversion

abbreviation *global-pinverse-fls-pseq* ::
 $'a :: \text{nontrivial-factorial-comm-ring fls-pseq} \Rightarrow 'a \text{ fls-pseq}$
 $((^{-1\forall p} [51] 50))$
where $\text{global-pinverse-fls-pseq } X \equiv (\lambda p. (X p)^{-1p})$

context
fixes $X :: 'a :: \text{nontrivial-factorial-unique-euclidean-bezout fls-pseq}$

begin

lemma *global-pinverse-fls-pseq-eq0 [simp]*:
 $(X^{-1\forall p}) = 0$ **if** $X \simeq_{\forall p} 0$
<proof>

lemma *global-pinverse-fls-pseq-eq0-iff*:
 $(X^{-1\forall p}) = 0 \longleftrightarrow$
 $X \simeq_{\forall p} 0$
<proof>

lemma *global-pinverse-fls-pseq-equiv0-iff*:
 $(X^{-1\forall p}) \simeq_{\forall p} 0 \longleftrightarrow$
 $X \simeq_{\forall p} 0$
<proof>

lemma *global-left-pinverse-fls-pseq*:
 $(X^{-1\forall p}) * X \simeq_{\forall p}$
 $(\lambda p. \text{of-bool } (X \neg\simeq_p 0))$
<proof>

lemma *global-right-pinverse-fls-pseq*:
 $X * (X^{-1\forall p}) \simeq_{\forall p}$
 $(\lambda p. \text{of-bool } (X \neg\simeq_p 0))$
<proof>

lemma *global-left-pinverse-fls-pseq'*:
 $(X^{-1\forall p}) * X \simeq_p 1$ **if** $X \neg\simeq_p 0$
for $p :: 'a \text{ prime}$
<proof>

lemma *global-right-pinverse-fls-pseq'*:
 $X * (X^{-1\forall p}) \simeq_p 1$ **if** $X \neg\simeq_p 0$
for $p :: 'a \text{ prime}$
<proof>

lemma *global-pinverse-pinverse-fls-pseq*:
 $((X^{-1\forall p})^{-1\forall p})$
 $\simeq_{\forall p} X$
<proof>

lemma *global-pinverse-mult-fls-pseq*:
 $((X * Y)^{-1\forall p}) \simeq_{\forall p}$
 $(X^{-1\forall p}) * (Y^{-1\forall p})$
<proof>

lemma *global-pinverse-pow-fls-pseq*:
 $((X \hat{=} n)^{-1\forall p}) \simeq_{\forall p}$
 $(X^{-1\forall p}) \hat{=} n$

<proof>

lemma *global-pinverse-uminus-fls-pseq:*

$((-X)^{-1\forall p}) \simeq_{\forall p}$
 $- (X^{-1\forall p})$

<proof>

end

lemma *globally-pinverse-pcong:*

$(X^{-1\forall p}) \simeq_p (Y^{-1\forall p})$

if $X \simeq_p Y$

for $p :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime}$ **and** $X Y :: 'a$
fls-pseq

<proof>

lemma *globally-pinverse-cong:*

$(X^{-1\forall p}) \simeq_{\forall p}$
 $(Y^{-1\forall p})$

if $X \simeq_{\forall p} Y$

for $X Y :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout fls-pseq}$

<proof>

lemma *globally-pinverse-global-unfrmzr-pows:*

$(\mathbf{p} f :: 'a \text{ fls-pseq})^{-1\forall p} = \mathbf{p} (- f)$

for $f :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow \text{int}$

<proof>

lemma *global-pinverse-fls-pseq0:*

$($
 $(0::'a::\text{nontrivial-factorial-unique-euclidean-bezout fls-pseq})^{-1\forall p}$
 $) = 0$

<proof>

lemma *global-pinverse-fls-pseq1:*

$($
 $(1::'a::\text{nontrivial-factorial-unique-euclidean-bezout fls-pseq})^{-1\forall p}$
 $) = 1$

<proof>

lemma *global-pinverse-diff-cancel-lead-coeff:*

$(($
 $(X^{-1\forall p}) -$
 $(Y^{-1\forall p})$

$)^{op}) > -n$

if $X \not\simeq_p 0$ **and** $X^{op} = n$

and $Y \not\simeq_p 0$ **and** $Y^{op} = n$

and $X \not\simeq_p Y$ **and** $((X - Y)^{op}) > n$

for $p :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime}$

and $X Y :: 'a \text{ fls-pseq}$
 $\langle \text{proof} \rangle$

4.8.3 Topology via global bounded depth

abbreviation $\text{fls-pseq-bdd-global-depth} \equiv p\text{-equality-depth-fls-pseq.bdd-global-depth}$

abbreviation $\text{fls-pseq-bdd-global-dist} \equiv p\text{-equality-depth-fls-pseq.bdd-global-dist}$

abbreviation $\text{fls-pseq-globally-cauchy} \equiv p\text{-equality-depth-fls-pseq.globally-cauchy}$

abbreviation

$\text{fls-pseq-global-cauchy-condition} \equiv p\text{-equality-depth-fls-pseq.global-cauchy-condition}$

lemma $\text{fls-pseq-global-cauchy-condition-pmod-uniformity}$:

$((X k p) \text{ pmod } p) \$\$ m = ((X k' p) \text{ pmod } p) \$\$ m$

if $\text{fls-pseq-global-cauchy-condition } X n K$ **and** $k \geq K$ **and** $k' \geq K$ **and** $m \leq \text{int } n$

for $p :: 'a :: \text{nontrivial-euclidean-ring-cancel prime}$ **and** $X :: \text{nat} \Rightarrow 'a \text{ fls-pseq}$
 $\langle \text{proof} \rangle$

abbreviation

$\text{fls-pseq-cauchy-lim } X \equiv$

$\lambda p. \text{fls-condition-lim } p (\lambda n. X n p) (\text{fls-pseq-global-cauchy-condition } X)$

lemma $\text{fls-pseq-cauchy-condition-lim}$:

$\forall_F k$ **in sequentially.** $\text{fls-pseq-bdd-global-dist } (X k) (\text{fls-pseq-cauchy-lim } X) < e$

if $p \text{ os: } e > 0$ **and** $\text{cauchy: fls-pseq-globally-cauchy } X$
 $\langle \text{proof} \rangle$

end

theory $p\text{Adic-Product}$

imports

$\text{FLS-Prime-Equiv-Depth}$

$\text{HOL-Computational-Algebra.Fraction-Field}$

$\text{HOL-Analysis.Elementary-Metric-Spaces}$

begin

5 p-Adic fields as equivalence classes of sequences of formal Laurent series

5.1 Preliminaries

lemma $\text{inj-on-vimage-image-eq}$:

$(f - ' (f ' B)) \cap A = B$ **if** $\text{inj-on } f A$ **and** $B \subseteq A$

$\langle \text{proof} \rangle$

5.2 The type definition as a quotient

quotient-type (overloaded) 'a p-adic-prod
 = 'a::nontrivial-factorial-idom fls-pseq / globally-p-equal
 ⟨proof⟩

lemmas rep-p-adic-prod-inverse = Quotient3-abs-rep[OF Quotient3-p-adic-prod]
 and p-adic-prod-lift-globally-p-equal = Quotient3-rel-abs[OF Quotient3-p-adic-prod]
 and globally-p-equal-fls-pseq-equals-rsp = equals-rsp[OF Quotient3-p-adic-prod]
 and p-adic-prod-eq-iff
 = Quotient3-rel-rep[OF Quotient3-p-adic-prod, symmetric]
 and globally-p-equal-fls-pseq-rep-p-adic-prod-refl
 = Quotient3-rep-reflp[OF Quotient3-p-adic-prod]
 and globally-p-equal-fls-pseq-rep-abs-p-adic-prod
 = rep-abs-rsp[OF Quotient3-p-adic-prod] rep-abs-rsp-left[OF Quotient3-p-adic-prod]

lemma p-adic-prod-globally-p-equal-self:
 (rep-p-adic-prod (abs-p-adic-prod r)) $\simeq_{\forall p}$ r
 ⟨proof⟩

lemma rep-p-adic-prod-p-equal-self: rep-p-adic-prod (abs-p-adic-prod r) \simeq_p r
 for p :: 'a::nontrivial-factorial-idom prime and r :: 'a fls-pseq
 ⟨proof⟩

lemma rep-p-adic-prod-set-inverse: abs-p-adic-prod ' (rep-p-adic-prod ' A) = A
 ⟨proof⟩

lemma p-adic-prod-cases [case-names abs-p-adic-prod, cases type: p-adic-prod]:
 C if $\bigwedge X. x = \text{abs-p-adic-prod } X \implies C$
 ⟨proof⟩

lemmas two-p-adic-prod-cases = p-adic-prod-cases[case-product p-adic-prod-cases]
 and three-p-adic-prod-cases =
 p-adic-prod-cases[case-product p-adic-prod-cases[case-product p-adic-prod-cases]]

lemma p-adic-prod-reduced-cases [case-names abs-p-adic-prod]:
 fixes x :: 'a::nontrivial-euclidean-ring-cancel p-adic-prod
 obtains X where x = abs-p-adic-prod X and $\forall p. (X p) \text{ pmod } p = X p$
 ⟨proof⟩

lemma p-adic-prod-unique-rep:
 $\exists! X. x = \text{abs-p-adic-prod } X \wedge (\forall p. (X p) \text{ pmod } p = X p)$
 for x :: 'a::nontrivial-euclidean-ring-cancel p-adic-prod
 ⟨proof⟩

abbreviation

reduced-rep-p-adic-prod x \equiv
 (THE X. x = abs-p-adic-prod X \wedge ($\forall p. (X p) \text{ pmod } p = X p$))

lemma *reduced-rep-p-adic-prod-is-rep*:

$x = \text{abs-p-adic-prod } (\text{reduced-rep-p-adic-prod } x)$
for $x :: 'a::\text{nontrivial-euclidean-ring-cancel } p\text{-adic-prod}$
 $\langle \text{proof} \rangle$

lemma *reduced-rep-p-adic-prod-is-reduced*:

$(\text{reduced-rep-p-adic-prod } x \ p) \text{ pmod } p = \text{reduced-rep-p-adic-prod } x \ p$
for $p :: 'a::\text{nontrivial-euclidean-ring-cancel prime}$ **and** $X :: 'a \text{ p-adic-prod}$
 $\langle \text{proof} \rangle$

lemma *abs-p-adic-prod-inverse*:

$\text{reduced-rep-p-adic-prod } (\text{abs-p-adic-prod } X) = X$ **if** $\forall p. (X \ p) \text{ pmod } p = X \ p$
for $X :: 'a::\text{nontrivial-euclidean-ring-cancel fls-pseq}$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-seq-cases* [*case-names abs-p-adic-prod*]:

C **if** $\bigwedge F. X = (\lambda n. \text{abs-p-adic-prod } (F \ n)) \implies C$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-seq-reduced-cases* [*case-names abs-p-adic-prod*]:

fixes $X :: \text{nat} \Rightarrow 'a::\text{nontrivial-euclidean-ring-cancel } p\text{-adic-prod}$
obtains F
where $X = (\lambda n. \text{abs-p-adic-prod } (F \ n))$
and $\forall (n::\text{nat}) (p::'a \text{ prime}). (F \ n \ p) \text{ pmod } p = F \ n \ p$
 $\langle \text{proof} \rangle$

5.3 Algebraic instantiations

The product of p-adic fields form a ring.

instantiation *p-adic-prod* :: (*nontrivial-factorial-idom*) *comm-ring-1*
begin

lift-definition *zero-p-adic-prod* :: '*a* *p-adic-prod* **is** $0::'a \text{ fls-pseq}$ $\langle \text{proof} \rangle$

lift-definition *one-p-adic-prod* :: '*a* *p-adic-prod* **is** $1::'a \text{ fls-pseq}$ $\langle \text{proof} \rangle$

lift-definition *plus-p-adic-prod* ::

$'a \text{ p-adic-prod} \Rightarrow 'a \text{ p-adic-prod} \Rightarrow 'a \text{ p-adic-prod}$
is $\lambda X \ Y. X + Y$
 $\langle \text{proof} \rangle$

lift-definition *uminus-p-adic-prod* :: '*a* *p-adic-prod* $\Rightarrow 'a \text{ p-adic-prod}$

is $\lambda X. - X$
 $\langle \text{proof} \rangle$

lift-definition *minus-p-adic-prod* ::

$'a \text{ p-adic-prod} \Rightarrow 'a \text{ p-adic-prod} \Rightarrow 'a \text{ p-adic-prod}$
is $\lambda X \ Y. X - Y$
 $\langle \text{proof} \rangle$

lift-definition *times-p-adic-prod* ::
'a p-adic-prod \Rightarrow *'a p-adic-prod* \Rightarrow *'a p-adic-prod*
is $\lambda X Y. X * Y$
 \langle *proof* \rangle

instance
 \langle *proof* \rangle

end

lemma *pow-p-adic-prod-abs-eq*:
 $(\text{abs-p-adic-prod } X) \wedge n = \text{abs-p-adic-prod } (X \wedge n)$
for $X :: 'a::\text{nontrivial-factorial-idom fls-pseq}$
 \langle *proof* \rangle

We can create inverses at the nonzero places.

instantiation *p-adic-prod* ::
 $(\text{nontrivial-factorial-unique-euclidean-bezout}) \{ \text{divide-trivial, inverse} \}$
begin

lift-definition *divide-p-adic-prod* ::
'a p-adic-prod \Rightarrow *'a p-adic-prod* \Rightarrow *'a p-adic-prod*
is $\lambda X Y. X * (Y^{-1 \nabla p})$
 \langle *proof* \rangle

lift-definition *inverse-p-adic-prod* :: *'a p-adic-prod* \Rightarrow *'a p-adic-prod*
is *global-pinverse-fls-pseq*
 \langle *proof* \rangle

instance
 \langle *proof* \rangle

end

5.4 Equivalence and depth relative to a prime

overloading

p-equal-p-adic-prod \equiv
p-equal :: *'a::nontrivial-factorial-idom prime* \Rightarrow *'a p-adic-prod* \Rightarrow
'a p-adic-prod \Rightarrow *bool*
p-restrict-p-adic-prod \equiv
p-restrict :: *'a p-adic-prod* \Rightarrow
 $('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ p-adic-prod}$
p-depth-p-adic-prod \equiv *p-depth* :: *'a prime* \Rightarrow *'a p-adic-prod* \Rightarrow *int*
global-unfrmrzr-pows-p-adic-prod \equiv
global-unfrmrzr-pows :: $('a \text{ prime} \Rightarrow \text{int}) \Rightarrow 'a \text{ p-adic-prod}$
begin

```

lift-definition p-equal-p-adic-prod ::
  'a::nontrivial-factorial-idom prime  $\Rightarrow$ 
    'a p-adic-prod  $\Rightarrow$  'a p-adic-prod  $\Rightarrow$  bool
is p-equal
  <proof>

lift-definition p-restrict-p-adic-prod ::
  'a::nontrivial-factorial-idom p-adic-prod  $\Rightarrow$ 
    ('a prime  $\Rightarrow$  bool)  $\Rightarrow$  'a p-adic-prod
is  $\lambda X P p$ . if P p then X p else 0
  <proof>

lift-definition p-depth-p-adic-prod ::
  'a::nontrivial-factorial-idom prime  $\Rightarrow$  'a p-adic-prod  $\Rightarrow$  int
is p-depth
  <proof>

lift-definition global-unfrmzr-pows-p-adic-prod ::
  ('a::nontrivial-factorial-idom prime  $\Rightarrow$  int)  $\Rightarrow$  'a p-adic-prod
is global-unfrmzr-pows <proof>

end

lemma p-equal-p-adic-prod-abs-eq0: (abs-p-adic-prod X  $\simeq_p$  0) = (X  $\simeq_p$  0)
for p :: 'a::nontrivial-factorial-idom prime and X :: 'a fls-pseq
  <proof>

lemma p-equal-p-adic-prod-abs-eq1: (abs-p-adic-prod X  $\simeq_p$  1) = (X  $\simeq_p$  1)
for p :: 'a::nontrivial-factorial-idom prime and X :: 'a fls-pseq
  <proof>

global-interpretation p-equality-p-adic-prod:
  p-equality-no-zero-divisors-1
  p-equal :: 'a::nontrivial-factorial-idom prime  $\Rightarrow$ 
    'a p-adic-prod  $\Rightarrow$  'a p-adic-prod  $\Rightarrow$  bool
  <proof>

overloading
  globally-p-equal-p-adic-prod  $\equiv$ 
  globally-p-equal ::
    'a::nontrivial-factorial-idom p-adic-prod  $\Rightarrow$  'a p-adic-prod  $\Rightarrow$  bool
begin

definition globally-p-equal-p-adic-prod ::
  'a::nontrivial-factorial-idom p-adic-prod  $\Rightarrow$  'a p-adic-prod  $\Rightarrow$  bool
where globally-p-equal-p-adic-prod-def[simp]:
  globally-p-equal-p-adic-prod = p-equality-p-adic-prod.globally-p-equal

end

```

global-interpretation *global-p-depth-p-adic-prod*:

global-p-equal-depth-no-zero-divisors-w-inj-index-consts
p-equal :: 'a::nontrivial-factorial-idom prime \Rightarrow
'a p-adic-prod \Rightarrow 'a p-adic-prod \Rightarrow bool
p-restrict ::
'a p-adic-prod \Rightarrow ('a prime \Rightarrow bool) \Rightarrow 'a p-adic-prod
p-depth :: 'a prime \Rightarrow 'a p-adic-prod \Rightarrow int
 $\lambda f. \text{abs-p-adic-prod } (\lambda p. \text{fls-const } (f p))$
Rep-prime
⟨proof⟩

lemma *p-depth-p-adic-prod-diff-abs-eq*:
 $((\text{abs-p-adic-prod } X - \text{abs-p-adic-prod } Y)^{\circ p}) = ((X - Y)^{\circ p})$
for *p* :: 'a::nontrivial-factorial-idom prime **and** *X Y* :: 'a fls-pseq
⟨proof⟩

lemma *p-depth-p-adic-prod-diff-abs-eq'*:
 $((\text{abs-p-adic-prod } X - \text{abs-p-adic-prod } Y)^{\circ p}) = ((X p - Y p)^{\circ p})$
for *p* :: 'a::nontrivial-factorial-idom prime **and** *X Y* :: 'a fls-pseq
⟨proof⟩

overloading
p-depth-set-p-adic-prod \equiv
p-depth-set ::
'a::nontrivial-factorial-idom prime \Rightarrow int \Rightarrow 'a p-adic-prod set
global-depth-set-p-adic-prod \equiv
global-depth-set :: int \Rightarrow 'a p-adic-prod set
begin

definition *p-depth-set-p-adic-prod* ::
'a::nontrivial-factorial-idom prime \Rightarrow int \Rightarrow 'a p-adic-prod set
where *p-depth-set-p-adic-prod-def*[simp]:
p-depth-set-p-adic-prod = *global-p-depth-p-adic-prod.p-depth-set*

definition *global-depth-set-p-adic-prod* ::
int \Rightarrow 'a::nontrivial-factorial-idom p-adic-prod set
where *global-depth-set-p-adic-prod-def*[simp]:
global-depth-set-p-adic-prod = *global-p-depth-p-adic-prod.global-depth-set*

end

lemma *p-depth-set-p-adic-prod-eq-projection*:
 $((\mathcal{P}_{@p}^n) :: 'a \text{ p-adic-prod set}) =$
 $\text{abs-p-adic-prod } ' (\mathcal{P}_{@p}^n)$
for *p* :: 'a::nontrivial-factorial-idom prime
⟨proof⟩

lemma *global-depth-set-p-adic-prod-eq-projection*:

$((\mathcal{P}_{\forall p}^n) :: 'a \text{ p-adic-prod set}) =$
 $\text{abs-p-adic-prod } (\mathcal{P}_{\forall p}^n)$
for $p :: 'a :: \text{nontrivial-factorial-idom prime}$
 $\langle \text{proof} \rangle$

context

fixes $x :: 'a :: \text{nontrivial-factorial-idom p-adic-prod}$
begin

lemma $p\text{-adic-prod-p-restrict-depth}$:
 $(x \text{ prestrict } P)^{\circ p} = (\text{if } P \text{ then } x^{\circ p} \text{ else } 0)$
for $P :: 'a \text{ prime} \Rightarrow \text{bool}$
 $\langle \text{proof} \rangle$

lemma $p\text{-adic-prod-global-depth-setI}$:
 $x \in \mathcal{P}_{\forall p}^n$
if $\bigwedge p :: 'a \text{ prime. } x \not\sim_p 0 \implies (x^{\circ p}) \geq n$
 $\langle \text{proof} \rangle$

lemma $p\text{-adic-prod-global-depth-setI'}$:
 $x \in \mathcal{P}_{\forall p}^n$
if $\bigwedge p :: 'a \text{ prime. } (x^{\circ p}) \geq n$
 $\langle \text{proof} \rangle$

lemma $p\text{-adic-prod-global-depth-set-p-restrictI}$:
 $p\text{-restrict } x \in \mathcal{P}_{\forall p}^n$
if $\bigwedge p :: 'a \text{ prime. } P \text{ p} \implies x \in \mathcal{P}_{@p}^n$
 $\langle \text{proof} \rangle$

lemma $p\text{-adic-prod-p-depth-setI}$:
 $x \in \mathcal{P}_{@p}^n$
if $x \not\sim_p 0 \implies (x^{\circ p}) \geq n$
for $p :: 'a \text{ prime}$
 $\langle \text{proof} \rangle$

end

context

fixes $p :: 'a :: \text{nontrivial-euclidean-ring-cancel prime}$
begin

lemma $p\text{-adic-prod-diff-depth-gt-equiv-trans}$:
 $((x - z)^{\circ p}) > n$
if $xy: x \simeq_p y$
and $yz: y \not\sim_p z ((y - z)^{\circ p}) > n$
for $x y z :: 'a \text{ p-adic-prod}$
 $\langle \text{proof} \rangle$

lemma $p\text{-adic-prod-diff-depth-gt0-equiv-trans}$:

$((x - z)^{op}) > n$
if $n \geq 0$ **and** $x \simeq_p y$ **and** $((y - z)^{op}) > n$
for $x y z :: 'a$ *p-adic-prod*
 $\langle proof \rangle$

lemma *p-adic-prod-diff-depth-gt-trans:*

$((x - z)^{op}) > n$
if $xy: x \not\simeq_p y$ $((x - y)^{op}) > n$
and $yz: y \not\simeq_p z$ $((y - z)^{op}) > n$
and $xz: x \not\simeq_p z$
for $x y z :: 'a$ *p-adic-prod*
 $\langle proof \rangle$

lemma *p-adic-prod-diff-depth-gt0-trans:*

$((x - z)^{op}) > n$
if $n \geq 0$
and $((x - y)^{op}) > n$
and $((y - z)^{op}) > n$
and $x \not\simeq_p z$
for $x y z :: 'a$ *p-adic-prod*
 $\langle proof \rangle$

end

lemma *p-adic-prod-diff-cancel-lead-coeff:*

$((inverse\ x - inverse\ y)^{op}) > -n$
if $x : x \not\simeq_p 0$ $x^{op} = n$
and $y : y \not\simeq_p 0$ $y^{op} = n$
and $xy: x \not\simeq_p y$ $((x - y)^{op}) > n$
for $p :: 'a::nontrivial-factorial-unique-euclidean-bezout\ prime$
and $x y :: 'a$ *p-adic-prod*
 $\langle proof \rangle$

lemma *global-unfrmzr-pows-p-adic-prod0:*

$(p\ (0 :: 'a::nontrivial-factorial-idom\ prime \Rightarrow int) :: 'a\ p-adic-prod) = 1$
 $\langle proof \rangle$

context

fixes $p :: 'a::nontrivial-factorial-idom\ prime$

begin

lemma *global-unfrmzr-pows-p-adic-prod-nequiv0:*

$(p\ f :: 'a\ p-adic-prod) \not\simeq_p 0$ **for** $f :: 'a\ prime \Rightarrow int$
 $\langle proof \rangle$

lemma *global-unfrmzr-pows-p-adic-prod:*

$(p\ f :: 'a\ p-adic-prod)^{op} = f\ p$ **for** $f :: 'a\ prime \Rightarrow int$
 $\langle proof \rangle$

lemma *global-unfrmrz-pows-p-adic-prod-depth-set:*

fixes $f :: 'a \text{ prime} \Rightarrow \text{int}$

defines $n \equiv f \ p$

shows $(\mathfrak{p} \ f :: 'a \ p\text{-adic-prod}) \in \mathcal{P}_{\otimes_p^n}$

$\langle \text{proof} \rangle$

lemma *global-unfrmrz-pows-p-adic-prod-pequiv-iff:*

$((\mathfrak{p} \ f :: 'a \ p\text{-adic-prod}) \simeq_p (\mathfrak{p} \ g)) = (f \ p = g \ p)$

for $f \ g :: 'a \text{ prime} \Rightarrow \text{int}$

$\langle \text{proof} \rangle$

end

lemma *global-unfrmrz-pows-p-adic-prod-global-depth-set:*

$(\mathfrak{p} \ f :: 'a \ p\text{-adic-prod}) \in \mathcal{P}_{\forall_p^n}$

if $\forall p. f \ p \geq n$ **for** $f :: 'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{int}$

$\langle \text{proof} \rangle$

lemma *global-unfrmrz-pows-p-adic-prod-global-depth-set-0:*

$(\mathfrak{p} \ (\text{int} \circ f) :: 'a \ p\text{-adic-prod}) \in \mathcal{O}_{\forall_p}$

for $f :: 'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{nat}$

$\langle \text{proof} \rangle$

lemma *global-unfrmrz-pows-prod-p-adic-prod:*

$(\mathfrak{p} \ f :: 'a \ p\text{-adic-prod}) * (\mathfrak{p} \ g) = \mathfrak{p} \ (f + g)$

for $f \ g :: 'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{int}$

$\langle \text{proof} \rangle$

context

fixes $p :: 'a::\text{nontrivial-factorial-idom prime}$

begin

lemma *global-unfrmrz-pows-p-adic-prod-nzero:*

$(\mathfrak{p} \ f :: 'a \ p\text{-adic-prod}) \not\simeq_p 0$ **for** $f :: 'a \text{ prime} \Rightarrow \text{int}$

$\langle \text{proof} \rangle$

lemma *prod-w-global-unfrmrz-pows-p-adic-prod:*

$x \not\simeq_p 0 \Longrightarrow$

$(x * \mathfrak{p} \ f)^{\circ p} = (x^{\circ p}) + f \ p$

for $f :: 'a \text{ prime} \Rightarrow \text{int}$ **and** $x :: 'a \ p\text{-adic-prod}$

$\langle \text{proof} \rangle$

lemma *p-adic-prod-normalize-depth:*

$(x * \mathfrak{p} \ (\lambda p::'a \text{ prime}. -(x^{\circ p})))^{\circ p} = 0$

for $x :: 'a \ p\text{-adic-prod}$

$\langle \text{proof} \rangle$

end

lemma *p-adic-prod-normalized-depth-product-equiv:*

$(x * \mathfrak{p} (\lambda p :: 'a \text{ prime. } -(x^{op}))) *$
 $(y * \mathfrak{p} (\lambda p :: 'a \text{ prime. } -(y^{op}))) =$
 $((x * y) * \mathfrak{p} (\lambda p :: 'a \text{ prime. } -((x * y)^{op}))$
for $x y :: 'a :: \text{nontrivial-factorial-idom } p\text{-adic-prod}$
 $\langle \text{proof} \rangle$

context

fixes $p :: 'a :: \text{nontrivial-factorial-idom prime}$
begin

lemma *trivial-global-unfrmzr-pows-p-adic-prod:*

$f p = 0 \implies (\mathfrak{p} f :: 'a \text{ p-adic-prod}) \simeq_p 1$
for $f :: 'a \text{ prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *prod-w-trivial-global-unfrmzr-pows-p-adic-prod:*

$f p = 0 \implies x * \mathfrak{p} f \simeq_p x$
for $f :: 'a \text{ prime} \Rightarrow \text{int}$ **and** $x :: 'a \text{ p-adic-prod}$
 $\langle \text{proof} \rangle$

end

lemma *global-unfrmzr-pows-p-adic-prod-pow:*

$(\mathfrak{p} f :: 'a \text{ p-adic-prod})^{\wedge} n = (\mathfrak{p} (\lambda p. \text{int } n * f p))$
for $f :: 'a :: \text{nontrivial-factorial-idom prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *global-unfrmzr-pows-p-adic-prod-inv:*

$(\mathfrak{p} (-f) :: 'a \text{ p-adic-prod}) * (\mathfrak{p} f) = 1$
for $f :: 'a :: \text{nontrivial-factorial-idom prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *global-unfrmzr-pows-p-adic-prod-inverse:*

inverse $(\mathfrak{p} f :: 'a \text{ p-adic-prod}) = \mathfrak{p} (-f)$
for $f :: 'a :: \text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow \text{int}$
 $\langle \text{proof} \rangle$

lemma *global-unfrmzr-pows-p-adic-prod-decomp:*

$x = ($
 $(x * \mathfrak{p} (\lambda p :: 'a \text{ prime. } -(x^{op}))) *$
 $\mathfrak{p} (\lambda p :: 'a \text{ prime. } x^{op})$
 $)$
for $p :: 'a :: \text{nontrivial-factorial-idom prime}$ **and** $x :: 'a \text{ p-adic-prod}$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-normalize-depthE:*

fixes $x :: 'a :: \text{nontrivial-factorial-idom } p\text{-adic-prod}$
obtains $x-0$
where $\forall p :: 'a \text{ prime. } x-0^{op} = 0$

and $x = x-0 * p$ ($\lambda p :: 'a \text{ prime. } x^{op}$)
 ⟨proof⟩

global-interpretation *p-adic-prod-div-inv*:

global-p-equal-depth-div-inv
p-equal :: $'a :: \text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow$
 $'a \text{ p-adic-prod} \Rightarrow 'a \text{ p-adic-prod} \Rightarrow \text{bool}$
p-depth :: $'a \text{ prime} \Rightarrow 'a \text{ p-adic-prod} \Rightarrow \text{int}$
p-restrict ::
 $'a \text{ p-adic-prod} \Rightarrow ('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ p-adic-prod}$
 ⟨proof⟩

5.5 Embeddings of the base ring, *nat*, *int*, and *rat*

5.5.1 Embedding of the base ring

abbreviation *p-adic-prod-consts* \equiv *global-p-depth-p-adic-prod.consts*

abbreviation *p-adic-prod-const* \equiv *global-p-depth-p-adic-prod.const*

lemma *p-depth-p-adic-prod-consts*:

$(p\text{-adic-prod-consts } f)^{op} = ((f \text{ } p)^{op})$

for $p :: 'a :: \text{nontrivial-factorial-idom prime}$ **and** $f :: 'a \text{ prime} \Rightarrow 'a$
 ⟨proof⟩

lemmas *p-depth-p-adic-prod-const* = *p-depth-p-adic-prod-consts*[*of* - $\lambda p. -$]

lemma *p-adic-prod-global-unfrmrzr*:

$(p (1 :: 'a :: \text{nontrivial-factorial-idom prime} \Rightarrow \text{int}) :: 'a \text{ p-adic-prod}) =$
 $p\text{-adic-prod-consts } \text{Rep-prime}$
 ⟨proof⟩

5.5.2 Embedding of the field of fractions of the base ring

global-interpretation *p-adic-prod-embeds*:

global-p-equal-div-inv-w-inj-idom-consts
p-equal :: $'a :: \text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow$
 $'a \text{ p-adic-prod} \Rightarrow 'a \text{ p-adic-prod} \Rightarrow \text{bool}$
p-restrict ::
 $'a \text{ p-adic-prod} \Rightarrow ('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ p-adic-prod}$
 $\lambda f. \text{abs-p-adic-prod } (\lambda p. \text{fls-const } (f \text{ } p))$
 ⟨proof⟩

global-interpretation *p-adic-prod-depth-embeds*:

global-p-equal-depth-div-inv-w-inj-index-consts
p-equal :: $'a :: \text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow$
 $'a \text{ p-adic-prod} \Rightarrow 'a \text{ p-adic-prod} \Rightarrow \text{bool}$
p-depth :: $'a \text{ prime} \Rightarrow 'a \text{ p-adic-prod} \Rightarrow \text{int}$
p-restrict ::
 $'a \text{ p-adic-prod} \Rightarrow ('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ p-adic-prod}$
 $\lambda f. \text{abs-p-adic-prod } (\lambda p. \text{fls-const } (f \text{ } p))$

Rep-prime
 ⟨proof⟩

abbreviation *p-adic-prod-shift-p-depth* \equiv *p-adic-prod-depth-embeds.shift-p-depth*
abbreviation *p-adic-prod-of-fract* \equiv *p-adic-prod-embeds.const-of-fract*

lemma *p-adic-prod-of-fract-depth*:
 (*p-adic-prod-of-fract* (*Fraction-Field.Fract* *a b*) :: '*a p-adic-prod*)^{op} =
 (*a*^{op}) - (*b*^{op})
if *a* ≠ 0 **and** *b* ≠ 0
for *p* :: '*a::nontrivial-factorial-unique-euclidean-bezout prime*
and *a b* :: '*a*
 ⟨proof⟩

Product of all p-adic embeddings of the field of fractions of the base ring.

abbreviation *p-adic-Fracts-prod* ::
 '*a::nontrivial-factorial-unique-euclidean-bezout p-adic-prod set*
 ($\mathcal{Q}_{\forall p}$)
where $\mathcal{Q}_{\forall p} \equiv$ *p-adic-prod-embeds.range-const-of-fract*

5.5.3 Characteristic zero embeddings

class *nontrivial-factorial-idom-char-0* = *nontrivial-factorial-idom* + *semiring-char-0*
begin
subclass *ring-char-0* ⟨proof⟩
end

instance *int* :: *nontrivial-factorial-idom-char-0* ⟨proof⟩

class *nontrivial-factorial-unique-euclidean-bezout-char-0* =
nontrivial-factorial-unique-euclidean-bezout + *nontrivial-factorial-idom-char-0*

instance *int* :: *nontrivial-factorial-unique-euclidean-bezout-char-0* ⟨proof⟩

instance *p-adic-prod* :: (*nontrivial-factorial-idom-char-0*) *ring-char-0*
 ⟨proof⟩

global-interpretation *p-adic-prod-consts-char-0*:
global-p-equality-w-inj-consts-char-0
p-equal :: '*a::nontrivial-factorial-idom-char-0 prime* \Rightarrow
 '*a p-adic-prod* \Rightarrow '*a p-adic-prod* \Rightarrow *bool*
p-restrict ::
 '*a p-adic-prod* \Rightarrow ('*a prime* \Rightarrow *bool*) \Rightarrow '*a p-adic-prod*
 $\lambda f. \text{abs-}p\text{-adic-prod } (\lambda p. \text{fls-const } (f p))$
 ⟨proof⟩

global-interpretation *p-adic-prod-div-inv-consts-char-0*:
global-p-equal-div-inv-w-inj-consts-char-0
p-equal :: '*a::nontrivial-factorial-unique-euclidean-bezout-char-0 prime* \Rightarrow

$'a \text{ p-adic-prod} \Rightarrow 'a \text{ p-adic-prod} \Rightarrow \text{bool}$
p-restrict ::
 $'a \text{ p-adic-prod} \Rightarrow ('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ p-adic-prod}$
 $\lambda f. \text{abs-p-adic-prod} (\lambda p. \text{fls-const} (f p))$
 ⟨proof⟩

lemma *p-adic-prod-of-nat-depth*:
 $(\text{of-nat } n :: 'a \text{ p-adic-prod})^{\circ p} = ((\text{of-nat } n :: 'a)^{\circ p})$
for $p :: 'a::\text{nontrivial-factorial-idom prime}$
 ⟨proof⟩

lemma *p-adic-prod-of-int-depth*:
 $(\text{of-int } z :: 'a \text{ p-adic-prod})^{\circ p} = ((\text{of-int } z :: 'a)^{\circ p})$
for $p :: 'a::\text{nontrivial-factorial-idom prime}$
 ⟨proof⟩

abbreviation *p-adic-prod-of-rat* \equiv *p-adic-prod-div-inv-consts-char-0.const-of-rat*

lemma *p-adic-prod-of-rat-depth*:
 $(\text{p-adic-prod-of-rat} (\text{Rat.Fract } a \ b) :: 'a \text{ p-adic-prod})^{\circ p} =$
 $((\text{of-int } a :: 'a)^{\circ p}) - ((\text{of-int } b :: 'a)^{\circ p})$
if $a \neq 0$ **and** $b \neq 0$
for $p :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout-char-0 prime}$
and $a \ b :: \text{int}$
 ⟨proof⟩

5.6 Topologies on products of p-adic fields

5.6.1 By local place

Completeness

abbreviation *p-adic-prod-p-open-nbhd* \equiv *global-p-depth-p-adic-prod.p-open-nbhd*

abbreviation *p-adic-prod-p-open-nbhds* \equiv *global-p-depth-p-adic-prod.p-open-nbhds*

abbreviation

p-adic-prod-p-limseq-condition \equiv *global-p-depth-p-adic-prod.p-limseq-condition*

abbreviation

p-adic-prod-p-cauchy-condition \equiv *global-p-depth-p-adic-prod.p-cauchy-condition*

abbreviation *p-adic-prod-p-limseq* \equiv *global-p-depth-p-adic-prod.p-limseq*

abbreviation *p-adic-prod-p-cauchy* \equiv *global-p-depth-p-adic-prod.p-cauchy*

abbreviation

p-adic-prod-p-open-nbhds-limseq \equiv *global-p-depth-p-adic-prod.p-open-nbhds-LIMSEQ*

abbreviation

p-adic-prod-local-p-open-nbhds \equiv *global-p-depth-p-adic-prod.local-p-open-nbhds*

lemma *abs-p-adic-prod-seq-cases* [*case-names abs-p-adic-prod*]:

C **if** $\bigwedge F. X = (\lambda n. \text{abs-p-adic-prod} (F n)) \implies C$
 ⟨proof⟩

lemma *abs-p-adic-prod-p-limseq-condition*:

p-adic-prod-p-limseq-condition p ($\lambda n. \text{abs-p-adic-prod } (F\ n)$) ($\text{abs-p-adic-prod } X$)
 $n\ K =$
fls-p-limseq-condition p ($\lambda n. F\ n\ p$) ($X\ p$) $n\ K$
 $\langle \text{proof} \rangle$

lemma *abs-p-adic-prod-p-limseq*:
p-adic-prod-p-limseq p ($\lambda n. \text{abs-p-adic-prod } (F\ n)$) ($\text{abs-p-adic-prod } X$) =
fls-p-limseq p ($\lambda n. F\ n\ p$) ($X\ p$)
 $\langle \text{proof} \rangle$

lemma *abs-p-adic-prod-p-cauchy-condition*:
p-adic-prod-p-cauchy-condition p ($\lambda n. \text{abs-p-adic-prod } (F\ n)$) $n\ K =$
fls-p-cauchy-condition p ($\lambda n. F\ n\ p$) $n\ K$
 $\langle \text{proof} \rangle$

lemma *abs-p-adic-prod-p-cauchy*:
p-adic-prod-p-cauchy p ($\lambda n. \text{abs-p-adic-prod } (F\ n)$) = *fls-p-cauchy* p ($\lambda n. F\ n\ p$)
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-p-restrict-cauchy-lim*:
p-adic-prod-p-limseq q
($\lambda n. \text{p-restrict } (X\ n) ((=)\ p)$)
($\text{abs-p-adic-prod } (\lambda q.$
if $q = p$ then *fls-p-cauchy-lim* p ($\lambda n. \text{rep-p-adic-prod } (X\ n)\ p$) else 0
))
if *p-adic-prod-p-cauchy* $p\ X$
 $\langle \text{proof} \rangle$

global-interpretation *p-complete-p-adic-prod*:
p-complete-global-p-equal-depth
p-equal :: 'a::nontrivial-euclidean-ring-cancel prime \Rightarrow
'a *p-adic-prod* \Rightarrow 'a *p-adic-prod* \Rightarrow bool
p-restrict ::
'a *p-adic-prod* \Rightarrow ('a prime \Rightarrow bool) \Rightarrow 'a *p-adic-prod*
p-depth :: 'a prime \Rightarrow 'a *p-adic-prod* \Rightarrow int
 $\langle \text{proof} \rangle$

global-interpretation *p-complete-p-adic-prod-div-inv*:
p-complete-global-p-equal-depth-div-inv
p-equal :: 'a::nontrivial-factorial-unique-euclidean-bezout prime \Rightarrow
'a *p-adic-prod* \Rightarrow 'a *p-adic-prod* \Rightarrow bool
p-depth :: 'a prime \Rightarrow 'a *p-adic-prod* \Rightarrow int
p-restrict ::
'a *p-adic-prod* \Rightarrow ('a prime \Rightarrow bool) \Rightarrow 'a *p-adic-prod*
 $\langle \text{proof} \rangle$

lemmas *p-adic-prod-hensel* = *p-complete-p-adic-prod-div-inv.hensel*

5.6.2 By bounded global depth

The metric

instantiation $p\text{-adic-prod} :: (\text{nontrivial-factorial-idom}) \text{ metric-space}$
begin

definition $\text{dist} = \text{global-p-depth-p-adic-prod.bdd-global-dist}$
declare $\text{dist-p-adic-prod-def} [\text{simp}]$

definition $\text{uniformity} = \text{global-p-depth-p-adic-prod.bdd-global-uniformity}$
declare $\text{uniformity-p-adic-prod-def} [\text{simp}]$

definition $\text{open-p-adic-prod} :: 'a \text{ p-adic-prod set} \Rightarrow \text{bool}$
where
 $\text{open-p-adic-prod } U = (\forall x \in U.$
 $\text{eventually } (\lambda(x', y). x' = x \longrightarrow y \in U) \text{ uniformity}$
 $)$

instance
 $\langle \text{proof} \rangle$

end

abbreviation $p\text{-adic-prod-global-depth} \equiv \text{global-p-depth-p-adic-prod.bdd-global-depth}$

lemma $\text{bdd-global-depth-p-adic-prod-abs-eq}$:
 $p\text{-adic-prod-global-depth } (\text{abs-p-adic-prod } X) = \text{fls-pseq-bdd-global-depth } X$
 $\langle \text{proof} \rangle$

lemma $\text{dist-p-adic-prod-abs-eq}$:
 $\text{dist } (\text{abs-p-adic-prod } X) (\text{abs-p-adic-prod } Y) = \text{fls-pseq-bdd-global-dist } X Y$
 $\langle \text{proof} \rangle$

lemma $p\text{-adic-prod-metric-is-finer}$:
 $\text{generate-topology } p\text{-adic-prod-p-open-nbhds } U \Longrightarrow \text{open } U$
 $\langle \text{proof} \rangle$

lemma $p\text{-adic-prod-metric-is-finer-than-purely-local}$:
 $\text{generate-topology } (p\text{-adic-prod-local-p-open-nbhds } p) U \Longrightarrow$
 $\text{open } U$
for $p :: 'a :: \text{nontrivial-factorial-idom prime}$
and $U :: 'a \text{ p-adic-prod set}$
 $\langle \text{proof} \rangle$

lemma $p\text{-adic-prod-limseq-abs-eq}$:
 $(\lambda n. \text{abs-p-adic-prod } (X n)) \longrightarrow \text{abs-p-adic-prod } x$
 \longleftrightarrow
 $(\forall e > 0. \forall_F n \text{ in sequentially. fls-pseq-bdd-global-dist } (X n) x < e)$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-bdd-metric-LIMSEQ*:
 $X \longrightarrow x = \text{global-p-depth-p-adic-prod.metric-space-by-bdd-depth.LIMSEQ } X x$
for $X :: \text{nat} \Rightarrow 'a::\text{nontrivial-factorial-idom p-adic-prod}$ **and** $x :: 'a \text{ p-adic-prod}$
<proof>

lemma *p-adic-prod-global-depth-set-lim-closed*:
 $x \in \mathcal{P}_{\forall p}^n$
if $\text{lim} : X \longrightarrow x$
and $\text{depth}: \forall_F k \text{ in sequentially. } X k \in \mathcal{P}_{\forall p}^n$
for $X :: \text{nat} \Rightarrow 'a::\text{nontrivial-factorial-unique-euclidean-bezout p-adic-prod}$
and $x :: 'a \text{ p-adic-prod}$
<proof>

lemma *p-adic-prod-metric-ball-multicentre*:
 $\text{ball } y e = \text{ball } x e$ **if** $y \in \text{ball } x e$
for $x y :: 'a::\text{nontrivial-factorial-idom p-adic-prod}$
<proof>

lemma *p-adic-prod-metric-ball-at-0*:
 $\text{ball } (0::'a::\text{nontrivial-factorial-idom p-adic-prod}) (\text{inverse } (2 \wedge n)) = \text{global-depth-set}$
 $(\text{int } n)$
<proof>

lemma *p-adic-prod-metric-ball-UNIV*:
 $\text{ball } x e = \text{UNIV}$ **if** $e > 1$
for $x :: 'a::\text{nontrivial-factorial-idom p-adic-prod}$
<proof>

lemma *p-adic-prod-metric-ball-at-0-normalize*:
 $\text{ball } (0::'a::\text{nontrivial-factorial-idom p-adic-prod}) e =$
 $\text{global-depth-set } \lfloor \log 2 (\text{inverse } e) \rfloor$
if $e > 0$ **and** $e \leq 1$
<proof>

lemma *p-adic-prod-metric-ball-translate*:
 $\text{ball } x e = x +_o \text{ball } 0 e$ **for** $x :: 'a::\text{nontrivial-factorial-idom p-adic-prod}$
<proof>

lemma *p-adic-prod-left-translate-metric-continuous*:
 $\text{continuous-on UNIV } ((+) x)$ **for** $x :: 'a::\text{nontrivial-factorial-idom p-adic-prod}$
<proof>

lemma *p-adic-prod-right-translate-metric-continuous*:
 $\text{continuous-on UNIV } (\lambda z. z + x)$ **for** $x :: 'a::\text{nontrivial-factorial-idom p-adic-prod}$
<proof>

lemma *p-adic-prod-left-bdd-mult-bdd-metric-continuous*:
 $\text{continuous-on UNIV } ((*) x)$

if *bdd*:
 $\forall p::'a \text{ prime. } x \dashv\sim_p 0 \longrightarrow$
 $(x^{op}) \geq n$
for $x :: 'a::\text{nontrivial-factorial-idom } p\text{-adic-prod}$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-bdd-metric-limseq-bdd-mult*:
 $(\lambda k. w * X k) \longrightarrow (w * x)$
if *bdd*:
 $\forall p::'a \text{ prime. } w \dashv\sim_p 0 \longrightarrow$
 $(w^{op}) \geq n$
and *seq*: $X \longrightarrow x$
for $w x :: 'a::\text{nontrivial-factorial-idom } p\text{-adic-prod}$ **and** $X :: \text{nat} \Rightarrow 'a \text{ p-adic-prod}$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-nonpos-global-depth-set-open*:
 $\text{ball } x \ 1 \subseteq (\mathcal{P}_{\forall p}^n)$
if $n \leq 0$ **and** $x \in (\mathcal{P}_{\forall p}^n)$
for $x :: 'a::\text{nontrivial-factorial-idom } p\text{-adic-prod}$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-global-depth-set-open*:
 $\text{open } ((\mathcal{P}_{\forall p}^n) :: 'a::\text{nontrivial-factorial-idom } p\text{-adic-prod set})$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-ball-abs-eq*:
 $\text{abs-p-adic-prod } \{ Y. \text{fls-pseq-bdd-global-dist } X \ Y < e \} =$
 $\text{ball } (\text{abs-p-adic-prod } X) \ e$
for $x :: 'a::\text{nontrivial-factorial-idom } p\text{-adic-prod}$
 $\langle \text{proof} \rangle$

Completeness

abbreviation *p-adic-prod-globally-cauchy* $\equiv \text{global-p-depth-p-adic-prod.globally-cauchy}$

abbreviation

p-adic-prod-global-cauchy-condition $\equiv \text{global-p-depth-p-adic-prod.global-cauchy-condition}$

lemma *p-adic-prod-global-cauchy-condition-abs-eq*:
 $\text{p-adic-prod-global-cauchy-condition } (\lambda n. \text{abs-p-adic-prod } (X \ n)) =$
 $\text{fls-pseq-global-cauchy-condition } X$
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-globally-cauchy-abs-eq*:
 $\text{p-adic-prod-globally-cauchy } (\lambda n. \text{abs-p-adic-prod } (X \ n)) = \text{fls-pseq-globally-cauchy}$
 X
 $\langle \text{proof} \rangle$

lemma *p-adic-prod-globally-cauchy-vs-bdd-metric-Cauchy*:
 $\text{p-adic-prod-globally-cauchy } X = \text{Cauchy } X$
for $X :: \text{nat} \Rightarrow 'a::\text{nontrivial-factorial-idom } p\text{-adic-prod}$

$\langle proof \rangle$

abbreviation

$p\text{-adic-prod-cauchy-lim } X \equiv$
 $abs\text{-}p\text{-adic-prod } (\lambda p.$
 $fls\text{-condition-lim } p$
 $(\lambda n. reduced\text{-rep-}p\text{-adic-prod } (X \ n) \ p)$
 $(p\text{-adic-prod-global-cauchy-condition } X)$
 $)$

lemma $p\text{-adic-prod-bdd-metric-complete}'$:

$X \longrightarrow p\text{-adic-prod-cauchy-lim } X$ **if** $Cauchy\ X$
 $\langle proof \rangle$

lemma $p\text{-adic-prod-bdd-metric-complete}$:

$complete$ ($UNIV :: 'a::nontrivial-euclidean-ring-cancel\ p\text{-adic-prod}\ set$)
 $\langle proof \rangle$

5.7 Hiding implementation details

lifting-update $p\text{-adic-prod.lifting}$

lifting-forget $p\text{-adic-prod.lifting}$

5.8 The subring of adelic integers

5.8.1 Type definition as a subtype of adeles

typedef (**overloaded**) $'a\ adelic\text{-int} =$

$\mathcal{O}_{\forall p} :: 'a::nontrivial-factorial-idom\ p\text{-adic-prod}\ set$
 $\langle proof \rangle$

lemma $Abs\text{-}adelic\text{-int}\text{-inverse}'$:

$Rep\text{-}adelic\text{-int } (Abs\text{-}adelic\text{-int } x) = x$ **if** $x \in \mathcal{O}_{\forall p}$
 $\langle proof \rangle$

lemma $Abs\text{-}adelic\text{-int}\text{-cases}$ [$case\text{-names } Abs\text{-}adelic\text{-int}$, $cases\ type: adelic\text{-int}$]:

P **if**
 $\bigwedge x. z = Abs\text{-}adelic\text{-int } x \implies$
 $x \in \mathcal{O}_{\forall p} \implies P$

$\langle proof \rangle$

lemmas $two\text{-}Abs\text{-}adelic\text{-int}\text{-cases} = Abs\text{-}adelic\text{-int}\text{-cases}[case\text{-product } Abs\text{-}adelic\text{-int}\text{-cases}]$

and $three\text{-}Abs\text{-}adelic\text{-int}\text{-cases} =$

$Abs\text{-}adelic\text{-int}\text{-cases}[case\text{-product } Abs\text{-}adelic\text{-int}\text{-cases}[case\text{-product } Abs\text{-}adelic\text{-int}\text{-cases}]]$

lemma $adelic\text{-int}\text{-seq}\text{-cases}$ [$case\text{-names } Abs\text{-}adelic\text{-int}$]:

fixes $X :: nat \Rightarrow 'a::nontrivial-factorial-idom\ adelic\text{-int}$

obtains $F :: nat \Rightarrow 'a\ p\text{-adic-prod}$

where $X = (\lambda n. Abs\text{-}adelic\text{-int } (F \ n))$

and $range\ F \subseteq \mathcal{O}_{\forall p}$

<proof>

5.8.2 Algebraic instantiaions

instantiation *adelic-int* :: (*nontrivial-factorial-idom*) *comm-ring-1*
begin

definition $0 = \text{Abs-adelic-int } 0$

lemma *zero-adelic-int-rep-eq*: $\text{Rep-adelic-int } 0 = 0$
<proof>

definition $1 \equiv \text{Abs-adelic-int } 1$

lemma *one-adelic-int-rep-eq*: $\text{Rep-adelic-int } 1 = 1$
<proof>

definition $x + y = \text{Abs-adelic-int } (\text{Rep-adelic-int } x + \text{Rep-adelic-int } y)$

lemma *plus-adelic-int-rep-eq*: $\text{Rep-adelic-int } (a + b) = \text{Rep-adelic-int } a + \text{Rep-adelic-int } b$
<proof>

lemma *plus-adelic-int-abs-eq*:

$x \in \mathcal{O}_{\forall p} \implies$
 $y \in \mathcal{O}_{\forall p} \implies$
 $\text{Abs-adelic-int } x + \text{Abs-adelic-int } y = \text{Abs-adelic-int } (x + y)$
<proof>

definition $-x = \text{Abs-adelic-int } (- \text{Rep-adelic-int } x)$

lemma *uminus-adelic-int-rep-eq*: $\text{Rep-adelic-int } (- x) = - \text{Rep-adelic-int } x$
<proof>

lemma *uminus-adelic-int-abs-eq*:

$x \in \mathcal{O}_{\forall p} \implies - \text{Abs-adelic-int } x = \text{Abs-adelic-int } (- x)$
<proof>

definition *minus-adelic-int* ::

$'a \text{ adelic-int} \Rightarrow 'a \text{ adelic-int} \Rightarrow 'a \text{ adelic-int}$
where $\text{minus-adelic-int} \equiv (\lambda x y. x + - y)$

lemma *minus-adelic-int-rep-eq*: $\text{Rep-adelic-int } (x - y) = \text{Rep-adelic-int } x - \text{Rep-adelic-int } y$
<proof>

lemma *minus-adelic-int-abs-eq*:

$x \in \mathcal{O}_{\forall p} \implies$
 $y \in \mathcal{O}_{\forall p} \implies$

$Abs\text{-adelic-int } x - Abs\text{-adelic-int } y = Abs\text{-adelic-int } (x - y)$
 ⟨proof⟩

definition $x * y = Abs\text{-adelic-int } (Rep\text{-adelic-int } x * Rep\text{-adelic-int } y)$

lemma $times\text{-adelic-int-rep-eq}$: $Rep\text{-adelic-int } (x * y) = Rep\text{-adelic-int } x * Rep\text{-adelic-int } y$
 ⟨proof⟩

lemma $times\text{-adelic-int-abs-eq}$:
 $x \in \mathcal{O}_{\forall p} \implies$
 $y \in \mathcal{O}_{\forall p} \implies$
 $Abs\text{-adelic-int } x * Abs\text{-adelic-int } y = Abs\text{-adelic-int } (x * y)$
 ⟨proof⟩

instance
 ⟨proof⟩

end

lemma $pow\text{-adelic-int-rep-eq}$: $Rep\text{-adelic-int } (x \wedge n) = (Rep\text{-adelic-int } x) \wedge n$
 ⟨proof⟩

lemma $pow\text{-adelic-int-abs-eq}$:
 $(Abs\text{-adelic-int } x) \wedge n = Abs\text{-adelic-int } (x \wedge n)$ **if** $x \in \mathcal{O}_{\forall p}$
 ⟨proof⟩

5.8.3 Equivalence and depth relative to a prime

overloading

$p\text{-equal-adelic-int} \equiv$
 $p\text{-equal} :: 'a :: \text{nontrivial-factorial-idom prime} \Rightarrow$
 $'a \text{ adelic-int} \Rightarrow 'a \text{ adelic-int} \Rightarrow \text{bool}$
 $p\text{-restrict-adelic-int} \equiv$
 $p\text{-restrict} :: 'a \text{ adelic-int} \Rightarrow$
 $('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow 'a \text{ adelic-int}$
 $p\text{-depth-adelic-int} \equiv$
 $p\text{-depth} :: 'a :: \text{nontrivial-factorial-idom prime} \Rightarrow 'a \text{ adelic-int} \Rightarrow \text{int}$
 $global\text{-unfrmzr-pows-adelic-int} \equiv$
 $global\text{-unfrmzr-pows} :: ('a \text{ prime} \Rightarrow \text{nat}) \Rightarrow 'a \text{ adelic-int}$

begin

definition $p\text{-equal-adelic-int} ::$
 $'a :: \text{nontrivial-factorial-idom prime} \Rightarrow$
 $'a \text{ adelic-int} \Rightarrow 'a \text{ adelic-int} \Rightarrow \text{bool}$
where $p\text{-equal-adelic-int } p \ x \ y \equiv (Rep\text{-adelic-int } x) \simeq_p (Rep\text{-adelic-int } y)$

definition $p\text{-restrict-adelic-int} ::$
 $'a :: \text{nontrivial-factorial-idom adelic-int} \Rightarrow$

(*'a prime* \Rightarrow *bool*) \Rightarrow *'a adelic-int*
where
p-restrict-adelic-int $x P \equiv$ *Abs-adelic-int* ((*Rep-adelic-int* x) *prestrict* P)

definition *p-depth-adelic-int* ::
'a::nontrivial-factorial-idom prime \Rightarrow *'a adelic-int* \Rightarrow *int*
where *p-depth-adelic-int* $p x \equiv$ ((*Rep-adelic-int* x) ^{o^p})

definition *global-unfrmzr-pows-adelic-int* ::
'a::nontrivial-factorial-idom prime \Rightarrow *nat*) \Rightarrow *'a adelic-int*
where
global-unfrmzr-pows-adelic-int $f \equiv$ *Abs-adelic-int* (*global-unfrmzr-pows* (*int* \circ f))

end

context
fixes $p :: 'a::nontrivial-factorial-idom prime$
begin

lemma *p-equal-adelic-int-abs-eq*:
(Abs-adelic-int $x) \simeq_p$ (*Abs-adelic-int* y) \longleftrightarrow
 $x \simeq_p y$
if $x \in \mathcal{O}_{\forall p}$
and $y \in \mathcal{O}_{\forall p}$
for $x y :: 'a p\text{-adic-prod}$
<proof>

lemma *p-equal-adelic-int-abs-eq0*:
(Abs-adelic-int $x) \simeq_p 0 \longleftrightarrow x \simeq_p 0$
if $x \in \mathcal{O}_{\forall p}$ **for** $x :: 'a p\text{-adic-prod}$
<proof>

lemma *p-equal-adelic-int-rep-eq0*:
(Rep-adelic-int $x) \simeq_p 0 \longleftrightarrow x \simeq_p 0$
for $x :: 'a adelic-int$
<proof>

lemma *p-equal-adelic-int-abs-eq1*:
(Abs-adelic-int $x) \simeq_p 1 \longleftrightarrow x \simeq_p 1$
if $x \in \mathcal{O}_{\forall p}$ **for** $x :: 'a p\text{-adic-prod}$
<proof>

lemma *p-depth-adelic-int-abs-eq*:
(Abs-adelic-int x) ^{o^p} $= (x^{o^p})$
if $x \in \mathcal{O}_{\forall p}$ **for** $x :: 'a p\text{-adic-prod}$
<proof>

lemma *adelic-int-depth*: $(x^{o^p}) \geq 0$ **for** $x :: 'a adelic-int$

<proof>

end

lemma *p-restrict-adelic-int-rep-eq:*

Rep-adelic-int (x prestrict P) = (Rep-adelic-int x) prestrict P

for *x :: 'a::nontrivial-factorial-idom adelic-int*

and *P :: 'a prime \Rightarrow bool*

<proof>

lemma *p-restrict-adelic-int-abs-eq:*

(Abs-adelic-int x) prestrict P = Abs-adelic-int (x prestrict P)

if *$x \in \mathcal{O}_{\forall p}$*

for *x :: 'a::nontrivial-factorial-idom p-adic-prod* **and** *P :: 'a prime \Rightarrow bool*

<proof>

lemma *global-unfrmzr-pows-adelic-int-rep-eq:*

Rep-adelic-int (p f :: 'a adelic-int) = p (int \circ f)

for *f :: 'a::nontrivial-factorial-idom prime \Rightarrow nat*

<proof>

global-interpretation *p-equality-adelic-int:*

p-equality-no-zero-divisors-1

p-equal :: 'a::nontrivial-factorial-idom prime \Rightarrow

'a adelic-int \Rightarrow 'a adelic-int \Rightarrow bool

<proof>

overloading

globally-p-equal-adelic-int \equiv

globally-p-equal ::

'a::nontrivial-factorial-idom adelic-int \Rightarrow 'a adelic-int \Rightarrow bool

begin

definition *globally-p-equal-adelic-int ::*

'a::nontrivial-factorial-idom adelic-int \Rightarrow 'a adelic-int \Rightarrow bool

where *globally-p-equal-adelic-int-def[simp]:*

globally-p-equal-adelic-int = p-equality-adelic-int.globally-p-equal

end

global-interpretation *global-p-depth-adelic-int:*

global-p-equal-depth-no-zero-divisors-w-inj-index-consts

p-equal :: 'a::nontrivial-factorial-idom prime \Rightarrow

'a adelic-int \Rightarrow 'a adelic-int \Rightarrow bool

p-restrict ::

'a adelic-int \Rightarrow ('a prime \Rightarrow bool) \Rightarrow 'a adelic-int

p-depth :: 'a prime \Rightarrow 'a adelic-int \Rightarrow int

λf . Abs-adelic-int (p-adic-prod-consts f)

Rep-prime

<proof>

lemma *p-depth-adelic-int-diff-abs-eq*:

$$((\text{Abs-adelic-int } x - \text{Abs-adelic-int } y)^{\circ p}) = ((x - y)^{\circ p})$$

if $x \in \mathcal{O}_{\forall p}$ **and** $y \in \mathcal{O}_{\forall p}$

for $p :: 'a :: \text{nontrivial-factorial-idom prime}$ **and** $x \ y :: 'a \text{ p-adic-prod}$

<proof>

lemma *p-depth-adelic-int-diff-abs-eq'*:

$$((\text{Abs-adelic-int } (\text{abs-p-adic-prod } X) - \text{Abs-adelic-int } (\text{abs-p-adic-prod } Y))^{\circ p}) = ((X - Y)^{\circ p})$$

if $X: X \in \mathcal{O}_{\forall p}$ **and** $Y: Y \in \mathcal{O}_{\forall p}$

for $p :: 'a :: \text{nontrivial-factorial-idom prime}$ **and** $X \ Y :: 'a \text{ fls-pseq}$

<proof>

context

fixes $p :: 'a :: \text{nontrivial-euclidean-ring-cancel prime}$

begin

lemma *adelic-int-diff-depth-gt-equiv-trans*:

$$((a - c)^{\circ p}) > n$$

if $a \simeq_p b$ **and** $b \not\simeq_p c$ **and** $((b - c)^{\circ p}) > n$

for $a \ b \ c :: 'a \text{ adelic-int}$

<proof>

lemma *adelic-int-diff-depth-gt0-equiv-trans*:

$$((a - c)^{\circ p}) > n$$

if $n \geq 0$ **and** $a \simeq_p b$ **and** $((b - c)^{\circ p}) > n$

for $a \ b \ c :: 'a \text{ adelic-int}$

<proof>

lemma *adelic-int-diff-depth-gt-trans*:

$$((a - c)^{\circ p}) > n$$

if $a \not\simeq_p b$ **and** $((a - b)^{\circ p}) > n$

and $b \not\simeq_p c$ **and** $((b - c)^{\circ p}) > n$

and $a \not\simeq_p c$

for $a \ b \ c :: 'a \text{ adelic-int}$

<proof>

lemma *adelic-int-diff-depth-gt0-trans*:

$$((a - c)^{\circ p}) > n$$

if $n \geq 0$

and $((a - b)^{\circ p}) > n$

and $((b - c)^{\circ p}) > n$

and $a \not\simeq_p c$

for $a \ b \ c :: 'a \text{ adelic-int}$

<proof>

lemma *adelic-int-prestrict-zero-depth*:

$1 \leq ((x \text{ prestrict } (\lambda p :: 'a \text{ prime. } (x^{op}) = 0) - x)^{op})$
if $x \text{ prestrict } (\lambda p :: 'a \text{ prime. } (x^{op}) = 0) \dashv\sim_p x$
for $x :: 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

end

lemma *adelic-int-normalize-depth*:
 $(x * \mathfrak{p} (\lambda p :: 'a \text{ prime. } -(x^{op}))) \in \mathcal{O}_{\forall p}$
for $x :: 'a :: \text{nontrivial-factorial-idom } p\text{-adic-prod}$
 $\langle \text{proof} \rangle$

lemma *global-unfrmzr-pows0-adelic-int*:
 $(\mathfrak{p} (0 :: 'a :: \text{nontrivial-factorial-idom } \text{prime} \Rightarrow \text{nat}) :: 'a \text{ adelic-int}) = 1$
 $\langle \text{proof} \rangle$

context
fixes $p :: 'a :: \text{nontrivial-factorial-idom } \text{prime}$
begin

lemma *global-unfrmzr-pows-adelic-int*:
 $(\mathfrak{p} f :: 'a \text{ adelic-int})^{op} = \text{int } (f p)$ **for** $f :: 'a \text{ prime} \Rightarrow \text{nat}$
 $\langle \text{proof} \rangle$

lemma *global-unfrmzr-pows-adelic-int-pequiv-iff*:
 $((\mathfrak{p} f :: 'a \text{ adelic-int}) \simeq_p (\mathfrak{p} g)) = (f p = g p)$
for $f g :: 'a \text{ prime} \Rightarrow \text{nat}$
 $\langle \text{proof} \rangle$

end

lemma *global-unfrmzr-pows-prod-adelic-int*:
 $(\mathfrak{p} f :: 'a \text{ adelic-int}) * (\mathfrak{p} g) = \mathfrak{p} (f + g)$
for $f g :: 'a :: \text{nontrivial-factorial-idom } \text{prime} \Rightarrow \text{nat}$
 $\langle \text{proof} \rangle$

context
fixes $p :: 'a :: \text{nontrivial-factorial-idom } \text{prime}$
begin

lemma *global-unfrmzr-pows-adelic-int-nzero*:
 $(\mathfrak{p} f :: 'a \text{ adelic-int}) \dashv\sim_p 0$ **for** $f :: 'a \text{ prime} \Rightarrow \text{nat}$
 $\langle \text{proof} \rangle$

lemma *prod-w-global-unfrmzr-pows-adelic-int*:
 $(x * \mathfrak{p} f)^{op} = (x^{op}) + \text{int } (f p)$
if $x \dashv\sim_p 0$ **for** $f :: 'a \text{ prime} \Rightarrow \text{nat}$ **and** $x :: 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

lemma *trivial-global-unfrmrzr-pows-adelic-int*:
 $(\mathfrak{p} f :: 'a \text{ adelic-int}) \simeq_p 1$ **if** $f p = 0$ **for** $f :: 'a \text{ prime} \Rightarrow \text{nat}$
 $\langle \text{proof} \rangle$

lemma *prod-w-trivial-global-unfrmrzr-pows-adelic-int*:
 $x * \mathfrak{p} f \simeq_p x$ **if** $f p = 0$
for $f :: 'a \text{ prime} \Rightarrow \text{nat}$ **and** $x :: 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

end

lemma *pow-global-unfrmrzr-pows-adelic-int*:
 $(\mathfrak{p} f :: 'a \text{ adelic-int}) ^ n = (\mathfrak{p} ((\lambda p. n) * f))$
for $f :: 'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{nat}$
 $\langle \text{proof} \rangle$

abbreviation *adelic-int-consts* \equiv *global-p-depth-adelic-int.consts*

abbreviation *adelic-int-const* \equiv *global-p-depth-adelic-int.const*

lemma *adelic-int-p-depth-consts*:
 $(\text{adelic-int-consts } f)^{\circ p} = ((f p)^{\circ p})$
for $p :: 'a::\text{nontrivial-factorial-idom prime}$ **and** $f :: 'a \text{ prime} \Rightarrow 'a$
 $\langle \text{proof} \rangle$

lemmas *adelic-int-p-depth-const* = *adelic-int-p-depth-consts*[*of* - $\lambda p. -$]

lemma *adelic-int-global-unfrmrzr*:
 $(\mathfrak{p} (1 :: 'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{nat}) :: 'a \text{ adelic-int})$
 $= \text{adelic-int-consts } \text{Rep-prime}$
 $\langle \text{proof} \rangle$

lemma *adelic-int-normalize-depthE*:
fixes $x :: 'a::\text{nontrivial-factorial-idom adelic-int}$
obtains $x-0$
where $\forall p::'a \text{ prime. } x-0^{\circ p} = 0$
and $x = x-0 * \mathfrak{p} (\lambda p::'a \text{ prime. nat } (x^{\circ p}))$
 $\langle \text{proof} \rangle$

5.8.4 Inverses

We can create inverses at depth-zero places.

instantiation *adelic-int* ::
 $(\text{nontrivial-factorial-unique-euclidean-bezout}) \{ \text{divide-trivial, inverse} \}$
begin

definition *inverse-adelic-int* ::
 $'a \text{ adelic-int} \Rightarrow 'a \text{ adelic-int}$
where
inverse-adelic-int $x =$

```

Abs-adelic-int (
  (inverse (Rep-adelic-int x)) prestrict ( $\lambda p :: 'a \text{ prime. } x^{op} = 0$ )
)

```

definition *divide-adelic-int* ::

```

'a adelic-int  $\Rightarrow$  'a adelic-int  $\Rightarrow$  'a adelic-int
where divide-adelic-int-def[simp]: divide-adelic-int x y = x * inverse y

```

instance

\langle *proof* \rangle

end

lemma *inverse-adelic-int-abs-eq*:

```

inverse (Abs-adelic-int x) =
  Abs-adelic-int (inverse x prestrict ( $\lambda p :: 'a \text{ prime. } x^{op} = 0$ ))
if x  $\in \mathcal{O}_{\forall p}$ 
for x :: 'a :: nontrivial-factorial-unique-euclidean-bezout p-adic-prod
 $\langle$ proof $\rangle$ 

```

lemma *divide-adelic-int-abs-eq*:

```

Abs-adelic-int x / Abs-adelic-int y =
  Abs-adelic-int ((x / y) prestrict ( $\lambda p :: 'a \text{ prime. } y^{op} = 0$ ))
if x  $\in \mathcal{O}_{\forall p}$  and y  $\in \mathcal{O}_{\forall p}$ 
for x y :: 'a :: nontrivial-factorial-unique-euclidean-bezout p-adic-prod
 $\langle$ proof $\rangle$ 

```

lemma *p-depth-adelic-int-inverse-diff-abs-eq*:

```

((inverse (Abs-adelic-int x) - inverse (Abs-adelic-int y))op) =
  ((inverse x - inverse y)op)
if x  $\in \mathcal{O}_{\forall p}$  xop = 0
and y  $\in \mathcal{O}_{\forall p}$  yop = 0
for p :: 'a :: nontrivial-factorial-unique-euclidean-bezout prime and x y :: 'a p-adic-prod
 $\langle$ proof $\rangle$ 

```

context

```

fixes x :: 'a :: nontrivial-factorial-unique-euclidean-bezout adelic-int
begin

```

lemma *inverse-adelic-int-rep-eq*:

```

Rep-adelic-int (inverse x) =
  (inverse (Rep-adelic-int x)) prestrict ( $\lambda p :: 'a \text{ prime. } x^{op} = 0$ )
 $\langle$ proof $\rangle$ 

```

lemma *adelic-int-inverse-equiv0-iff*:

```

inverse x  $\simeq_p$  0  $\longleftrightarrow$ 
  (x  $\simeq_p$  0  $\vee$  (xop)  $\neq$  0)
for p :: 'a prime
 $\langle$ proof $\rangle$ 

```

lemma *adelic-int-inverse-equiv0-iff'*:

$inverse\ x \simeq_p\ 0 \iff$
 $(x \simeq_p\ 0 \vee (x^{op} \geq 1))$
for $p :: 'a\ prime$
{proof}

end

lemma *adelic-int-inverse-eq0-iff*:

$inverse\ x = 0 \iff$
 $(\forall p :: 'a\ prime.\ x^{op} = 0 \implies x \simeq_p\ 0)$
for $x :: 'a :: nontrivial-factorial-unique-euclidean-bezout\ adelic-int$
{proof}

lemma *divide-adelic-int-rep-eq*:

$Rep-adelic-int\ (x / y) =$
 $(Rep-adelic-int\ x / Rep-adelic-int\ y)\ prestrict\ (\lambda p :: 'a\ prime.\ y^{op} = 0)$
for $x\ y :: 'a :: nontrivial-factorial-unique-euclidean-bezout\ adelic-int$
{proof}

lemma *adelic-int-divide-by-pequiv0*:

$x / y \simeq_p\ 0$ **if** $y \simeq_p\ 0$
for $p :: 'a :: nontrivial-factorial-unique-euclidean-bezout\ prime$ **and** $x\ y :: 'a\ adelic-int$
{proof}

lemma *adelic-int-divide-by-pos-depth*:

$x / y \simeq_p\ 0$ **if** $(y^{op}) > 0$
for $p :: 'a :: nontrivial-factorial-unique-euclidean-bezout\ prime$ **and** $x\ y :: 'a\ adelic-int$
{proof}

lemma *adelic-int-inverse0[simp]*:

$inverse\ (0 :: 'a :: nontrivial-factorial-unique-euclidean-bezout\ adelic-int) = 0$
{proof}

lemma *adelic-int-inverse1[simp]*:

$inverse\ (1 :: 'a :: nontrivial-factorial-unique-euclidean-bezout\ adelic-int) = 1$
{proof}

lemma *adelic-int-inverse-mult*:

$inverse\ (x * y) = inverse\ x * inverse\ y$
for $x\ y :: 'a :: nontrivial-factorial-unique-euclidean-bezout\ adelic-int$
{proof}

lemma *adelic-int-inverse-pcong*:

$(inverse\ x) \simeq_p\ (inverse\ y)$ **if** $x \simeq_p\ y$
for $p :: 'a :: nontrivial-factorial-unique-euclidean-bezout\ prime$ **and** $x\ y :: 'a\ adelic-int$
{proof}

context

fixes $x :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout adelic-int}$
begin

lemma *adelic-int-inverse-uminus*: $\text{inverse } (-x) = - \text{inverse } x$
(proof)

lemma *adelic-int-inverse-inverse*:
 $\text{inverse } (\text{inverse } x) = x \text{ prestrict } (\lambda p::'a \text{ prime. } x^{\circ p} = 0)$
(proof)

lemma *adelic-int-inverse-pow*: $\text{inverse } (x \wedge n) = (\text{inverse } x) \wedge n$
(proof)

lemma *adelic-int-divide-self'*:
 $x / x \simeq_p 1$ **if** $x \not\simeq_p 0$ **and** $x^{\circ p} = 0$
for $p :: 'a \text{ prime}$
(proof)

lemma *adelic-int-global-divide-self*:
 $x / x = 1$
if $\forall p::'a \text{ prime. } x \not\simeq_p 0$
and $\forall p::'a \text{ prime. } x^{\circ p} = 0$
(proof)

lemma *adelic-int-divide-self*:
 $x / x =$
 $1 \text{ prestrict } (\lambda p::'a \text{ prime. } x \not\simeq_p 0 \wedge x^{\circ p} = 0)$
(proof)

lemma *adelic-int-p-restrict-inverse*:
 $(\text{inverse } x) \text{ prestrict } P = \text{inverse } (x \text{ prestrict } P)$ **for** $P :: 'a \text{ prime} \Rightarrow \text{bool}$
(proof)

lemma *adelic-int-inverse-depth*: $(\text{inverse } x)^{\circ p} = 0$ **for** $p :: 'a \text{ prime}$
(proof)

end

lemma *adelic-int-consts-divide-self*:
 $(\text{adelic-int-consts } f) / (\text{adelic-int-consts } f) =$
 $1 \text{ prestrict } (\lambda p::'a \text{ prime. } f p \neq 0 \wedge (f p)^{\circ p} = 0)$
for $f :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow 'a$
(proof)

lemma *adelic-int-const-divide-self*:
 $(\text{adelic-int-const } c) / (\text{adelic-int-const } c) =$
 $1 \text{ prestrict } (\lambda p::'a \text{ prime. } c^{\circ p} = 0)$
if $c \neq 0$ **for** $c :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout}$

<proof>

lemma *adelic-int-consts-divide-self'*:

$(\text{adelic-int-consts } f) / (\text{adelic-int-consts } f) \simeq_p 1$

if $f p \neq 0$ **and** $(f p)^{\circ p} = 0$

for $f :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow 'a$

<proof>

lemma *adelic-int-divide-depth*:

$(x / y)^{\circ p} = (x^{\circ p})$ **if** $x / y \not\sim_p 0$

for $x y :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout adelic-int}$ **and** $p :: 'a$
prime

<proof>

lemma *global-unfrmzr-pows-adelic-int-inverse*:

$\text{inverse } (\mathfrak{p} f :: 'a \text{ adelic-int}) = 1 \text{ prestrict } (\lambda p::'a \text{ prime. } f p = 0)$

for $f :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow \text{nat}$

<proof>

lemma *adelic-int-diff-cancel-lead-coeff*:

$((\text{inverse } x - \text{inverse } y)^{\circ p}) > 0$

if $x : x \not\sim_p 0$ $x^{\circ p} = 0$

and $y : y \not\sim_p 0$ $y^{\circ p} = 0$

and $xy : x \not\sim_p y$ $((x - y)^{\circ p}) > 0$

for $p :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime}$

and $x y :: 'a \text{ adelic-int}$

<proof>

5.8.5 The ideal of primes

overloading

$p\text{-depth-set-adelic-int} \equiv$

$p\text{-depth-set} ::$

$'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{int} \Rightarrow 'a \text{ adelic-int set}$

$\text{global-depth-set-adelic-int} \equiv$

$\text{global-depth-set} :: \text{int} \Rightarrow 'a \text{ adelic-int set}$

begin

definition $p\text{-depth-set-adelic-int} ::$

$'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{int} \Rightarrow 'a \text{ adelic-int set}$

where $p\text{-depth-set-adelic-int-def[simp]}$:

$p\text{-depth-set-adelic-int} = \text{global-p-depth-adelic-int.p-depth-set}$

definition $\text{global-depth-set-adelic-int} ::$

$\text{int} \Rightarrow 'a::\text{nontrivial-factorial-idom adelic-int set}$

where $\text{global-depth-set-adelic-int-def[simp]}$:

$\text{global-depth-set-adelic-int} = \text{global-p-depth-adelic-int.global-depth-set}$

end

context

fixes $x :: 'a::\text{nontrivial-factorial-idom adelic-int}$
begin

lemma *adelic-int-global-depth-setI*:

$x \in \mathcal{P}_{\forall p}^n$
if $\bigwedge p::'a \text{ prime. } x \dashv\sim_p 0 \implies (x^{op}) \geq n$
<proof>

lemma *adelic-int-global-depth-setI'*:

$x \in \mathcal{P}_{\forall p}^n$
if $\bigwedge p::'a \text{ prime. } (x^{op}) \geq n$
<proof>

lemma *adelic-int-global-depth-set-p-restrictI*:

p-restrict $x P \in \mathcal{P}_{\forall p}^n$
if $\bigwedge p::'a \text{ prime. } P p \implies x \in \mathcal{P}_{@p}^n$
<proof>

end

context

fixes $p :: 'a::\text{nontrivial-factorial-idom prime}$
begin

lemma *adelic-int-p-depth-setI*:

$x \in \mathcal{P}_{@p}^n$
if $x \dashv\sim_p 0 \implies (x^{op}) \geq n$
for $x :: 'a \text{ adelic-int}$
<proof>

lemma *nonpos-p-depth-set-adelic-int*:

$(\mathcal{P}_{@p}^n) = (\text{UNIV} :: 'a::\text{nontrivial-factorial-idom adelic-int set})$
if $n \leq 0$
<proof>

lemma *p-depth-set-adelic-int-eq-projection*:

$((\mathcal{P}_{@p}^n) :: 'a::\text{nontrivial-factorial-idom adelic-int set}) =$
Abs-adelic-int ' ((\mathcal{P}_{@p}^n) \cap \mathcal{O}_{\forall p})
<proof>

lemma *lift-p-depth-set-adelic-int*:

Rep-adelic-int ' ((\mathcal{P}_{@p}^n) :: 'a::\text{nontrivial-factorial-idom adelic-int set})
 $= (\mathcal{P}_{@p}^n) \cap \mathcal{O}_{\forall p}$
<proof>

end

lemma *nonpos-global-depth-set-adelic-int*:
 $(\mathcal{P}_{\forall p}^n) = (UNIV :: 'a::nontrivial-factorial-idom\ adelic-int\ set)$
if $n \leq 0$
 $\langle proof \rangle$

lemma *nonneg-global-depth-set-adelic-int-eq-projection*:
 $((\mathcal{P}_{\forall p}^n) :: 'a::nontrivial-factorial-idom\ adelic-int\ set) =$
 $Abs-adelic-int\ \text{' } \mathcal{P}_{\forall p}^n$
if $n: n \geq 0$
 $\langle proof \rangle$

lemma *lift-nonneg-global-depth-set-adelic-int*:
 $Rep-adelic-int\ \text{'}$
 $((\mathcal{P}_{\forall p}^n) :: 'a::nontrivial-factorial-idom\ adelic-int\ set)$
 $= \mathcal{P}_{\forall p}^n$
if $n: n \geq 0$
 $\langle proof \rangle$

5.8.6 Topology p-open neighbourhoods

General properties of p-open sets

abbreviation *adelic-int-p-open-nbhd* $\equiv global-p-depth-adelic-int.p-open-nbhd$

abbreviation *adelic-int-p-open-nbhds* $\equiv global-p-depth-adelic-int.p-open-nbhds$

abbreviation *adelic-int-local-p-open-nbhds* $\equiv global-p-depth-adelic-int.local-p-open-nbhds$

lemma *adelic-int-nonpos-p-open-nbhd*:
 $adelic-int-p-open-nbhd\ p\ n\ x = UNIV$ **if** $n \leq 0$
for $p :: 'a::nontrivial-factorial-idom\ prime$
 $\langle proof \rangle$

lemma *adelic-int-p-open-nbhd-bound*:
 $adelic-int-p-open-nbhd\ p\ n\ x = adelic-int-p-open-nbhd\ p\ (int\ (nat\ n))\ x$
 $\langle proof \rangle$

lemma *adelic-int-p-open-nbhd-abs-eq*:
 $adelic-int-p-open-nbhd\ p\ n\ (Abs-adelic-int\ x) =$
 $Abs-adelic-int\ \text{' } (p-adic-prod-p-open-nbhd\ p\ n\ x \cap \mathcal{O}_{\forall p})$
if $x \in \mathcal{O}_{\forall p}$
for $p :: 'a::nontrivial-factorial-idom\ prime$
 $\langle proof \rangle$

lemma *adelic-int-p-open-nbhd-rep-eq*:
 $Rep-adelic-int\ \text{' } adelic-int-p-open-nbhd\ p\ n\ x =$
 $p-adic-prod-p-open-nbhd\ p\ n\ (Rep-adelic-int\ x) \cap \mathcal{O}_{\forall p}$
for $p :: 'a::nontrivial-factorial-idom\ prime$
 $\langle proof \rangle$

lemma *adelic-int-local-depth-ring-lift-cover*:

```

fixes  $n :: \text{int}$ 
and  $p :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime}$ 
and  $A :: 'a \text{ adelic-int set set}$ 
defines
  ( $\mathcal{A}' :: 'a \text{ p-adic-prod set set}$ )  $\equiv$ 
  ( $\lambda A.$ 
    ( $\lambda x. x \text{ prestrict } ((=) p)$ ) '  $\text{Rep-adelic-int } A +$ 
     $\text{range } (\lambda x. x \text{ prestrict } ((\neq) p))$ 
  ) '  $\mathcal{A}$ 
assumes  $\text{nonneg: } n \geq 0$ 
and  $\text{cover:}$ 
  ( $\lambda x. x \text{ prestrict } ((=) p)$ ) ' ( $\mathcal{P}_{\forall p} n$ )  $\subseteq$ 
   $\bigcup \mathcal{A}$ 
shows
  ( $\lambda x. x \text{ prestrict } ((=) p)$ ) ' ( $\mathcal{P}_{\forall p} n$ )  $\subseteq$ 
   $\bigcup \mathcal{A}'$ 
<proof>

lemma  $\text{adelic-int-lift-local-p-open:}$ 
fixes  $p :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime}$ 
and  $A :: 'a \text{ adelic-int set}$ 
defines
   $A' \equiv$ 
  ( $\lambda x. x \text{ prestrict } ((=) p)$ ) '  $\text{Rep-adelic-int } A +$ 
   $\text{range } (\lambda x. x \text{ prestrict } ((\neq) p))$ 
assumes  $\text{adelic-int-p-open: generate-topology (adelic-int-local-p-open-nbhds } p) A$ 
shows  $\text{generate-topology (p-adic-prod-local-p-open-nbhds } p) A'$ 
<proof>

```

Sequences

abbreviation

$\text{adelic-int-p-limseq-condition} \equiv \text{global-p-depth-adelic-int.p-limseq-condition}$

abbreviation

$\text{adelic-int-p-cauchy-condition} \equiv \text{global-p-depth-adelic-int.p-cauchy-condition}$

abbreviation $\text{adelic-int-p-limseq} \equiv \text{global-p-depth-adelic-int.p-limseq}$

abbreviation $\text{adelic-int-p-cauchy} \equiv \text{global-p-depth-adelic-int.p-cauchy}$

abbreviation $\text{adelic-int-p-open-nbhds-limseq} \equiv \text{global-p-depth-adelic-int.p-open-nbhds-LIMSEQ}$

context

fixes $p :: 'a::\text{nontrivial-factorial-idom prime}$

and $X :: \text{nat} \Rightarrow 'a \text{ p-adic-prod}$

and $Y :: \text{nat} \Rightarrow 'a \text{ adelic-int}$

defines $Y \equiv \lambda n. \text{Abs-adelic-int } (X n)$

assumes $\text{range-X: range } X \subseteq \mathcal{O}_{\forall p}$

begin

lemma $\text{adelic-int-p-limseq-condition-abs-eq:}$

$\text{adelic-int-p-limseq-condition } p Y (\text{Abs-adelic-int } x) =$
 $\text{p-adic-prod-p-limseq-condition } p X x$

if $x \in \mathcal{O}_{\forall p}$
 ⟨*proof*⟩

lemma *adelic-int-p-limseq-abs-eq*:

adelic-int-p-limseq p Y (*Abs-adelic-int* x) = *p-adic-prod-p-limseq* p X x
if $x \in \mathcal{O}_{\forall p}$
 ⟨*proof*⟩

lemma *adelic-int-p-cauchy-condition-abs-eq*:

adelic-int-p-cauchy-condition p Y = *p-adic-prod-p-cauchy-condition* p X
 ⟨*proof*⟩

lemma *adelic-int-p-cauchy-abs-eq*: *adelic-int-p-cauchy* p Y = *p-adic-prod-p-cauchy*
 p X

⟨*proof*⟩

end

lemma *adelic-int-p-open-nbhds-limseq-abs-eq*:

adelic-int-p-open-nbhds-limseq $(\lambda n. \text{Abs-adelic-int } (X \ n))$ (*Abs-adelic-int* x) =
p-adic-prod-p-open-nbhds-limseq X x
if $\text{range } X \subseteq \mathcal{O}_{\forall p}$ **and** $x \in \mathcal{O}_{\forall p}$
 ⟨*proof*⟩

5.8.7 Topology via global depth

The metric

instantiation *adelic-int* :: (*nontrivial-factorial-idom*) *metric-space*
begin

definition *dist* = *global-p-depth-adelic-int.bdd-global-dist*

declare *dist-adelic-int-def* [*simp*]

definition *uniformity* = *global-p-depth-adelic-int.bdd-global-uniformity*

declare *uniformity-adelic-int-def* [*simp*]

definition *open-adelic-int* :: '*a* *adelic-int set* \Rightarrow *bool*

where

open-adelic-int U =
 ($\forall x \in U.$
 eventually $(\lambda(x', y). x' = x \longrightarrow y \in U)$ *uniformity*
)

instance

⟨*proof*⟩

end

abbreviation *adelic-int-global-depth* \equiv *global-p-depth-adelic-int.bdd-global-depth*

lemma *adelic-int-bdd-global-depth-abs-eq*:
 $adelic-int-global-depth (Abs-adelic-int x) = p-adic-prod-global-depth x$
if $x \in \mathcal{O}_{\forall p}$ **for** $x :: 'a::nontrivial-factorial-idom p-adic-prod$
 $\langle proof \rangle$

lemma *dist-adelic-int-abs-eq*:
 $dist (Abs-adelic-int x) (Abs-adelic-int y) = dist x y$
if $x \in \mathcal{O}_{\forall p}$ **and** $y \in \mathcal{O}_{\forall p}$
for $x y :: 'a::nontrivial-factorial-idom p-adic-prod$
 $\langle proof \rangle$

lemma *adelic-int-limseq-abs-eq*:
 $(\lambda n. Abs-adelic-int (X n)) \longrightarrow Abs-adelic-int x$
 $\longleftarrow X \longrightarrow x$
if $range X \subseteq \mathcal{O}_{\forall p}$ **and** $x \in \mathcal{O}_{\forall p}$
 $\langle proof \rangle$

lemma *adelic-int-bdd-metric-LIMSEQ*:
 $X \longrightarrow x = global-p-depth-adelic-int.metric-space-by-bdd-depth.LIMSEQ X x$
for $X :: nat \Rightarrow 'a::nontrivial-factorial-idom adelic-int$ **and** $x :: 'a adelic-int$
 $\langle proof \rangle$

lemma *adelic-int-global-depth-set-lim-closed*:
 $x \in \mathcal{P}_{\forall p}^n$
if $lim : X \longrightarrow x$
and $depth: \forall_F k \text{ in sequentially. } X k \in \mathcal{P}_{\forall p}^n$
for $X :: nat \Rightarrow 'a::nontrivial-factorial-unique-euclidean-bezout adelic-int$
and $x :: 'a adelic-int$
 $\langle proof \rangle$

lemma *adelic-int-metric-ball-multicentre*:
 $ball y e = ball x e$ **if** $y \in ball x e$ **for** $x y :: 'a::nontrivial-factorial-idom adelic-int$
 $\langle proof \rangle$

lemma *adelic-int-metric-ball-at-0*:
 $ball (0::'a::nontrivial-factorial-idom adelic-int) (inverse (2 ^ n)) = global-depth-set (int n)$
 $\langle proof \rangle$

lemma *adelic-int-metric-ball-at-0-normalize*:
 $ball (0::'a::nontrivial-factorial-idom adelic-int) e =$
 $global-depth-set \lfloor \log 2 (inverse e) \rfloor$
if $e > 0$ **and** $e \leq 1$
 $\langle proof \rangle$

lemma *adelic-int-metric-ball-translate*:
 $ball x e = x +_o ball 0 e$ **for** $x :: 'a::nontrivial-factorial-idom adelic-int$
 $\langle proof \rangle$

lemma *adelic-int-metric-ball-UNIV*:

ball x $e = UNIV$ **if** $e \geq 1$ **for** $x :: 'a::nontrivial-factorial-idom$ *adelic-int*
 <proof>

lemma *adelic-int-left-translate-metric-continuous*:

continuous-on UNIV $((+) x)$ **for** $x :: 'a::nontrivial-factorial-idom$ *adelic-int*
 <proof>

lemma *adelic-int-right-translate-metric-continuous*:

continuous-on UNIV $(\lambda z. z + x)$ **for** $x :: 'a::nontrivial-factorial-idom$ *adelic-int*
 <proof>

lemma *adelic-int-left-mult-bdd-metric-continuous*:

continuous-on UNIV $((*) x)$
for $x :: 'a::nontrivial-factorial-idom$ *adelic-int*
 <proof>

lemma *adelic-int-bdd-metric-limseq-mult*:

$(\lambda k. w * X k) \longrightarrow (w * x)$ **if** $X \longrightarrow x$
for $w x :: 'a::nontrivial-factorial-idom$ *adelic-int* **and** $X :: nat \Rightarrow 'a$ *adelic-int*
 <proof>

lemma *adelic-int-global-depth-set-open*:

open $((\mathcal{P}_{\forall p}^n) :: 'a::nontrivial-factorial-idom$ *adelic-int set*)
 <proof>

lemma *adelic-int-ball-abs-eq*:

Abs-adelic-int ' *ball* x $e = ball$ (*Abs-adelic-int* x) e
if $x \in \mathcal{O}_{\forall p}$ **and** $e \leq 1$
for $x :: 'a::nontrivial-factorial-idom$ *p-adic-prod*
 <proof>

lemma *open-adelic-int-abs-eq*:

open $U = open$ (*Abs-adelic-int* ' U) **if** $U \subseteq (\mathcal{O}_{\forall p})$
for $U :: 'a::nontrivial-factorial-idom$ *p-adic-prod set*
 <proof>

Completeness

abbreviation *adelic-int-globally-cauchy* $\equiv global-p-depth-adelic-int.globally-cauchy$

abbreviation

adelic-int-global-cauchy-condition $\equiv global-p-depth-adelic-int.global-cauchy-condition$

lemma *adelic-int-global-cauchy-condition-abs-eq*:

adelic-int-global-cauchy-condition $(\lambda n. Abs-adelic-int (X n)) =$
p-adic-prod-global-cauchy-condition X
if $range X \subseteq \mathcal{O}_{\forall p}$
 <proof>

lemma *adelic-int-globally-cauchy-abs-eq*:
adelic-int-globally-cauchy ($\lambda n. \text{Abs-adelic-int } (X \ n)$) = *p-adic-prod-globally-cauchy*
 X
if $\text{range } X \subseteq \mathcal{O}_{\forall p}$
 $\langle \text{proof} \rangle$

lemma *adelic-int-globally-cauchy-vs-bdd-metric-Cauchy*:
adelic-int-globally-cauchy $X = \text{Cauchy } X$
for $X :: \text{nat} \Rightarrow 'a::\text{nontrivial-factorial-idom adelic-int}$
 $\langle \text{proof} \rangle$

lemma *adelic-int-Cauchy-abs-eq*:
 $\text{Cauchy } (\lambda n. \text{Abs-adelic-int } (X \ n)) = \text{Cauchy } X$
if $\text{range } X \subseteq \mathcal{O}_{\forall p}$
for $X :: \text{nat} \Rightarrow 'a::\text{nontrivial-factorial-idom p-adic-prod}$
 $\langle \text{proof} \rangle$

abbreviation

adelic-int-cauchy-lim $X \equiv$
Abs-adelic-int (*p-adic-prod-cauchy-lim* ($\lambda n. \text{Rep-adelic-int } (X \ n)$))

lemma *adelic-int-bdd-metric-complete'*:
 $X \longrightarrow \text{adelic-int-cauchy-lim } X$ **if** $\text{Cauchy } X$
for $X :: \text{nat} \Rightarrow 'a::\text{nontrivial-factorial-unique-euclidean-bezout adelic-int}$
 $\langle \text{proof} \rangle$

lemma *adelic-int-bdd-metric-complete*:
 $\text{complete } (\text{UNIV} :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout adelic-int set})$
 $\langle \text{proof} \rangle$

end

theory *Fin-Field-Product*

imports

Distinguished-Quotients
pAdic-Product

begin

6 Finite fields of prime order as quotients of the ring of integers of p-adic fields

6.1 The type

6.1.1 The prime ideal as the distinguished ideal of the ring of adelic integers

instantiation *adelic-int* ::
(*nontrivial-factorial-idom*) *comm-ring-1-with-distinguished-ideal*
begin

definition *distinguished-subset-adelic-int* :: 'a *adelic-int set*
where *distinguished-subset-adelic-int-def*[*simp*]:
distinguished-subset-adelic-int \equiv $\mathcal{P}_{\mathbb{V}_p}$

instance
(*proof*)

end

lemma *distinguished-subset-adelic-int-inverse*:
inverse $x = 0$ **if** $x \in \mathcal{P}_{\mathbb{V}_p}$
for $x :: 'a :: \text{nontrivial-factorial-unique-euclidean-bezout } \text{adelic-int}$
(*proof*)

6.1.2 The finite field product type as a quotient

Create type wrapper *coset-by-dist-sbgrp* over *adelic-int*.

typedef (**overloaded**) 'a *adelic-int-quot* =
UNIV :: 'a :: *nontrivial-factorial-idom adelic-int coset-by-dist-sbgrp set*
(*proof*)

abbreviation

adelic-int-quot \equiv $\lambda x. \text{Abs-adelic-int-quot } (\text{distinguished-quotient-coset } x)$

abbreviation *p-adic-prod-int-quot* \equiv $\lambda x. \text{adelic-int-quot } (\text{Abs-adelic-int } x)$

lemma *adelic-int-quot-cases* [*case-names adelic-int-quot*]:
obtains y **where** $x = \text{adelic-int-quot } y$
(*proof*)

lemma *Abs-adelic-int-quot-cases* [*case-names adelic-int-quot-Abs-adelic-int*]:
obtains z **where** $x = \text{p-adic-prod-int-quot } z$ $z \in \mathcal{O}_{\mathbb{V}_p}$
(*proof*)

lemmas *two-adelic-int-quot-cases* = *adelic-int-quot-cases*[*case-product adelic-int-quot-cases*]
and *three-adelic-int-quot-cases* =
adelic-int-quot-cases[*case-product adelic-int-quot-cases*[*case-product adelic-int-quot-cases*]]

lemma *abs-adelic-int-quot-eq-iff*:
 $\text{adelic-int-quot } x = \text{adelic-int-quot } y \iff$
 $y - x \in \mathcal{P}_{\forall p}$
 ⟨proof⟩

lemma *p-adic-prod-int-quot-eq-iff*:
 $p\text{-adic-prod-int-quot } x = p\text{-adic-prod-int-quot } y \iff$
 $y - x \in \mathcal{P}_{\forall p}$
if $x \in \mathcal{O}_{\forall p}$ **and** $y \in \mathcal{O}_{\forall p}$
 ⟨proof⟩

6.1.3 Algebraic instantiations

instantiation *adelic-int-quot* :: (nontrivial-factorial-idom) comm-ring-1
begin

definition $0 = \text{Abs-adelic-int-quot } 0$

lemma *zero-adelic-int-quot-rep-eq*: $\text{Rep-adelic-int-quot } 0 = 0$
 ⟨proof⟩

lemma *zero-adelic-int-quot*: $0 = \text{adelic-int-quot } 0$
 ⟨proof⟩

definition $1 \equiv \text{Abs-adelic-int-quot } 1$

lemma *one-adelic-int-quot-rep-eq*: $\text{Rep-adelic-int-quot } 1 = 1$
 ⟨proof⟩

lemma *one-adelic-int-quot*: $1 = \text{adelic-int-quot } 1$
 ⟨proof⟩

definition $x + y = \text{Abs-adelic-int-quot } (\text{Rep-adelic-int-quot } x + \text{Rep-adelic-int-quot } y)$

lemma *plus-adelic-int-quot-rep-eq*:
 $\text{Rep-adelic-int-quot } (a + b) = \text{Rep-adelic-int-quot } a + \text{Rep-adelic-int-quot } b$
 ⟨proof⟩

lemma *plus-adelic-int-quot-abs-eq*:
 $\text{Abs-adelic-int-quot } x + \text{Abs-adelic-int-quot } y = \text{Abs-adelic-int-quot } (x + y)$
 ⟨proof⟩

lemma *plus-adelic-int-quot*: $\text{adelic-int-quot } x + \text{adelic-int-quot } y = \text{adelic-int-quot } (x + y)$
 ⟨proof⟩

definition $-x = \text{Abs-adelic-int-quot } (- \text{Rep-adelic-int-quot } x)$

lemma *uminus-adelic-int-quot-rep-eq*: $\text{Rep-adelic-int-quot } (-x) = - \text{Rep-adelic-int-quot } x$

<proof>

lemma *uminus-adelic-int-quot-abs-eq*: $- \text{Abs-adelic-int-quot } x = \text{Abs-adelic-int-quot } (-x)$

<proof>

lemma *uminus-adelic-int-quot*: $- \text{adelic-int-quot } x = \text{adelic-int-quot } (-x)$

<proof>

definition *minus-adelic-int-quot* ::

$'a \text{ adelic-int-quot} \Rightarrow 'a \text{ adelic-int-quot} \Rightarrow 'a \text{ adelic-int-quot}$

where $\text{minus-adelic-int-quot} \equiv (\lambda x y. x + - y)$

lemma *minus-adelic-int-quot-rep-eq*:

$\text{Rep-adelic-int-quot } (x - y) = \text{Rep-adelic-int-quot } x - \text{Rep-adelic-int-quot } y$

<proof>

lemma *minus-adelic-int-quot-abs-eq*:

$\text{Abs-adelic-int-quot } x - \text{Abs-adelic-int-quot } y = \text{Abs-adelic-int-quot } (x - y)$

<proof>

lemma *minus-adelic-int-quot*: $\text{adelic-int-quot } x - \text{adelic-int-quot } y = \text{adelic-int-quot } (x - y)$

<proof>

definition $x * y = \text{Abs-adelic-int-quot } (\text{Rep-adelic-int-quot } x * \text{Rep-adelic-int-quot } y)$

lemma *times-adelic-int-quot-rep-eq*:

$\text{Rep-adelic-int-quot } (x * y) = \text{Rep-adelic-int-quot } x * \text{Rep-adelic-int-quot } y$

<proof>

lemma *times-adelic-int-quot-abs-eq*:

$\text{Abs-adelic-int-quot } x * \text{Abs-adelic-int-quot } y = \text{Abs-adelic-int-quot } (x * y)$

<proof>

lemma *times-adelic-int-quot*: $\text{adelic-int-quot } x * \text{adelic-int-quot } y = \text{adelic-int-quot } (x * y)$

<proof>

instance

<proof>

end

instantiation *adelic-int-quot* ::

$(\text{nontrivial-factorial-unique-euclidean-bezout}) \{ \text{divide-trivial}, \text{inverse} \}$

begin

lift-definition *inverse-adelic-int-coset* ::
'a adelic-int coset-by-dist-sbgrp \Rightarrow *'a adelic-int coset-by-dist-sbgrp*
is *inverse*
<proof>

definition *inverse-adelic-int-quot* :: *'a adelic-int-quot* \Rightarrow *'a adelic-int-quot*
where
inverse-adelic-int-quot *x* \equiv
Abs-adelic-int-quot (inverse-adelic-int-coset (Rep-adelic-int-quot x))

lemma *inverse-adelic-int-quot: inverse (adelic-int-quot x) = adelic-int-quot (inverse x)*
<proof>

lemma *inverse-adelic-int-quot-0[simp]: inverse (0 :: 'a adelic-int-quot) = 0*
<proof>

lemma *inverse-adelic-int-quot-1[simp]: inverse (1 :: 'a adelic-int-quot) = 1*
<proof>

definition *divide-adelic-int-quot* ::
'a adelic-int-quot \Rightarrow *'a adelic-int-quot* \Rightarrow *'a adelic-int-quot*
where *divide-adelic-int-quot-def[simp]: divide-adelic-int-quot x y* \equiv *x * inverse y*

instance *<proof>*

end

lemma *divide-adelic-int-quot: adelic-int-quot x / adelic-int-quot y = adelic-int-quot (x / y)*
for *x y :: 'a::nontrivial-factorial-unique-euclidean-bezout adelic-int*
<proof>

6.2 Equivalence relative to a prime

6.2.1 Equivalence

overloading

p-equal-adelic-int-coset \equiv
p-equal :: 'a::nontrivial-factorial-idom prime \Rightarrow
'a adelic-int coset-by-dist-sbgrp \Rightarrow
'a adelic-int coset-by-dist-sbgrp \Rightarrow *bool*
p-equal-adelic-int-quot \equiv
p-equal :: 'a::nontrivial-factorial-idom prime \Rightarrow
'a adelic-int-quot \Rightarrow *'a adelic-int-quot* \Rightarrow *bool*
p-restrict-adelic-int-coset \equiv
p-restrict :: 'a adelic-int coset-by-dist-sbgrp \Rightarrow
('a prime \Rightarrow *bool)* \Rightarrow *'a adelic-int coset-by-dist-sbgrp*

p -restrict-adelic-int-quot \equiv
 p -restrict $:: 'a$ adelic-int-quot \Rightarrow
 $('a$ prime \Rightarrow bool) $\Rightarrow 'a$ adelic-int-quot
begin

lift-definition p -equal-adelic-int-coset $::$
 $'a::$ nontrivial-factorial-idom prime \Rightarrow
 $'a$ adelic-int coset-by-dist-sbgrp \Rightarrow
 $'a$ adelic-int coset-by-dist-sbgrp \Rightarrow bool
is λp x y . $x \simeq_p y \vee ((x - y)^{\circ p}) \geq 1$
 \langle proof \rangle

definition p -equal-adelic-int-quot $::$
 $'a::$ nontrivial-factorial-idom prime \Rightarrow
 $'a$ adelic-int-quot $\Rightarrow 'a$ adelic-int-quot \Rightarrow bool
where
 p -equal-adelic-int-quot p x $y \equiv$
 $(\text{Rep-adelic-int-quot } x) \simeq_p (\text{Rep-adelic-int-quot } y)$

lift-definition p -restrict-adelic-int-coset $::$
 $'a::$ nontrivial-factorial-idom adelic-int coset-by-dist-sbgrp \Rightarrow
 $('a$ prime \Rightarrow bool) $\Rightarrow 'a$ adelic-int coset-by-dist-sbgrp
is λx P . x prestrict P
 \langle proof \rangle

definition p -restrict-adelic-int-quot $::$
 $'a::$ nontrivial-factorial-idom adelic-int-quot \Rightarrow
 $('a$ prime \Rightarrow bool) $\Rightarrow 'a$ adelic-int-quot
where
 p -restrict-adelic-int-quot x $P \equiv$
 $\text{Abs-adelic-int-quot } ((\text{Rep-adelic-int-quot } x) \text{ prestrict } P)$

end

context
fixes $p :: 'a::$ nontrivial-factorial-idom prime
begin

lemma p -equal-adelic-int-coset-abs-eq0:
distinguished-quotient-coset $x \simeq_p 0 \iff$
 $x \simeq_p 0 \vee (x^{\circ p}) \geq 1$
for $x :: 'a$ adelic-int
 \langle proof \rangle

lemma p -equal-adelic-int-coset-abs-eq1:
distinguished-quotient-coset $x \simeq_p 1 \iff$
 $x \simeq_p 1 \vee ((x - 1)^{\circ p}) \geq 1$
for $x :: 'a$ adelic-int
 \langle proof \rangle

lemma *p-equal-adelic-int-quot-abs-eq*:
 $(\text{adelic-int-quot } x) \simeq_p (\text{adelic-int-quot } y) \longleftrightarrow$
 $x \simeq_p y \vee ((x - y)^{\circ p}) \geq 1$
for $x y :: 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

lemma *p-equal-adelic-int-quot-abs-eq0*:
 $\text{adelic-int-quot } x \simeq_p 0 \longleftrightarrow$
 $x \simeq_p 0 \vee (x^{\circ p}) \geq 1$
for $x :: 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

lemma *p-equal-adelic-int-quot-abs-eq1*:
 $\text{adelic-int-quot } x \simeq_p 1 \longleftrightarrow$
 $x \simeq_p 1 \vee ((x - 1)^{\circ p}) \geq 1$
for $x :: 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

end

lemma *p-restrict-adelic-int-quot-abs-eq*:
 $(\text{adelic-int-quot } x) \text{ prestrict } P = \text{adelic-int-quot } (x \text{ prestrict } P)$
for $P :: 'a::\text{nontrivial-factorial-idom prime} \Rightarrow \text{bool}$ **and** $x :: 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

global-interpretation *p-equality-adelic-int-quot*:
p-equality-no-zero-divisors-1
 $p\text{-equal} :: 'a::\text{nontrivial-euclidean-ring-cancel prime} \Rightarrow$
 $'a \text{ adelic-int-quot} \Rightarrow 'a \text{ adelic-int-quot} \Rightarrow \text{bool}$
 $\langle \text{proof} \rangle$

overloading
 $\text{globally-p-equal-adelic-int-quot} \equiv$
 $\text{globally-p-equal} ::$
 $'a::\text{nontrivial-euclidean-ring-cancel adelic-int-quot} \Rightarrow$
 $'a \text{ adelic-int-quot} \Rightarrow \text{bool}$

begin

definition *globally-p-equal-adelic-int-quot* ::
 $'a::\text{nontrivial-euclidean-ring-cancel adelic-int-quot} \Rightarrow$
 $'a \text{ adelic-int-quot} \Rightarrow \text{bool}$
where *globally-p-equal-adelic-int-quot-def[simp]*:
 $\text{globally-p-equal-adelic-int-quot} = p\text{-equality-adelic-int-quot.globally-p-equal}$

end

6.2.2 Embedding of constants

global-interpretation *global-p-equal-adelic-int-quot*:

global-p-equal-w-consts

p-equal :: 'a::nontrivial-euclidean-ring-cancel prime \Rightarrow
 'a *adelic-int-quot* \Rightarrow 'a *adelic-int-quot* \Rightarrow bool

p-restrict ::

'a *adelic-int-quot* \Rightarrow ('a prime \Rightarrow bool) \Rightarrow
 'a *adelic-int-quot*

λf . *adelic-int-quot* (*adelic-int-consts* f)

<proof>

abbreviation *adelic-int-quot-consts* \equiv *global-p-equal-adelic-int-quot.consts*

abbreviation *adelic-int-quot-const* \equiv *global-p-equal-adelic-int-quot.const*

lemma *p-equal-adelic-int-quot-consts-iff*:

(*adelic-int-quot-consts* f) \simeq_p (*adelic-int-quot-consts* g) \longleftrightarrow
 (f p) *pmod* p = (g p) *pmod* p

(is ?L \longleftrightarrow ?R)

<proof>

lemma *p-equal0-adelic-int-quot-consts-iff*:

(*adelic-int-quot-consts* f) \simeq_p 0 \longleftrightarrow (f p) *pmod* p = 0

(is ?L \longleftrightarrow ?R)

<proof>

lemma *adelic-int-quot-consts-eq-iff*:

adelic-int-quot-consts f = *adelic-int-quot-consts* g \longleftrightarrow

($\forall p$. (f p) *pmod* p = (g p) *pmod* p)

<proof>

lemma *adelic-int-quot-consts-eq0-iff*:

(*adelic-int-quot-consts* f) = 0 \longleftrightarrow

($\forall p$. (f p) *pmod* p = 0)

<proof>

lemmas

p-equal-adelic-int-quot-const-iff =

p-equal-adelic-int-quot-consts-iff[of - λp . - λp . -]

and *p-equal0-adelic-int-quot-const-iff* = *p-equal0-adelic-int-quot-consts-iff*[of - λp . -]

and *adelic-int-quot-const-eq-iff* = *adelic-int-quot-consts-eq-iff*[of λp . - λp . -]

and *adelic-int-quot-const-eq0-iff* = *adelic-int-quot-consts-eq0-iff*[of λp . -]

lemma *adelic-int-quot-UNIV-eq-consts-range*:

(UNIV::'a::nontrivial-euclidean-ring-cancel *adelic-int-quot* set) =
 range (*adelic-int-quot-consts*)

<proof>

lemma *adelic-int-quot-constsE*:

fixes $x :: 'a::\text{nontrivial-euclidean-ring-cancel adelic-int-quot}$
obtains f **where** $x = \text{adelic-int-quot-consts } f$
 $\langle \text{proof} \rangle$

lemma *adelic-int-quot-reduced-constsE*:
fixes $x :: 'a::\text{nontrivial-euclidean-ring-cancel adelic-int-quot}$
obtains f
where $x = \text{adelic-int-quot-consts } f$
and $\forall p::'a \text{ prime. } (f\ p) \text{ pdiv } p = 0$
 $\langle \text{proof} \rangle$

lemma *adelic-int-quot-prestrict-zero-depth-abs-eq*:
 $\text{adelic-int-quot } (x \text{ prestrict } (\lambda p::'a \text{ prime. } (x^{op}) = 0)) =$
 $\text{adelic-int-quot } x$
for $x :: 'a::\text{nontrivial-euclidean-ring-cancel adelic-int}$
 $\langle \text{proof} \rangle$

6.2.3 Division and inverses

global-interpretation *p-equal-div-inv-adelic-int-quot*:
 $\text{global-p-equal-div-inv}$
 $p\text{-equal} :: 'a::\text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow$
 $'a \text{ adelic-int-quot} \Rightarrow 'a \text{ adelic-int-quot} \Rightarrow \text{bool}$
 $p\text{-restrict} ::$
 $'a \text{ adelic-int-quot} \Rightarrow ('a \text{ prime} \Rightarrow \text{bool}) \Rightarrow$
 $'a \text{ adelic-int-quot}$
 $\langle \text{proof} \rangle$

6.3 Special case of integer

context
fixes $p :: \text{int prime}$
begin

lemma *inj-on-const-prestrict-single-int-prime*:
 $\text{inj-on } (\lambda k. \text{ adelic-int-quot-const } k \text{ prestrict } ((=) p)) \{0..\text{Rep-prime } p - 1\}$
 $\langle \text{proof} \rangle$

lemma *range-prestrict-single-int-prime*:
 $\text{range } (\lambda x. x \text{ prestrict } ((=) p)) =$
 $(\lambda k. \text{ adelic-int-quot-const } k \text{ prestrict } ((=) p)) \{0..\text{Rep-prime } p - 1\}$
 $\langle \text{proof} \rangle$

lemma *finite-range-prestrict-single-int-prime*:
 $\text{finite } (\text{range } (\lambda x::\text{int adelic-int-quot. } x \text{ prestrict } ((=) p)))$
 $\langle \text{proof} \rangle$

lemma *card-range-prestrict-single-int-prime*:
 $\text{card } (\text{range } (\lambda x::\text{int adelic-int-quot. } x \text{ prestrict } ((=) p))) = \text{nat } (\text{Rep-prime } p)$
 $\langle \text{proof} \rangle$

end

end

theory *More-pAdic-Product*

imports *Fin-Field-Product*

begin

7 Compactness of local ring of integers

7.1 Preliminaries

lemma *infinite-vimageE*:

fixes $f :: 'a \Rightarrow 'b$

assumes *infinite* ($UNIV :: 'a \text{ set}$) **and** $\text{range } f \subseteq B$ **and** *finite* B

obtains b **where** $b \in B$ **and** *infinite* ($f^{-1}\{b\}$)

<proof>

primrec *subset-subseq* ::

$(\text{nat} \Rightarrow 'a) \Rightarrow 'a \text{ set} \Rightarrow \text{nat} \Rightarrow \text{nat}$

where

$\text{subset-subseq } f \ A \ 0 = (\text{LEAST } k. f \ k \in A) \mid$

$\text{subset-subseq } f \ A \ (\text{Suc } n) = (\text{LEAST } k. k > \text{subset-subseq } f \ A \ n \wedge f \ k \in A)$

lemma

assumes *infinite* ($f^{-1} \ A$)

shows *subset-subseq-strict-mono*: *strict-mono* ($\text{subset-subseq } f \ A$)

and *subset-subseq-range* : $f \ (\text{subset-subseq } f \ A \ n) \in A$

<proof>

7.2 Sequential compactness

7.2.1 Creating a Cauchy subsequence

abbreviation

p-adic-prod-int-convergent-subseq-seq-step $p \ X \ g \ n \equiv$

$\lambda k. \text{p-adic-prod-int-quot } (\text{p-adic-prod-shift-p-depth } p \ (- \ \text{int } n) \ ((X \ (g \ k) - X \ (g \ 0))))$

primrec *p-adic-prod-int-convergent-subseq-seq* ::

$'a::\text{nontrivial-factorial-unique-euclidean-bezout prime} \Rightarrow$

$(\text{nat} \Rightarrow 'a \ \text{p-adic-prod}) \Rightarrow \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{nat})$

where *p-adic-prod-int-convergent-subseq-seq* $p \ X \ 0 = \text{id} \mid$

p-adic-prod-int-convergent-subseq-seq $p \ X \ (\text{Suc } n) =$

p-adic-prod-int-convergent-subseq-seq $p \ X \ n \circ$

subset-subseq

```

(
  p-adic-prod-int-convergent-subseq-seq-step p X (
    p-adic-prod-int-convergent-subseq-seq p X n
  ) n
)
{SOME z.
  infinite (
    p-adic-prod-int-convergent-subseq-seq-step p X (
      p-adic-prod-int-convergent-subseq-seq p X n
    ) n
    - ' {z}
  )
}

```

abbreviation

```

p-adic-prod-int-convergent-subseq p X n ≡
p-adic-prod-int-convergent-subseq-seq p X n n

```

lemma restricted-X-infinite-quot-vimage:

```

∃ z. infinite ((λk. p-adic-prod-int-quot (X k)) - ' {z})
if fin-p-quot : finite (range (λx::'a adelic-int-quot. x prestrict ((=) p)))
and range-X : range X ⊆ O∇ p
and restricted-X: ∀ n. X n = (X n) prestrict ((=) p)
for p :: 'a::nontrivial-factorial-idom prime and X :: nat ⇒ 'a p-adic-prod
⟨proof⟩

```

context

```

fixes p      :: 'a::nontrivial-factorial-unique-euclidean-bezout prime
and X       :: nat ⇒ 'a p-adic-prod
and ss      :: nat ⇒ (nat ⇒ nat)
and ss-step :: nat ⇒ (nat ⇒ 'a adelic-int-quot)
and nth-im  :: nat ⇒ 'a adelic-int-quot
and subss   :: nat ⇒ (nat ⇒ nat)

```

defines

```

ss ≡ p-adic-prod-int-convergent-subseq-seq p X
and
ss-step ≡ λn. p-adic-prod-int-convergent-subseq-seq-step p X (ss n) n
and nth-im ≡ λn. (SOME z. infinite (ss-step n - ' {z}))
and subss ≡ λn. subset-subseq (ss-step n) {nth-im n}
assumes fin-p-quot : finite (range (λx::'a adelic-int-quot. x prestrict ((=) p)))
and range-X : range X ⊆ O∇ p
and restricted-X: ∀ n. X n = (X n) prestrict ((=) p)

```

begin

lemma p-adic-prod-int-convergent-subseq-seq-step-infinite-quot-vimage:

```

∃ z. infinite (ss-step n - ' {z}) (is ?A)
and p-adic-prod-int-convergent-subseq-seq-step-depth:
∀ k. (X (ss n k)) ≅p (X (ss n 0)) →
((X (ss n k) - X (ss n 0))op) ≥ int n

```

(is ?B)
 ⟨proof⟩

lemma *p-adic-prod-int-convergent-subset-subseq-strict-mono*:
 strict-mono (subset-subseq (ss-step n) {nth-im n})
 ⟨proof⟩

lemma *p-adic-prod-int-convergent-subseq-strict-mono'*: strict-mono (ss n)
 ⟨proof⟩

lemma *p-adic-prod-int-convergent-subseq-strict-mono*:
 strict-mono (p-adic-prod-int-convergent-subseq p X)
 ⟨proof⟩

lemma *p-adic-prod-int-convergent-subseq-Cauchy*:
 p-adic-prod-p-cauchy p (X ∘ p-adic-prod-int-convergent-subseq p X)
 ⟨proof⟩

lemma *p-adic-prod-int-convergent-subseq-limseq*:
 x ∈ (λz. z prestrict ((=) p)) ‘ $\mathcal{O}_{\forall p}$
 if p-adic-prod-p-open-nbhds-limseq (X ∘ g) x
 ⟨proof⟩

end

7.2.2 Proving sequential compactness

context

fixes p :: 'a::nontrivial-factorial-unique-euclidean-bezout prime

assumes fin-p-quot: finite (range (λx::'a adelic-int-quot. x prestrict ((=) p)))

begin

lemma *p-adic-prod-local-int-ring-seq-compact'*:
 ∃ g::nat⇒nat. strict-mono g ∧ p-adic-prod-p-cauchy p (X ∘ g)
 if range-X:
 range X ⊆ (λx. x prestrict ((=) p)) ‘ $(\mathcal{O}_{\forall p})$
 for X :: nat ⇒ 'a p-adic-prod
 ⟨proof⟩

lemma *p-adic-prod-local-int-ring-seq-compact*:
 ∃ x∈(λx. x prestrict ((=) p)) ‘ $(\mathcal{O}_{\forall p})$.
 ∃ g::nat⇒nat.
 strict-mono g ∧ p-adic-prod-p-open-nbhds-limseq (X ∘ g) x
 if range-X:
 range X ⊆ (λx. x prestrict ((=) p)) ‘ $(\mathcal{O}_{\forall p})$
 for X :: nat ⇒ 'a p-adic-prod
 ⟨proof⟩

lemma *adelic-int-local-int-ring-seq-abs*:

$\text{range } X \subseteq (\lambda x. x \text{ prestrict } ((=) p)) \text{ ' } (\mathcal{O}_{\forall p})$
if $\text{range-}X$: $\text{range } X \subseteq \mathcal{O}_{\forall p}$
and $\text{range-abs-}X$:
 $\text{range } (\lambda k. \text{Abs-adelic-int } (X k)) \subseteq \text{range } (\lambda x. x \text{ prestrict } ((=) p))$
for $X :: \text{nat} \Rightarrow 'a \text{ p-adic-prod}$
 $\langle \text{proof} \rangle$

lemma $\text{adelic-int-local-int-ring-seq-compact}'$:
 $\exists g :: \text{nat} \Rightarrow \text{nat}. \text{strict-mono } g \wedge \text{adelic-int-p-cauchy } p (X \circ g)$
if $\text{range-}X$:
 $\text{range } X \subseteq (\lambda x. x \text{ prestrict } ((=) p)) \text{ ' } (\mathcal{O}_{\forall p})$
for $X :: \text{nat} \Rightarrow 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

lemma $\text{adelic-int-local-int-ring-seq-compact}$:
 $\exists x \in \text{range } (\lambda x. x \text{ prestrict } ((=) p)).$
 $\exists g :: \text{nat} \Rightarrow \text{nat}.$
 $\text{strict-mono } g \wedge \text{adelic-int-p-open-nbhds-limseq } (X \circ g) x$
if $\text{range-}X$:
 $\text{range } X \subseteq \text{range } (\lambda x. x \text{ prestrict } ((=) p))$
for $X :: \text{nat} \Rightarrow 'a \text{ adelic-int}$
 $\langle \text{proof} \rangle$

end

lemma $\text{int-adic-prod-local-int-ring-seq-compact}$:
 $\exists x \in (\lambda x. x \text{ prestrict } ((=) p)) \text{ ' } (\mathcal{O}_{\forall p}).$
 $\exists g :: \text{nat} \Rightarrow \text{nat}.$
 $\text{strict-mono } g \wedge \text{p-adic-prod-p-open-nbhds-limseq } (X \circ g) x$
if $\text{range } X \subseteq (\lambda x. x \text{ prestrict } ((=) p)) \text{ ' } (\mathcal{O}_{\forall p})$
for $p :: \text{int prime}$
and $X :: \text{nat} \Rightarrow \text{int p-adic-prod}$
 $\langle \text{proof} \rangle$

lemma $\text{int-adelic-int-local-int-ring-seq-compact}$:
 $\exists x \in \text{range } (\lambda x. x \text{ prestrict } ((=) p)).$
 $\exists g :: \text{nat} \Rightarrow \text{nat}.$
 $\text{strict-mono } g \wedge \text{adelic-int-p-open-nbhds-limseq } (X \circ g) x$
if $\text{range } X \subseteq \text{range } (\lambda x. x \text{ prestrict } ((=) p))$
for $p :: \text{int prime}$
and $X :: \text{nat} \Rightarrow \text{int adelic-int}$
 $\langle \text{proof} \rangle$

7.3 Finite-open-cover compactness

function $\text{exclusion-sequence} ::$
 $'a \text{ set} \Rightarrow ('a \Rightarrow 'a \text{ set}) \Rightarrow$
 $'a \Rightarrow \text{nat} \Rightarrow 'a$
where $\text{exclusion-sequence } A f a 0 = a \mid$

$exclusion\text{-}sequence\ A\ f\ a\ (Suc\ n) =$
 $(SOME\ x.\ x \in A \wedge x \notin (\bigcup k \leq n. f\ (exclusion\text{-}sequence\ A\ f\ a\ k)))$
 $\langle proof \rangle$
termination $\langle proof \rangle$

lemma
assumes $\neg A \subseteq (\bigcup k \leq n. f\ (exclusion\text{-}sequence\ A\ f\ a\ k))$
shows $exclusion\text{-}sequence\text{-}mem:\ exclusion\text{-}sequence\ A\ f\ a\ (Suc\ n) \in A$
and $exclusion\text{-}sequence\text{-}excludes:$
 $exclusion\text{-}sequence\ A\ f\ a\ (Suc\ n) \notin (\bigcup k \leq n. f\ (exclusion\text{-}sequence\ A\ f\ a\ k))$
 $\langle proof \rangle$

context
fixes $p \quad :: 'a::nontrivial\text{-}factorial\text{-}unique\text{-}euclidean\text{-}bezout\ prime$
assumes $fin\text{-}p\text{-}quot:\ finite\ (range\ (\lambda x::'a\ adelic\text{-}int\text{-}quot.\ x\ prestrict\ ((=)\ p)))$
begin

lemma $p\text{-}adic\text{-}prod\text{-}local\text{-}int\text{-}ring\text{-}finite\text{-}cover:$
 $\exists C.\ finite\ C \wedge$
 $C \subseteq (\lambda x.\ x\ prestrict\ ((=)\ p))\ ' \mathcal{O}_{\forall p} \wedge$
 $(\lambda x.\ x\ prestrict\ ((=)\ p))\ ' \mathcal{O}_{\forall p} \subseteq$
 $(\bigcup c \in C.\ p\text{-}adic\text{-}prod\text{-}p\text{-}open\text{-}nbhd\ p\ n\ c)$
for $C :: 'a\ set$
 $\langle proof \rangle$

lemma $p\text{-}adic\text{-}prod\text{-}local\text{-}int\text{-}ring\text{-}lebesgue\text{-}number:$
 $\exists d.\ \forall x \in (\lambda x.\ x\ prestrict\ ((=)\ p))\ ' \mathcal{O}_{\forall p}.$
 $\exists A \in \mathcal{A}.\ p\text{-}adic\text{-}prod\text{-}p\text{-}open\text{-}nbhd\ p\ d\ x \subseteq A$
if $cover:$
 $(\lambda x.\ x\ prestrict\ ((=)\ p))\ ' \mathcal{O}_{\forall p} \subseteq \bigcup \mathcal{A}$
and $by\text{-}opens:$
 $\forall A \in \mathcal{A}.\ generate\text{-}topology\ (p\text{-}adic\text{-}prod\text{-}local\text{-}p\text{-}open\text{-}nbhds\ p)\ A$
 $\langle proof \rangle$

lemma $p\text{-}adic\text{-}prod\text{-}local\text{-}int\text{-}ring\text{-}compact:$
 $\exists \mathcal{B} \subseteq \mathcal{A}.\ finite\ \mathcal{B} \wedge$
 $(\lambda x.\ x\ prestrict\ ((=)\ p))\ ' \mathcal{O}_{\forall p} \subseteq \bigcup \mathcal{B}$
if $cover:$
 $(\lambda x.\ x\ prestrict\ ((=)\ p))\ ' \mathcal{O}_{\forall p} \subseteq \bigcup \mathcal{A}$
and $by\text{-}opens:$
 $\forall A \in \mathcal{A}.\ generate\text{-}topology\ (p\text{-}adic\text{-}prod\text{-}local\text{-}p\text{-}open\text{-}nbhds\ p)\ A$
 $\langle proof \rangle$

lemma $p\text{-}adic\text{-}prod\text{-}local\text{-}scaled\text{-}int\text{-}ring\text{-}compact:$
 $\exists \mathcal{B} \subseteq \mathcal{A}.\ finite\ \mathcal{B} \wedge$
 $(\lambda x.\ x\ prestrict\ ((=)\ p))\ ' (\mathcal{P}_{\forall p}^n)$
 $\subseteq \bigcup \mathcal{B}$
if $cover:$
 $(\lambda x.\ x\ prestrict\ ((=)\ p))\ ' (\mathcal{P}_{\forall p}^n)$

$\subseteq \bigcup \mathcal{A}$
and *by-opens*:
 $\forall A \in \mathcal{A}. \text{generate-topology } (p\text{-adic-prod-local-}p\text{-open-nbhds } p) A$
for $\mathcal{A} :: 'a \text{ } p\text{-adic-prod set set}$
 $\langle \text{proof} \rangle$

lemma *adelic-int-local-int-ring-compact*:
 $\exists \mathcal{B} \subseteq \mathcal{A}. \text{finite } \mathcal{B} \wedge$
 $\text{range } (\lambda x. x \text{ prestrict } ((=) p)) \subseteq \bigcup \mathcal{B}$
if *fin-p-quot*: $\text{finite } (\text{range } (\lambda x :: 'a \text{ adelic-int-quot. } x \text{ prestrict } ((=) p)))$
and *cover*: $\text{range } (\lambda x. x \text{ prestrict } ((=) p)) \subseteq \bigcup \mathcal{A}$
and *by-opens*:
 $\forall A \in \mathcal{A}. \text{generate-topology } (\text{adelic-int-local-}p\text{-open-nbhds } p) A$
for $\mathcal{A} :: 'a \text{ adelic-int set set}$
 $\langle \text{proof} \rangle$

lemma *adelic-int-local-scaled-int-ring-compact*:
 $\exists \mathcal{B} \subseteq \mathcal{A}. \text{finite } \mathcal{B} \wedge$
 $(\lambda x. x \text{ prestrict } ((=) p)) ' (\mathcal{P}_{\forall p}^n)$
 $\subseteq \bigcup \mathcal{B}$
if *fin-p-quot*: $\text{finite } (\text{range } (\lambda x :: 'a \text{ adelic-int-quot. } x \text{ prestrict } ((=) p)))$
and *cover*:
 $(\lambda x. x \text{ prestrict } ((=) p)) ' (\mathcal{P}_{\forall p}^n)$
 $\subseteq \bigcup \mathcal{A}$
and *by-opens*:
 $\forall A \in \mathcal{A}. \text{generate-topology } (\text{adelic-int-local-}p\text{-open-nbhds } p) A$
for $\mathcal{A} :: 'a \text{ adelic-int set set}$
 $\langle \text{proof} \rangle$

end

lemma *int-adic-prod-local-int-ring-compact*:
 $\exists \mathcal{B} \subseteq \mathcal{A}. \text{finite } \mathcal{B} \wedge$
 $(\lambda x. x \text{ prestrict } ((=) p)) ' \mathcal{O}_{\forall p} \subseteq \bigcup \mathcal{B}$
if $(\lambda x. x \text{ prestrict } ((=) p)) ' \mathcal{O}_{\forall p} \subseteq \bigcup \mathcal{A}$
and $\forall A \in \mathcal{A}. \text{generate-topology } (p\text{-adic-prod-local-}p\text{-open-nbhds } p) A$
for $p :: \text{int prime}$
and $\mathcal{A} :: \text{int } p\text{-adic-prod set set}$
 $\langle \text{proof} \rangle$

lemma *int-adic-prod-local-scaled-int-ring-compact*:
 $\exists \mathcal{B} \subseteq \mathcal{A}. \text{finite } \mathcal{B} \wedge$
 $(\lambda x. x \text{ prestrict } ((=) p)) ' (\mathcal{P}_{\forall p}^n)$
 $\subseteq \bigcup \mathcal{B}$
if
 $(\lambda x. x \text{ prestrict } ((=) p)) ' (\mathcal{P}_{\forall p}^n)$
 $\subseteq \bigcup \mathcal{A}$
and $\forall A \in \mathcal{A}. \text{generate-topology } (p\text{-adic-prod-local-}p\text{-open-nbhds } p) A$
for $p :: \text{int prime}$

and $\mathcal{A} :: \text{int } p\text{-adic-prod set set}$
 $\langle \text{proof} \rangle$

lemma *int-adelic-int-local-int-ring-compact:*

$\exists \mathcal{B} \subseteq \mathcal{A}. \text{finite } \mathcal{B} \wedge$
 $\text{range } (\lambda x. x \text{ prestrict } ((=) p)) \subseteq \bigcup \mathcal{B}$
if $\text{range } (\lambda x. x \text{ prestrict } ((=) p)) \subseteq \bigcup \mathcal{A}$
and $\forall A \in \mathcal{A}. \text{generate-topology } (\text{adelic-int-local-}p\text{-open-nbhds } p) A$
for $p :: \text{int prime}$
and $\mathcal{A} :: \text{int adelic-int set set}$
 $\langle \text{proof} \rangle$

lemma *int-adelic-int-local-scaled-int-ring-compact:*

$\exists \mathcal{B} \subseteq \mathcal{A}. \text{finite } \mathcal{B} \wedge$
 $(\lambda x. x \text{ prestrict } ((=) p)) ' (\mathcal{P}_{\forall p}^n)$
 $\subseteq \bigcup \mathcal{B}$
if
 $(\lambda x. x \text{ prestrict } ((=) p)) ' (\mathcal{P}_{\forall p}^n)$
 $\subseteq \bigcup \mathcal{A}$
and $\forall A \in \mathcal{A}. \text{generate-topology } (\text{adelic-int-local-}p\text{-open-nbhds } p) A$
for $p :: \text{int prime}$
and $\mathcal{A} :: \text{int adelic-int set set}$
 $\langle \text{proof} \rangle$

end

References

- [1] J. W. S. Cassels. *Local Fields*, volume 3 of *London Mathematical Society Student Texts*. Cambridge University Press, 1986.
- [2] D. S. Dummit and R. M. Foote. *Abstract Algebra*. Wiley, 3rd edition, 2004.
- [3] J. R. Munkres. *Topology*. Prentice Hall, 2nd edition, 1975.