

# $p$ -adic Fields

Aaron Crighton

May 26, 2024

## Contents

<b>1</b>	<b>The Field of Fractions of a Ring</b>	<b>6</b>
1.1	The Monoid of Nonzero Elements in a Domain . . . . .	6
1.2	Numerator and Denominator Choice Functions . . . . .	7
1.3	Defining the Field of Fractions . . . . .	9
1.3.1	Numerator and Denominator Choice Functions for <code>domain_frac</code>	10
1.3.2	The inclusion of $R$ into its fraction field . . . . .	10
1.3.3	Basic Properties of <code>numer</code> , <code>denom</code> , and <code>frac</code> . . . . .	11
1.4	The Fraction Field as a Field . . . . .	13
1.5	Facts About Ring Units . . . . .	16
1.6	Facts About Fraction Field Units . . . . .	17
<b>2</b>	<b>Multivariable Polynomials Over a Commutative Ring</b>	<b>19</b>
2.1	Lemmas about multisets . . . . .	19
2.2	Turning monomials into polynomials . . . . .	21
2.3	Degree Functions . . . . .	22
2.3.1	Total Degree Function . . . . .	22
2.3.2	Degree in One Variable . . . . .	23
2.4	Constructing the Multiplication Operation on the Ring of Indexed Polynomials . . . . .	25
2.4.1	The Set of Factors of a Fixed Monomial . . . . .	25
2.4.2	Finiteness of the Factor Set of a Monomial . . . . .	26
2.4.3	Definition of Indexed Polynomial Multiplication. . . . .	27
2.4.4	Distributivity Laws for Polynomial Multiplication . . . . .	27
2.4.5	Multiplication Commutes with <code>indexed_pmult</code> . . . . .	28
2.4.6	Associativity of Polynomial Multiplication. . . . .	28
2.4.7	Commutativity of Polynomial Multiplication . . . . .	31
2.4.8	Closure properties for multiplication . . . . .	31
2.5	Multivariable Polynomial Induction . . . . .	33
2.6	Subtraction of Polynomials . . . . .	35
2.7	The Carrier of the Ring of Indexed Polynomials . . . . .	35
2.8	Scalar Multiplication . . . . .	36

2.9	Defining the Ring of Indexed Polynomials . . . . .	37
2.10	Defining the R-Algebra of Indexed Polynomials . . . . .	41
2.11	Evaluation of Polynomials and Subring Structure . . . . .	42
2.11.1	Nesting of Polynomial Rings According to Nesting of Index Sets . . . . .	43
2.11.2	Inclusion Maps . . . . .	44
2.11.3	Restricting a Multiset to a Subset of Variables . . . . .	44
2.11.4	Total evaluation of a monomial . . . . .	46
2.11.5	Partial Evaluation of a Polynomial . . . . .	47
2.11.6	Partial Evaluation is a Homomorphism . . . . .	54
2.11.7	Total Evaluation of a Polynomial . . . . .	54
2.12	Constructing Homomorphisms from Indexed Polynomial Rings and a Universal Property . . . . .	57
2.12.1	Mapping $R[x] \rightarrow S[x]$ along a homomorphism $R \rightarrow S$ .	57
2.12.2	A Universal Property for Indexed Polynomial Rings .	59
2.13	Mapping Multivariate Polynomials over a Single Variable to Univariate Polynomials . . . . .	60
2.14	Mapping Univariate Polynomials to Multivariate Polynomials over a Singleton Variable Set . . . . .	63
2.14.1	The isomorphism $R[I \cup J] \sim R[I][J]$ , where $I$ and $J$ are disjoint variable sets . . . . .	64
2.14.2	Viewing a Multivariable Polynomial as a Univariate Polynomial over a Multivariate Polynomial Base Ring	67
2.14.3	Application: A Polynomial Ring over a Domain is a Domain . . . . .	68
2.14.4	Relabelling of Variables for Indexed Polynomial Rings	69
<b>3</b>	<b>Basic Lemmas for Manipulating Indices and Lists</b>	<b>70</b>
<b>4</b>	<b>Cartesian Powers of a Ring</b>	<b>76</b>
4.1	Constructing the Cartesian Power of a Ring . . . . .	76
4.2	Mapping the Carrier of a Ring to its 1-Dimensional Cartesian Power. . . . .	79
4.3	Simple Cartesian Products . . . . .	80
4.4	Cartesian Products at Arbitrary Indices . . . . .	83
4.5	Function Rings on Cartesian Powers . . . . .	90
4.6	Coordinate Functions . . . . .	92
4.7	Graphs of functions . . . . .	93
<b>5</b>	<b>Coordinate Rings on Cartesian Powers</b>	<b>94</b>
5.1	Basic Facts and Definitions . . . . .	94
5.2	Total Evaluation of a Polynomial . . . . .	96
5.3	Partial Evaluation of a Polynomial . . . . .	98
5.4	An induction rule for coordinate rings . . . . .	99

5.5	Algebraic Sets in Cartesian Powers . . . . .	100
5.5.1	The Zero Set of a Single Polynomial . . . . .	100
5.5.2	The Zero Set of a Collection of Polynomials . . . . .	100
5.5.3	Finite Unions and Intersections of Algebraic Sets are Algebraic . . . . .	102
5.5.4	Finite Sets Are Algebraic . . . . .	105
5.6	Polynomial Maps . . . . .	106
5.7	The Action of Index Permutations on Polynomials . . . . .	106
5.8	Inverse Images of Sets by Tuples of Polynomials . . . . .	108
5.9	Composing Polynomial Tuples With Polynomials . . . . .	111
5.10	Extensional Polynomial Maps . . . . .	113
<b>6</b>	<b>Nesting of Polynomial Rings</b>	<b>114</b>
6.1	Diagonal sets in even powers of $R$ . . . . .	117
6.2	Tuples of Functions . . . . .	118
6.3	Generic Univariate Polynomials . . . . .	122
6.4	Factoring a Polynomial as a Univariate Polynomial over a Multivariable Polynomial Ring . . . . .	126
<b>7</b>	<b>Restricted Inverse Images and Complements</b>	<b>133</b>
7.1	Inverse image of a function . . . . .	133
<b>8</b>	<b>Constructing the <math>p</math>-adic Valued Field</b>	<b>135</b>
8.1	A Locale for $p$ -adic Fields . . . . .	135
8.2	The Valuation Ring in $\mathbb{Q}_p$ . . . . .	136
8.3	The Valuation on $\mathbb{Q}_p$ . . . . .	145
8.3.1	Extending the Valuation from $\mathbb{Z}_p$ to $\mathbb{Q}_p$ . . . . .	145
8.3.2	Properties of the Valuation . . . . .	145
8.3.3	The Ultrametric Inequality on $\mathbb{Q}_p$ . . . . .	152
8.4	Constructing the Angular Component Maps on $\mathbb{Q}_p$ . . . . .	156
8.4.1	Unreduced Angular Component Map . . . . .	156
8.4.2	Reduced Angular Component Maps . . . . .	160
8.5	An Inverse for the inclusion map $\iota$ . . . . .	164
<b>9</b>	<b><math>p</math>-adic Univariate Polynomials and Hensel's Lemma</b>	<b>167</b>
9.1	Gauss Norms of Polynomials . . . . .	167
9.2	Mapping Polynomials with Value Ring Coefficients to Polynomials over $\mathbb{Z}_p$ . . . . .	170
9.3	Hensel's Lemma for $p$ -adic fields . . . . .	175
<b>10</b>	<b>Topology of <math>p</math>-adic Fields</b>	<b>176</b>
10.1	$p$ -adic Balls . . . . .	176
10.2	$p$ -adic Open Sets . . . . .	180
10.3	Convex Subsets of the Value Group . . . . .	183

<b>11</b>	<b>Generated Boolean Algebras of Sets</b>	<b>185</b>
11.1	Definitions and Basic Lemmas . . . . .	185
<b>12</b>	<b>Basic notions about boolean algebras over a set <math>S</math>, generated by a set of generators <math>B</math></b>	<b>186</b>
12.1	Turning a Family of Sets into a Family of Disjoint Sets . . . . .	188
12.2	The Atoms Generated by Collections of Sets . . . . .	190
12.2.1	Defining the Atoms of a Family of Sets . . . . .	190
12.2.2	Atoms Induced by Types of Points . . . . .	192
12.2.3	Atoms of Generated Boolean Algebras . . . . .	193
12.3	Partitions of a Set . . . . .	194
12.4	Intersections of Families of Sets . . . . .	195
<b>13</b>	<b>Cartesian Powers of <math>p</math>-adic Fields</b>	<b>197</b>
13.1	Polynomials over $\mathbb{Q}_p$ and Polynomial Maps . . . . .	198
13.2	Evaluation of Polynomials in $\mathbb{Q}_p$ . . . . .	199
13.3	Mapping Univariate Polynomials to Multivariable Polynomials in One Variable . . . . .	201
13.4	$n^{\text{th}}$ -Power Sets over $\mathbb{Q}_p$ . . . . .	203
13.5	Semialgebraic Sets . . . . .	204
13.5.1	Defining Semialgebraic Sets . . . . .	204
13.5.2	Algebraic Sets over $p$ -adic Fields . . . . .	207
13.5.3	Basic Lemmas about the Semialgebraic Predicate . . . . .	207
13.5.4	One-Dimensional Semialgebraic Sets . . . . .	209
13.5.5	Defining the $p$ -adic Valuation Semialgebraically . . . . .	210
13.5.6	Inverse Images of Semialgebraic Sets by Polynomial Maps . . . . .	212
13.5.7	One Dimensional $p$ -adic Balls are Semialgebraic . . . . .	216
13.5.8	Finite Unions and Intersections of Semialgebraic Sets . . . . .	216
13.6	Cartesian Products of Semialgebraic Sets . . . . .	218
13.7	$N^{\text{th}}$ Power Residues . . . . .	220
13.8	Semialgebraic Sets Defined by Congruences . . . . .	227
13.8.1	$p$ -adic ord Congruence Sets . . . . .	227
13.8.2	Congruence Sets for the order of the Evaluation of a Polynomial . . . . .	228
13.8.3	Congruence Sets for Angular Components . . . . .	229
13.9	Permutations of indices of semialgebraic sets . . . . .	231
13.10	Semialgebraic Functions . . . . .	233
13.10.1	Defining Semialgebraic Functions . . . . .	234
13.11	More on graphs of functions . . . . .	241
13.11.1	Tuples of Semialgebraic Functions . . . . .	245
13.11.2	Semialgebraic Functions are Closed under Composition with Semialgebraic Tuples . . . . .	253
13.11.3	Algebraic Operations on Semialgebraic Functions . . . . .	253

13.12	Sets Defined by Residues of Valuation Ring Elements . . . . .	254
<b>14</b>	<b>Rings of Semialgebraic Functions</b>	<b>261</b>
14.1	Some eint Arithmetic . . . . .	261
14.2	Lemmas on Function Ring Operations . . . . .	262
14.3	Defining the Rings of Semialgebraic Functions . . . . .	265
14.4	Defining Semialgebraic Maps . . . . .	274
14.5	Examples of Semialgebraic Maps . . . . .	277
14.6	Application of Functions to Segments of Tuples . . . . .	279
14.7	Level Sets of Semialgebraic Functions . . . . .	281
14.8	Partitioning Semialgebraic Sets According to Valuations of Functions . . . . .	285
14.9	Valuative Congruence Sets for Semialgebraic Functions . . . . .	287
14.10	Gluing Functions Along Semialgebraic Sets . . . . .	288
	14.10.1 Defining Piecewise Semialgebraic Functions . . . . .	288
	14.10.2 Turning Functions into Units Via Gluing . . . . .	291
14.11	Inclusions of Lower Dimensional Function Rings . . . . .	293
14.12	Miscellaneous . . . . .	294
14.13	Semialgebraic Polynomials . . . . .	296
	14.13.1 Common Morphisms on Polynomial Rings . . . . .	305
	14.13.2 Gluing Semialgebraic Polynomials . . . . .	308
	14.13.3 Polynomials over the Valuation Ring . . . . .	309
14.14	Partitioning Semialgebraic Sets By Zero Sets of Function . . . . .	311

### Abstract

We formalize the fields  $\mathbb{Q}_p$  of  $p$ -adic numbers within the framework of the HOL-Algebra library. The  $p$ -adic field is defined simply as the fraction field of the ring of  $p$ -adic integers. The valuation, and basic topological properties of  $\mathbb{Q}_p$  are developed, including deducing Hensel's Lemma for  $\mathbb{Q}_p$  from the same theorem for  $\mathbb{Z}_p$ . The theory of semialgebraic subsets of  $\mathbb{Q}_p^n$  and semialgebraic functions is also developed, as outlined in [1]. In order to formulate these results, general theory about multivariable polynomial rings and cartesian powers of a ring must also be developed. This work is done with a view to formalizing the proof in [1] of Macintyre's quantifier elimination theorem for semialgebraic subsets of  $\mathbb{Q}_p^n$ .

```

theory Fraction-Field
  imports HOL-Algebra.UnivPoly
           Localization-Ring.Localization
           HOL-Algebra.Subrings
           Padic-Ints.Supplementary-Ring-Facts
begin

```

# 1 The Field of Fractions of a Ring

This theory defines the fraction field of a domain and establishes its basic properties. The fraction field is defined in the standard way as the localization of a domain at its nonzero elements. This is done by importing the AFP session `Localization_Ring`. Choice functions for numerator and denominators of fractions are introduced, and the inclusion of a domain into its ring of fractions is defined.

## 1.1 The Monoid of Nonzero Elements in a Domain

**locale** *domain-frac* = *domain*

**lemma** *zero-not-in-nonzero*:  $0_R \notin \text{nonzero } R$   
*<proof>*

**lemma**(*in domain*) *nonzero-is-submonoid*: *submonoid*  $R$  (*nonzero*  $R$ )  
*<proof>*

**lemma**(*in domain*) *nonzero-closed*:  
**assumes**  $a \in \text{nonzero } R$   
**shows**  $a \in \text{carrier } R$   
*<proof>*

**lemma**(*in domain*) *nonzero-mult-closed*:  
**assumes**  $a \in \text{nonzero } R$   
**assumes**  $b \in \text{nonzero } R$   
**shows**  $a \otimes b \in \text{carrier } R$   
*<proof>*

**lemma**(*in domain*) *nonzero-one-closed*:  
 $1 \in \text{nonzero } R$   
*<proof>*

**lemma**(*in domain*) *nonzero-memI*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $a \neq 0$   
**shows**  $a \in \text{nonzero } R$   
*<proof>*

**lemma**(*in domain*) *nonzero-memE*:  
**assumes**  $a \in \text{nonzero } R$   
**shows**  $a \in \text{carrier } R$   $a \neq 0$   
*<proof>*

**lemma**(*in domain*) *not-nonzero-memE*:  
**assumes**  $a \notin \text{nonzero } R$   
**assumes**  $a \in \text{carrier } R$

**shows**  $a = \mathbf{0}$   
 ⟨*proof*⟩

**lemma**(**in** *domain*) *not-nonzero-memI*:  
**assumes**  $a = \mathbf{0}$   
**shows**  $a \notin \text{nonzero } R$   
 ⟨*proof*⟩

**lemma**(**in** *domain*) *one-nonzero*:  
 $\mathbf{1} \in \text{nonzero } R$   
 ⟨*proof*⟩

**lemma**(**in** *domain-frac*) *eq-obj-rng-of-frac-nonzero*:  
*eq-obj-rng-of-frac*  $R$  (*nonzero*  $R$ )  
 ⟨*proof*⟩

## 1.2 Numerator and Denominator Choice Functions

**definition**(**in** *eq-obj-rng-of-frac*) *denom* **where**  
*denom*  $a = (\text{if } (a = \mathbf{0}_{\text{rec-rng-of-frac}}) \text{ then } \mathbf{1} \text{ else } (\text{snd } (\text{SOME } x. x \in a)))$

The choice function for numerators must be compatible with *denom*:

**definition**(**in** *eq-obj-rng-of-frac*) *numer* **where**  
*numer*  $a = (\text{if } (a = \mathbf{0}_{\text{rec-rng-of-frac}}) \text{ then } \mathbf{0} \text{ else } (\text{fst } (\text{SOME } x. x \in a \wedge (\text{snd } x = \text{denom } a))))$

Basic properties of *numer* and *denom*:

**lemma**(**in** *eq-obj-rng-of-frac*) *numer-denom-facts0*:  
**assumes** *domain*  $R$   
**assumes**  $\mathbf{0} \notin S$   
**assumes**  $a \in \text{carrier } \text{rec-rng-of-frac}$   
**assumes**  $a \neq \mathbf{0}_{\text{rec-rng-of-frac}}$   
**shows**  $a = ((\text{numer } a) \mid_{\text{rel}} (\text{denom } a))$   
 $(\text{numer } a) \in \text{carrier } R$   
 $(\text{denom } a) \in S$   
 $\text{numer } a = \mathbf{0} \implies a = \mathbf{0}_{\text{rec-rng-of-frac}}$   
 $a \otimes_{\text{rec-rng-of-frac}} (\text{rng-to-rng-of-frac}(\text{denom } a)) = \text{rng-to-rng-of-frac } (\text{numer } a)$   
 $(\text{rng-to-rng-of-frac}(\text{denom } a)) \otimes_{\text{rec-rng-of-frac}} a = \text{rng-to-rng-of-frac } (\text{numer } a)$   
 ⟨*proof*⟩

**lemma**(**in** *eq-obj-rng-of-frac*) *frac-zero*:  
**assumes** *domain*  $R$   
**assumes**  $\mathbf{0} \notin S$   
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in S$   
**assumes**  $(a \mid_{\text{rel}} b) = \mathbf{0}_{\text{rec-rng-of-frac}}$   
**shows**  $a = \mathbf{0}$

*<proof>*

When  $S$  does not contain 0, and  $R$  is a domain, the localization is a domain.

**lemma**(*in eq-obj-rng-of-frac*) *rec-rng-of-frac-is-domain*:

**assumes** *domain R*

**assumes**  $0 \notin S$

**shows** *domain rec-rng-of-frac*

*<proof>*

**lemma**(*in eq-obj-rng-of-frac*) *numer-denom-facts*:

**assumes** *domain R*

**assumes**  $0 \notin S$

**assumes**  $a \in \text{carrier } \text{rec-rng-of-frac}$

**shows**  $a = (\text{numer } a \mid_{\text{rel}} \text{denom } a)$

$(\text{numer } a) \in \text{carrier } R$

$(\text{denom } a) \in S$

$a \neq \mathbf{0}_{\text{rec-rng-of-frac}} \implies (\text{numer } a) \neq \mathbf{0}$

$a \otimes_{\text{rec-rng-of-frac}} (\text{rng-to-rng-of-frac}(\text{denom } a)) = \text{rng-to-rng-of-frac}(\text{numer } a)$

$(\text{rng-to-rng-of-frac}(\text{denom } a)) \otimes_{\text{rec-rng-of-frac}} a = \text{rng-to-rng-of-frac}(\text{numer } a)$

*<proof>*

**lemma**(*in eq-obj-rng-of-frac*) *numer-denom-closed*:

**assumes** *domain R*

**assumes**  $0 \notin S$

**assumes**  $a \in \text{carrier } \text{rec-rng-of-frac}$

**shows**  $(\text{numer } a, \text{denom } a) \in \text{carrier rel}$

*<proof>*

Fraction function which suppresses the "rel" argument.

**definition**(*in eq-obj-rng-of-frac*) *fraction where*

*fraction*  $x \ y = (x \mid_{\text{rel}} y)$

**lemma**(*in eq-obj-rng-of-frac*) *a-is-fraction*:

**assumes** *domain R*

**assumes**  $0 \notin S$

**assumes**  $a \in \text{carrier } \text{rec-rng-of-frac}$

**shows**  $a = \text{fraction}(\text{numer } a)(\text{denom } a)$

*<proof>*

**lemma**(*in eq-obj-rng-of-frac*) *add-fraction*:

**assumes** *domain R*

**assumes**  $0 \notin S$

**assumes**  $a \in \text{carrier } R$

**assumes**  $b \in S$

**assumes**  $c \in \text{carrier } R$

**assumes**  $d \in S$

**shows**  $(\text{fraction } a \ b) \oplus_{\text{rec-rng-of-frac}} (\text{fraction } c \ d) = (\text{fraction } ((a \otimes d) \oplus (b \otimes c)) (b \otimes d))$   
 ⟨proof⟩

**lemma**(in *eq-obj-rng-of-frac*) *mult-fraction*:

**assumes** *domain R*

**assumes**  $0 \notin S$

**assumes**  $a \in \text{carrier } R$

**assumes**  $b \in S$

**assumes**  $c \in \text{carrier } R$

**assumes**  $d \in S$

**shows**  $(\text{fraction } a \ b) \otimes_{\text{rec-rng-of-frac}} (\text{fraction } c \ d) = (\text{fraction } (a \otimes c) (b \otimes d))$

⟨proof⟩

**lemma**(in *eq-obj-rng-of-frac*) *fraction-zero*:

$0_{\text{rec-rng-of-frac}} = \text{fraction } 0 \ 1$

⟨proof⟩

**lemma**(in *eq-obj-rng-of-frac*) *fraction-zero'*:

**assumes**  $a \in S$

**assumes**  $0 \notin S$

**shows**  $0_{\text{rec-rng-of-frac}} = \text{fraction } 0 \ a$

⟨proof⟩

**lemma**(in *eq-obj-rng-of-frac*) *fraction-one*:

$1_{\text{rec-rng-of-frac}} = \text{fraction } 1 \ 1$

⟨proof⟩

**lemma**(in *eq-obj-rng-of-frac*) *fraction-one'*:

**assumes** *domain R*

**assumes**  $0 \notin S$

**assumes**  $a \in S$

**shows**  $\text{fraction } a \ a = 1_{\text{rec-rng-of-frac}}$

⟨proof⟩

**lemma**(in *eq-obj-rng-of-frac*) *fraction-closed*:

**assumes** *domain R*

**assumes**  $0 \notin S$

**assumes**  $a \in \text{carrier } R$

**assumes**  $b \in S$

**shows**  $\text{fraction } a \ b \in \text{carrier rec-rng-of-frac}$

⟨proof⟩

### 1.3 Defining the Field of Fractions

**definition** *Frac* **where**

$\text{Frac } R = \text{eq-obj-rng-of-frac.rec-rng-of-frac } R$  (*nonzero R*)

**lemma**(in *domain-frac*) *fraction-field-is-domain*:

*domain* (*Frac R*)  
*<proof>*

### 1.3.1 Numerator and Denominator Choice Functions for `domain_frac`

**definition** *numerator* **where**  
*numerator R = eq-obj-rng-of-frac.numer R (nonzero R)*

**abbreviation**(**in** *domain-frac*)(*input*) *numer* **where**  
*numer*  $\equiv$  *numerator R*

**definition** *denominator* **where**  
*denominator R = eq-obj-rng-of-frac.denom R (nonzero R)*

**abbreviation**(**in** *domain-frac*)(*input*) *denom* **where**  
*denom*  $\equiv$  *denominator R*

**definition** *fraction* **where**  
*fraction R = eq-obj-rng-of-frac.fraction R (nonzero R)*

**abbreviation**(**in** *domain-frac*)(*input*) *frac* **where**  
*frac*  $\equiv$  *fraction R*

### 1.3.2 The inclusion of $\mathbf{R}$ into its fraction field

**definition** *Frac-inc* **where**  
*Frac-inc R = eq-obj-rng-of-frac.rng-to-rng-of-frac R (nonzero R)*

**abbreviation**(**in** *domain-frac*)(*input*) *inc* ( $\iota$ ) **where**  
*inc*  $\equiv$  *Frac-inc R*

**lemma**(**in** *domain-frac*) *inc-equation*:  
**assumes**  $a \in \text{carrier } R$   
**shows**  $\iota a = \text{frac } a \mathbf{1}$   
*<proof>*

**lemma**(**in** *domain-frac*) *inc-is-hom*:  
 $\text{inc} \in \text{ring-hom } R (\text{Frac } R)$   
*<proof>*

**lemma**(**in** *domain-frac*) *inc-is-hom1*:  
 $\text{ring-hom-ring } R (\text{Frac } R) \text{ inc}$   
*<proof>*

Inclusion map is injective:

**lemma**(**in** *domain-frac*) *inc-inj0*:  
 $a\text{-kernel } R (\text{Frac } R) \text{ inc} = \{\mathbf{0}\}$   
*<proof>*

**lemma**(**in** *domain-frac*) *inc-inj1*:

**assumes**  $a \in \text{carrier } R$   
**assumes**  $\text{inc } a = \mathbf{0}_{\text{Frac } R}$   
**shows**  $a = \mathbf{0}$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *domain-frac*) *inc-inj2*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{carrier } R$   
**assumes**  $\text{inc } a = \text{inc } b$   
**shows**  $a = b$   
 $\langle \text{proof} \rangle$

Image of inclusion map is a subring:

**lemma**(**in** *domain-frac*) *inc-im-is-subring*:  
*subring*  $(\iota \text{ ` } (\text{carrier } R)) (\text{Frac } R)$   
 $\langle \text{proof} \rangle$

### 1.3.3 Basic Properties of numer, denom, and frac

**lemma**(**in** *domain-frac*) *numer-denom-facts*:  
**assumes**  $a \in \text{carrier } (\text{Frac } R)$   
**shows**  $a = \text{frac } (\text{numer } a) (\text{denom } a)$   
 $(\text{numer } a) \in \text{carrier } R$   
 $(\text{denom } a) \in \text{nonzero } R$   
 $a \neq \mathbf{0}_{\text{Frac } R} \implies \text{numer } a \neq \mathbf{0}$   
 $a \otimes_{\text{Frac } R} (\text{inc } (\text{denom } a)) = \text{inc } (\text{numer } a)$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *domain-frac*) *frac-add*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{nonzero } R$   
**assumes**  $c \in \text{carrier } R$   
**assumes**  $d \in \text{nonzero } R$   
**shows**  $(\text{frac } a b) \oplus_{\text{Frac } R} (\text{frac } c d) = (\text{frac } ((a \otimes d) \oplus (b \otimes c)) (b \otimes d))$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *domain-frac*) *frac-mult*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{nonzero } R$   
**assumes**  $c \in \text{carrier } R$   
**assumes**  $d \in \text{nonzero } R$   
**shows**  $(\text{frac } a b) \otimes_{\text{Frac } R} (\text{frac } c d) = (\text{frac } (a \otimes c) (b \otimes d))$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *domain-frac*) *frac-one*:  
**assumes**  $a \in \text{nonzero } R$   
**shows**  $\text{frac } a a = \mathbf{1}_{\text{Frac } R}$   
 $\langle \text{proof} \rangle$

**lemma**(in *domain-frac*) *frac-closed*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{nonzero } R$   
**shows**  $\text{frac } a \ b \in \text{carrier } (\text{Frac } R)$   
 $\langle \text{proof} \rangle$

**lemma**(in *domain-frac*) *nonzero-fraction*:  
**assumes**  $a \in \text{nonzero } R$   
**assumes**  $b \in \text{nonzero } R$   
**shows**  $\text{frac } a \ b \neq \mathbf{0}_{\text{Frac } R}$   
 $\langle \text{proof} \rangle$

**lemma**(in *comm-monoid*) *UnitsI*:  
**assumes**  $a \in \text{carrier } G$   
**assumes**  $b \in \text{carrier } G$   
**assumes**  $a \otimes b = \mathbf{1}$   
**shows**  $a \in \text{Units } G \ b \in \text{Units } G$   
 $\langle \text{proof} \rangle$

**lemma**(in *comm-monoid*) *is-invI*:  
**assumes**  $a \in \text{carrier } G$   
**assumes**  $b \in \text{carrier } G$   
**assumes**  $a \otimes b = \mathbf{1}$   
**shows**  $\text{inv}_G \ b = a \ \text{inv}_G \ a = b$   
 $\langle \text{proof} \rangle$

**lemma**(in *ring*) *ring-in-Units-imp-not-zero*:  
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**assumes**  $a \in \text{Units } R$   
**shows**  $a \neq \mathbf{0}$   
 $\langle \text{proof} \rangle$

**lemma**(in *domain-frac*) *in-Units-imp-not-zero*:  
**assumes**  $a \in \text{Units } R$   
**shows**  $a \neq \mathbf{0}$   
 $\langle \text{proof} \rangle$

**lemma**(in *domain-frac*) *units-of-fraction-field0*:  
**assumes**  $a \in \text{carrier } (\text{Frac } R)$   
**assumes**  $a \neq \mathbf{0}_{\text{Frac } R}$   
**shows**  $\text{inv}_{\text{Frac } R} \ a = \text{frac } (\text{denom } a) \ (\text{numer } a)$   
 $a \otimes_{\text{Frac } R} (\text{inv}_{\text{Frac } R} \ a) = \mathbf{1}_{\text{Frac } R}$   
 $(\text{inv}_{\text{Frac } R} \ a) \otimes_{\text{Frac } R} a = \mathbf{1}_{\text{Frac } R}$   
 $\langle \text{proof} \rangle$

**lemma**(in *domain-frac*) *units-of-fraction-field*:  
 $\text{Units } (\text{Frac } R) = \text{carrier } (\text{Frac } R) - \{\mathbf{0}_{\text{Frac } R}\}$   
 $\langle \text{proof} \rangle$

## 1.4 The Fraction Field as a Field

**lemma**(in *domain-frac*) *fraction-field-is-field*:  
*field* (*Frac R*)  
{*proof*}

**lemma**(in *domain-frac*) *frac-inv*:  
**assumes**  $a \in \text{nonzero } R$   
**assumes**  $b \in \text{nonzero } R$   
**shows**  $\text{inv}_{\text{Frac } R} (\text{frac } a \ b) = (\text{frac } b \ a)$   
{*proof*}

**lemma**(in *domain-frac*) *frac-inv-id*:  
**assumes**  $a \in \text{nonzero } R$   
**assumes**  $b \in \text{nonzero } R$   
**assumes**  $c \in \text{nonzero } R$   
**assumes**  $d \in \text{nonzero } R$   
**assumes**  $\text{frac } a \ b = \text{frac } c \ d$   
**shows**  $\text{frac } b \ a = \text{frac } d \ c$   
{*proof*}

**lemma**(in *domain-frac*) *frac-uminus*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{nonzero } R$   
**shows**  $\ominus_{\text{Frac } R} (\text{frac } a \ b) = \text{frac } (\ominus a) \ b$   
{*proof*}

**lemma**(in *domain-frac*) *frac-eqI*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{nonzero } R$   
**assumes**  $c \in \text{carrier } R$   
**assumes**  $d \in \text{nonzero } R$   
**assumes**  $a \otimes d \ominus b \otimes c = \mathbf{0}$   
**shows**  $\text{frac } a \ b = \text{frac } c \ d$   
{*proof*}

**lemma**(in *domain-frac*) *frac-eqI'*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{nonzero } R$   
**assumes**  $c \in \text{carrier } R$   
**assumes**  $d \in \text{nonzero } R$   
**assumes**  $a \otimes d = b \otimes c$   
**shows**  $\text{frac } a \ b = \text{frac } c \ d$   
{*proof*}

**lemma**(in *domain-frac*) *frac-cancel-l*:  
**assumes**  $a \in \text{nonzero } R$   
**assumes**  $b \in \text{nonzero } R$   
**assumes**  $c \in \text{carrier } R$   
**shows**  $\text{frac } (a \otimes c) \ (a \otimes b) = \text{frac } c \ b$

*<proof>*

**lemma**(in *domain-frac*) *frac-cancel-r*:  
 **assumes**  $a \in \text{nonzero } R$   
 **assumes**  $b \in \text{nonzero } R$   
 **assumes**  $c \in \text{carrier } R$   
 **shows**  $\text{frac } (c \otimes a) (b \otimes a) = \text{frac } c b$   
*<proof>*

**lemma**(in *domain-frac*) *frac-cancel-lr*:  
 **assumes**  $a \in \text{nonzero } R$   
 **assumes**  $b \in \text{nonzero } R$   
 **assumes**  $c \in \text{carrier } R$   
 **shows**  $\text{frac } (a \otimes c) (b \otimes a) = \text{frac } c b$   
*<proof>*

**lemma**(in *domain-frac*) *frac-cancel-rl*:  
 **assumes**  $a \in \text{nonzero } R$   
 **assumes**  $b \in \text{nonzero } R$   
 **assumes**  $c \in \text{carrier } R$   
 **shows**  $\text{frac } (c \otimes a) (a \otimes b) = \text{frac } c b$   
*<proof>*

**lemma**(in *domain-frac*) *i-mult*:  
 **assumes**  $a \in \text{carrier } R$   
 **assumes**  $c \in \text{carrier } R$   
 **assumes**  $d \in \text{nonzero } R$   
 **shows**  $(\iota a) \otimes_{\text{Frac } R} (\text{frac } c d) = \text{frac } (a \otimes c) d$   
*<proof>*

**lemma**(in *domain-frac*) *frac-eqE*:  
 **assumes**  $a \in \text{carrier } R$   
 **assumes**  $b \in \text{nonzero } R$   
 **assumes**  $c \in \text{carrier } R$   
 **assumes**  $d \in \text{nonzero } R$   
 **assumes**  $\text{frac } a b = \text{frac } c d$   
 **shows**  $a \otimes d = b \otimes c$   
*<proof>*

**lemma**(in *domain-frac*) *frac-add-common-denom*:  
 **assumes**  $a \in \text{carrier } R$   
 **assumes**  $b \in \text{carrier } R$   
 **assumes**  $c \in \text{nonzero } R$   
 **shows**  $(\text{frac } a c) \oplus_{\text{Frac } R} (\text{frac } b c) = \text{frac } (a \oplus b) c$   
*<proof>*

**lemma**(in *domain-frac*) *common-denominator*:  
 **assumes**  $x \in \text{carrier } (\text{Frac } R)$   
 **assumes**  $y \in \text{carrier } (\text{Frac } R)$

**obtains**  $a b c$  **where**  
 $a \in \text{carrier } R$   
 $b \in \text{carrier } R$   
 $c \in \text{nonzero } R$   
 $x = \text{frac } a c$   
 $y = \text{frac } b c$   
 $\langle \text{proof} \rangle$

**sublocale**  $\text{domain-frac} < F: \text{field } \text{Frac } R$   
 $\langle \text{proof} \rangle$

Inclusions of natural numbers into  $(\text{Frac } R)$ :

**lemma**(**in**  $\text{domain-frac}$ )  $\text{nat-0}$ :  
 $[(0::\text{nat})]\cdot\mathbf{1} = \mathbf{0}$   
 $\langle \text{proof} \rangle$

**lemma**(**in**  $\text{domain-frac}$ )  $\text{nat-suc}$ :  
 $[\text{Suc } n]\cdot\mathbf{1} = \mathbf{1} \oplus [n]\cdot\mathbf{1}$   
 $\langle \text{proof} \rangle$

**lemma**(**in**  $\text{domain-frac}$ )  $\text{nat-inc-rep}$ :  
**fixes**  $n::\text{nat}$   
**shows**  $[n]\cdot_{\text{Frac } R} \mathbf{1}_{\text{Frac } R} = \text{frac } ([n]\cdot\mathbf{1}) \mathbf{1}$   
 $\langle \text{proof} \rangle$

**lemma**(**in**  $\text{domain-frac}$ )  $\text{pow-0}$ :  
**assumes**  $a \in \text{nonzero } R$   
**shows**  $a[\ulcorner](0::\text{nat}) = \mathbf{1}$   
 $\langle \text{proof} \rangle$

**lemma**(**in**  $\text{domain-frac}$ )  $\text{pow-suc}$ :  
**assumes**  $a \in \text{carrier } R$   
**shows**  $a[\ulcorner](\text{Suc } n) = a \otimes (a[\ulcorner]n)$   
 $\langle \text{proof} \rangle$

**lemma**(**in**  $\text{domain-frac}$ )  $\text{nat-inc-add}$ :  
 $[(n::\text{nat}) + (m::\text{nat})]\cdot\mathbf{1} = [n]\cdot\mathbf{1} \oplus [m]\cdot\mathbf{1}$   
 $\langle \text{proof} \rangle$

**lemma**(**in**  $\text{domain-frac}$ )  $\text{int-inc-add}$ :  
 $[(n::\text{int}) + (m::\text{int})]\cdot\mathbf{1} = [n]\cdot\mathbf{1} \oplus [m]\cdot\mathbf{1}$   
 $\langle \text{proof} \rangle$

**lemma**(**in**  $\text{domain-frac}$ )  $\text{nat-inc-mult}$ :  
 $[(n::\text{nat}) * (m::\text{nat})]\cdot\mathbf{1} = [n]\cdot\mathbf{1} \otimes [m]\cdot\mathbf{1}$   
 $\langle \text{proof} \rangle$

**lemma**(**in**  $\text{domain-frac}$ )  $\text{int-inc-mult}$ :  
 $[(n::\text{int}) * (m::\text{int})]\cdot\mathbf{1} = [n]\cdot\mathbf{1} \otimes [m]\cdot\mathbf{1}$

*<proof>*

## 1.5 Facts About Ring Units

**lemma**(in *ring*) *Units-nonzero*:

**assumes**  $u \in \text{Units } R$

**assumes**  $\mathbf{1}_R \neq \mathbf{0}_R$

**shows**  $u \in \text{nonzero } R$

*<proof>*

**lemma**(in *ring*) *Units-inverse*:

**assumes**  $u \in \text{Units } R$

**shows**  $\text{inv } u \in \text{Units } R$

*<proof>*

**lemma**(in *cring*) *invI*:

**assumes**  $x \in \text{carrier } R$

**assumes**  $y \in \text{carrier } R$

**assumes**  $x \otimes_R y = \mathbf{1}_R$

**shows**  $y = \text{inv }_R x$

$x = \text{inv }_R y$

*<proof>*

**lemma**(in *cring*) *inv-cancelR*:

**assumes**  $x \in \text{Units } R$

**assumes**  $y \in \text{carrier } R$

**assumes**  $z \in \text{carrier } R$

**assumes**  $y = x \otimes_R z$

**shows**  $\text{inv}_R x \otimes_R y = z$

$y \otimes_R (\text{inv}_R x) = z$

*<proof>*

**lemma**(in *cring*) *inv-cancelL*:

**assumes**  $x \in \text{Units } R$

**assumes**  $y \in \text{carrier } R$

**assumes**  $z \in \text{carrier } R$

**assumes**  $y = z \otimes_R x$

**shows**  $\text{inv}_R x \otimes_R y = z$

$y \otimes_R (\text{inv}_R x) = z$

*<proof>*

**lemma**(in *domain-frac*) *nat-pow-nonzero*:

**assumes**  $x \in \text{nonzero } R$

**shows**  $x^{[n]} \in \text{nonzero } R$

*<proof>*

**lemma**(in *monoid*) *Units-int-pow-closed*:

**assumes**  $x \in \text{Units } G$

**shows**  $x^{[n]} \in \text{Units } G$

*<proof>*

## 1.6 Facts About Fraction Field Units

**lemma**(in *domain-frac*) *frac-nonzero-Units*:  
 $nonzero (Frac R) = Units (Frac R)$   
*<proof>*

**lemma**(in *domain-frac*) *frac-nonzero-inv-Unit*:  
**assumes**  $b \in nonzero (Frac R)$   
**shows**  $inv_{Frac R} b \in Units (Frac R)$   
*<proof>*

**lemma**(in *domain-frac*) *frac-nonzero-inv-closed*:  
**assumes**  $b \in nonzero (Frac R)$   
**shows**  $inv_{Frac R} b \in carrier (Frac R)$   
*<proof>*

**lemma**(in *domain-frac*) *frac-nonzero-inv*:  
**assumes**  $b \in nonzero (Frac R)$   
**shows**  $b \otimes_{Frac R} inv_{Frac R} b = \mathbf{1}_{Frac R}$   
 $inv_{Frac R} b \otimes_{Frac R} b = \mathbf{1}_{Frac R}$   
*<proof>*

**lemma**(in *domain-frac*) *fract-cancel-right[simp]*:  
**assumes**  $a \in carrier (Frac R)$   
**assumes**  $b \in nonzero (Frac R)$   
**shows**  $b \otimes_{Frac R} (a \otimes_{Frac R} inv_{Frac R} b) = a$   
*<proof>*

**lemma**(in *domain-frac*) *fract-cancel-left[simp]*:  
**assumes**  $a \in carrier (Frac R)$   
**assumes**  $b \in nonzero (Frac R)$   
**shows**  $(a \otimes_{Frac R} inv_{Frac R} b) \otimes_{Frac R} b = a$   
*<proof>*

**lemma**(in *domain-frac*) *fract-mult-inv*:  
**assumes**  $b \in nonzero (Frac R)$   
**assumes**  $d \in nonzero (Frac R)$   
**shows**  $(inv_{Frac R} b) \otimes_{Frac R} (inv_{Frac R} d) = (inv_{Frac R} (b \otimes_{Frac R} d))$   
*<proof>*

**lemma**(in *domain-frac*) *fract-mult*:  
**assumes**  $a \in carrier (Frac R)$   
**assumes**  $b \in nonzero (Frac R)$   
**assumes**  $c \in carrier (Frac R)$   
**assumes**  $d \in nonzero (Frac R)$   
**shows**  $(a \otimes_{Frac R} inv_{Frac R} b) \otimes_{Frac R} (c \otimes_{Frac R} inv_{Frac R} d) = ((a \otimes_{Frac R} c) \otimes_{Frac R} inv_{Frac R} (b \otimes_{Frac R} d))$

*<proof>*

**lemma**(in *domain-frac*) *Frac-nat-pow-nonzero*:  
 **assumes**  $x \in \text{nonzero } (\text{Frac } R)$   
 **shows**  $x[\wedge]_{\text{Frac } R}(n::\text{nat}) \in \text{nonzero } (\text{Frac } R)$   
 *<proof>*

**lemma**(in *domain-frac*) *Frac-nonzero-nat-pow*:  
 **assumes**  $x \in \text{carrier } (\text{Frac } R)$   
 **assumes**  $n > 0$   
 **assumes**  $x[\wedge]_{\text{Frac } R}(n::\text{nat}) \in \text{nonzero } (\text{Frac } R)$   
 **shows**  $x \in \text{nonzero } (\text{Frac } R)$   
 *<proof>*

**lemma**(in *domain-frac*) *Frac-int-pow-nonzero*:  
 **assumes**  $x \in \text{nonzero } (\text{Frac } R)$   
 **shows**  $x[\wedge]_{\text{Frac } R}(n::\text{int}) \in \text{nonzero } (\text{Frac } R)$   
 *<proof>*

**lemma**(in *domain-frac*) *Frac-nonzero-int-pow*:  
 **assumes**  $x \in \text{carrier } (\text{Frac } R)$   
 **assumes**  $n > 0$   
 **assumes**  $x[\wedge]_{\text{Frac } R}(n::\text{int}) \in \text{nonzero } (\text{Frac } R)$   
 **shows**  $x \in \text{nonzero } (\text{Frac } R)$   
 *<proof>*

**lemma**(in *domain-frac*) *numer-denom-frac[simp]*:  
 **assumes**  $a \in \text{nonzero } R$   
 **assumes**  $b \in \text{nonzero } R$   
 **shows**  $\text{frac } (\text{numer } (\text{frac } a b)) (\text{denom } (\text{frac } a b)) = \text{frac } a b$   
 *<proof>*

**lemma**(in *domain-frac*) *numer-denom-swap*:  
 **assumes**  $a \in \text{nonzero } R$   
 **assumes**  $b \in \text{nonzero } R$   
 **shows**  $a \otimes (\text{denom } (\text{frac } a b)) = b \otimes (\text{numer } (\text{frac } a b))$   
 *<proof>*

**lemma**(in *domain-frac*) *numer-nonzero*:  
 **assumes**  $a \in \text{nonzero } (\text{Frac } R)$   
 **shows**  $\text{numer } a \in \text{nonzero } R$   
 *<proof>*

**lemma**(in *domain-frac*) *fraction-zero[simp]*:  
 **assumes**  $b \in \text{nonzero } R$   
 **shows**  $\text{frac } \mathbf{0} b = \mathbf{0}_{\text{Frac } R}$   
 *<proof>*

**end**

```

theory Cring-Multivariable-Poly
imports HOL-Algebra.Indexed-Polynomials Padic-Ints.Cring-Poly
begin

```

## 2 Multivariable Polynomials Over a Commutative Ring

This theory extends the content of `HOL-Algebra.Indexed_Polynomials`, mainly in the context of a commutative base ring. In particular, the ring of polynomials over an arbitrary variable set is explicitly witnessed as a ring itself, which is commutative if the base is commutative, and a domain if the base ring is a domain. A universal property for polynomial evaluation is proved, which allows us to embed polynomial rings in a ring of functions over the base ring, as well as construe multivariable polynomials as univariate polynomials over a small base polynomial ring.

```

type-synonym 'a monomial = 'a multiset
type-synonym ('b,'a) mvar-poly = 'a multiset  $\Rightarrow$  'b
type-synonym ('a,'b) ring-hom = 'a  $\Rightarrow$  'b
type-synonym 'a u-poly = nat  $\Rightarrow$  'a

```

### 2.1 Lemmas about multisets

Since multisets function as monomials in this formalization, we'll need some simple lemmas about multisets in order to define degree functions and certain lemmas about factorizations. Those are provided in this section.

```

lemma count-size:
  assumes size  $m \leq K$ 
  shows count  $m\ i \leq K$ 
  <proof>

```

The following defines the set of monomials with nonzero coefficients for a given polynomial:

```

definition monomials-of :: ('a,'c) ring-scheme  $\Rightarrow$  ('a, 'b) mvar-poly  $\Rightarrow$  ('b monomial) set
where
  monomials-of  $R\ P = \{m. P\ m \neq \mathbf{0}_R\}$ 

```

```

context ring
begin

```

```

lemma monomials-of-index-free:
  assumes index-free  $P\ i$ 
  assumes  $m \in \text{monomials-of } R\ P$ 
  shows count  $m\ i = 0$ 
  <proof>

```

**lemma** *index-freeI*:

**assumes**  $\bigwedge m. m \in \text{monomials-of } R \ P \implies \text{count } m \ i = 0$

**shows** *index-free*  $P \ i$

*<proof>*

*monomials\_of*  $R$  is subadditive

**lemma** *monomials-of-add*:

*monomials-of*  $R \ (P \oplus \ Q) \subseteq (\text{monomials-of } R \ P) \cup (\text{monomials-of } R \ Q)$

*<proof>*

**lemma** *monomials-of-add-finite*:

**assumes** *finite* (*monomials-of*  $R \ P$ )

**assumes** *finite* (*monomials-of*  $R \ Q$ )

**shows** *finite* (*monomials-of*  $R \ (P \oplus \ Q)$ )

*<proof>*

**lemma** *monomials-ofE*:

**assumes**  $m \in \text{monomials-of } R \ p$

**shows**  $p \ m \neq \mathbf{0}$

*<proof>*

**lemma** *complement-of-monomials-of*:

**assumes**  $m \notin \text{monomials-of } R \ P$

**shows**  $P \ m = \mathbf{0}$

*<proof>*

Multiplication by an indexed variable corresponds to adding that index to each monomial:

**lemma** *monomials-of-p-mult*:

*monomials-of*  $R \ (P \otimes \ i) = \{m. \exists n \in (\text{monomials-of } R \ P). m = \text{add-mset } i \ n\}$

*<proof>*

**lemma** *monomials-of-p-mult'*:

*monomials-of*  $R \ (p \otimes \ i) = \text{add-mset } i \ ' (\text{monomials-of } R \ p)$

*<proof>*

**lemma** *monomials-of-p-mult-finite*:

**assumes** *finite* (*monomials-of*  $R \ P$ )

**shows** *finite* (*monomials-of*  $R \ (P \otimes \ i)$ )

*<proof>*

Monomials of a constant either consist of the empty multiset, or nothing:

**lemma** *monomials-of-const*:

*monomials-of*  $R \ (\text{indexed-const } k) = (\text{if } (k = \mathbf{0}) \text{ then } \{\} \text{ else } \{\#\})$

*<proof>*

**lemma** *monomials-of-const-finite*:

*finite* (*monomials-of*  $R \ (\text{indexed-const } k)$ )

*<proof>*

A polynomial always has finitely many monomials:

**lemma** *monomials-finite*:  
**assumes**  $P \in \text{indexed-pset } K \ I$   
**shows** *finite (monomials-of R P)*  
 $\langle \text{proof} \rangle$   
**end**

## 2.2 Turning monomials into polynomials

Constructor for turning a monomial into a polynomial

**definition** *mset-to-IP* ::  $('a, 'b) \text{ ring-scheme} \Rightarrow 'c \text{ monomial} \Rightarrow ('a, 'c) \text{ mvar-poly}$   
**where**  
*mset-to-IP*  $R \ m \ n = (\text{if } (n = m) \text{ then } \mathbf{1}_R \text{ else } \mathbf{0}_R)$

**definition** *var-to-IP* ::  $('a, 'b) \text{ ring-scheme} \Rightarrow 'c \Rightarrow ('a, 'c) \text{ mvar-poly (pvar)}$  **where**  
*var-to-IP*  $R \ i = \text{mset-to-IP } R \ \{ \#i \# \}$

**context** *ring*  
**begin**

**lemma** *mset-to-IP-simp[simp]*:  
*mset-to-IP*  $R \ m \ m = \mathbf{1}$   
 $\langle \text{proof} \rangle$

**lemma** *mset-to-IP-simp'[simp]*:  
**assumes**  $n \neq m$   
**shows** *mset-to-IP*  $R \ m \ n = \mathbf{0}$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *cring*) *monomials-of-mset-to-IP*:  
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**shows** *monomials-of*  $R \ (\text{mset-to-IP } R \ m) = \{m\}$   
 $\langle \text{proof} \rangle$

**end**

The set of monomials of a fixed  $P$  which include a given variable:

**definition** *monomials-with* ::  $('a, 'b) \text{ ring-scheme} \Rightarrow 'c \Rightarrow ('a, 'c) \text{ mvar-poly} \Rightarrow ('c \text{ monomial}) \text{ set}$  **where**  
*monomials-with*  $R \ i \ P = \{m. m \in \text{monomials-of } R \ P \wedge i \in \# m\}$

**context** *ring*  
**begin**

**lemma** *monomials-withE*:  
**assumes**  $m \in \text{monomials-with } R \ i \ P$   
**shows**  $i \in \# m$   
 $m \in \text{monomials-of } R \ P$

*<proof>*

**lemma** *monomials-withI*:

**assumes**  $i \in \# m$

**assumes**  $m \in \text{monomials-of } R P$

**shows**  $m \in \text{monomials-with } R i P$

*<proof>*

**end**

Restricting a polynomial to be zero outside of a given monomial set:

**definition** *restrict-poly-to-monom-set* ::

$('a, 'b) \text{ ring-scheme} \Rightarrow ('a, 'c) \text{ mvar-poly} \Rightarrow ('c \text{ monomial}) \text{ set} \Rightarrow ('a, 'c) \text{ mvar-poly}$

**where**

$\text{restrict-poly-to-monom-set } R P m\text{-set } m = (\text{if } m \in m\text{-set then } P m \text{ else } \mathbf{0}_R)$

**context** *ring*

**begin**

**lemma** *restrict-poly-to-monom-set-coeff*:

**assumes** *carrier-coeff*  $P$

**shows** *carrier-coeff*  $(\text{restrict-poly-to-monom-set } R P Ms)$

*<proof>*

**lemma** *restrict-poly-to-monom-set-coeff'*:

**assumes**  $P \in \text{indexed-pset } K I$

**assumes**  $I \neq \{\}$

**assumes**  $\bigwedge m. P m \in S$

**assumes**  $\mathbf{0} \in S$

**shows**  $\bigwedge m. (\text{restrict-poly-to-monom-set } R P m\text{-set } m) \in S$

*<proof>*

**lemma** *restrict-poly-to-monom-set-monom*:

**assumes**  $P \in \text{indexed-pset } I K$

**assumes**  $m\text{-set} \subseteq \text{monomials-of } R P$

**shows**  $\text{monomials-of } R (\text{restrict-poly-to-monom-set } R P m\text{-set}) = m\text{-set}$

*<proof>*

**end**

## 2.3 Degree Functions

### 2.3.1 Total Degree Function

**lemma** *multi-set-size-count*:

**fixes**  $m :: 'c \text{ monomial}$

**shows**  $\text{size } m \geq \text{count } m i$

*<proof>*

Total degree function

**definition** *total-degree* ::  $('a, 'b) \text{ ring-scheme} \Rightarrow ('a, 'c) \text{ mvar-poly} \Rightarrow \text{nat}$  **where**

$total-degree\ R\ P = (if\ (monomials-of\ R\ P = \{\})\ then\ 0\ else\ Max\ (size\ '(\ monomials-of\ R\ P)))$

**context** *ring*  
**begin**

**lemma** *total-degree-ineq*:  
**assumes**  $m \in monomials-of\ R\ P$   
**assumes** *finite* (*monomials-of*  $R\ P$ )  
**shows**  $total-degree\ R\ P \geq size\ m$   
 $\langle proof \rangle$

**lemma** *total-degree-in-monomials-of*:  
**assumes**  $monomials-of\ R\ P \neq \{\}$   
**assumes** *finite* (*monomials-of*  $R\ P$ )  
**obtains**  $m$  **where**  $m \in monomials-of\ R\ P \wedge size\ m = total-degree\ R\ P$   
 $\langle proof \rangle$

**lemma** *total-degreeI*:  
**assumes** *finite* (*monomials-of*  $R\ P$ )  
**assumes**  $\exists m. m \in monomials-of\ R\ P \wedge size\ m = n$   
**assumes**  $\bigwedge m. m \in monomials-of\ R\ P \implies size\ m \leq n$   
**shows**  $n = total-degree\ R\ P$   
 $\langle proof \rangle$   
**end**

### 2.3.2 Degree in One Variable

**definition** *degree-in-var* ::  
 $( 'a, 'b)\ ring\ scheme \implies ( 'a, 'c)\ mvar\ poly \implies 'c \implies nat$  **where**  
 $degree-in-var\ R\ P\ i = (if\ (monomials-of\ R\ P = \{\})\ then\ 0\ else\ Max\ ((\lambda m. count\ m\ i)\ '(\ monomials-of\ R\ P)))$

**context** *ring*  
**begin**

**lemma** *degree-in-var-ineq*:  
**assumes**  $m \in monomials-of\ R\ P$   
**assumes** *finite* (*monomials-of*  $R\ P$ )  
**shows**  $degree-in-var\ R\ P\ i \geq count\ m\ i$   
 $\langle proof \rangle$

**lemma** *degree-in-var-in-monomials-of*:  
**assumes**  $monomials-of\ R\ P \neq \{\}$   
**assumes** *finite* (*monomials-of*  $R\ P$ )  
**obtains**  $m$  **where**  $m \in monomials-of\ R\ P \wedge count\ m\ i = degree-in-var\ R\ P\ i$   
 $\langle proof \rangle$

**lemma** *degree-in-varI*:

**assumes** *finite (monomials-of R P)*  
**assumes**  $\exists m. m \in \text{monomials-of } R P \wedge \text{count } m i = n$   
**assumes**  $\bigwedge c. c \in \text{monomials-of } R P \implies \text{count } c i \leq n$   
**shows**  $n = \text{degree-in-var } R P i$   
 ⟨*proof*⟩

Total degree bounds degree in a single variable:

**lemma** *total-degree-degree-in-var:*  
**assumes** *finite (monomials-of R P)*  
**shows**  $\text{total-degree } R P \geq \text{degree-in-var } R P i$   
 ⟨*proof*⟩  
**end**

The set of monomials of maximal degree in variable  $i$ :

**definition** *max-degree-monomials-in-var ::*  
*('a, 'b) ring-scheme  $\Rightarrow$  ('a, 'c) mvar-poly  $\Rightarrow$  'c  $\Rightarrow$  ('c monomial) set* **where**  
*max-degree-monomials-in-var R P i = {m. m  $\in$  monomials-of R P  $\wedge$  count m i =*  
*degree-in-var R P i}*

**context** *ring*  
**begin**

**lemma** *max-degree-monomials-in-var-memI:*  
**assumes**  $m \in \text{monomials-of } R P$   
**assumes**  $\text{count } m i = \text{degree-in-var } R P i$   
**shows**  $m \in \text{max-degree-monomials-in-var } R P i$   
 ⟨*proof*⟩

**lemma** *max-degree-monomials-in-var-memE:*  
**assumes**  $m \in \text{max-degree-monomials-in-var } R P i$   
**shows**  $m \in \text{monomials-of } R P$   
 $\text{count } m i = \text{degree-in-var } R P i$   
 ⟨*proof*⟩  
**end**

The set of monomials of  $P$  of fixed degree in variable  $i$ :

**definition** *fixed-degree-in-var ::*  
*('a, 'b) ring-scheme  $\Rightarrow$  ('a, 'c) mvar-poly  $\Rightarrow$  'c  $\Rightarrow$  nat  $\Rightarrow$  ('c monomial) set* **where**  
*fixed-degree-in-var R P i n = {m. m  $\in$  monomials-of R P  $\wedge$  count m i = n}*

**context** *ring*  
**begin**

**lemma** *fixed-degree-in-var-subset:*  
 $\text{fixed-degree-in-var } R P i n \subseteq \text{monomials-of } R P$   
 ⟨*proof*⟩

**lemma** *fixed-degree-in-var-max-degree-monomials-in-var:*  
 $\text{max-degree-monomials-in-var } R P i = \text{fixed-degree-in-var } R P i (\text{degree-in-var } R P i)$

*<proof>*

**lemma** *fixed-degree-in-varI:*

**assumes**  $m \in \text{monomials-of } R P$

**assumes**  $\text{count } m i = n$

**shows**  $m \in \text{fixed-degree-in-var } R P i n$

*<proof>*

**lemma** *fixed-degree-in-varE:*

**assumes**  $m \in \text{fixed-degree-in-var } R P i n$

**shows**  $m \in \text{monomials-of } R P$

$\text{count } m i = n$

*<proof>*

**definition** *restrict-to-var-deg ::*

$('a, 'c) \text{ mvar-poly} \Rightarrow 'c \Rightarrow \text{nat} \Rightarrow 'c \text{ monomial} \Rightarrow 'a$  **where**

$\text{restrict-to-var-deg } P i n = \text{restrict-poly-to-monom-set } R P (\text{fixed-degree-in-var } R P i n)$

**lemma** *restrict-to-var-deg-var-deg:*

**assumes** *finite* (*monomials-of*  $R P$ )

**assumes**  $Q = \text{restrict-to-var-deg } P i n$

**assumes** *monomials-of*  $R Q \neq \{\}$

**shows**  $n = \text{degree-in-var } R Q i$

*<proof>*

**lemma** *index-free-degree-in-var[simp]:*

**assumes** *index-free*  $P i$

**shows**  $\text{degree-in-var } R P i = 0$

*<proof>*

**lemma** *degree-in-var-index-free:*

**assumes**  $\text{degree-in-var } R P i = 0$

**assumes** *finite* (*monomials-of*  $R P$ )

**shows** *index-free*  $P i$

*<proof>*

**end**

## 2.4 Constructing the Multiplication Operation on the Ring of Indexed Polynomials

### 2.4.1 The Set of Factors of a Fixed Monomial

The following function maps a monomial to the set of monomials which divide it:

**definition** *mset-factors :: 'c monomial  $\Rightarrow$  ('c monomial) set* **where**

$\text{mset-factors } m = \{n. n \subseteq\# m\}$

**context** *ring*  
**begin**

**lemma** *monom-divides-factors*:  
 $n \in (\text{mset-factors } m) \longleftrightarrow n \subseteq\# m$   
 ⟨*proof*⟩

**lemma** *mset-factors-mono*:  
**assumes**  $n \subseteq\# m$   
**shows**  $\text{mset-factors } n \subseteq \text{mset-factors } m$   
 ⟨*proof*⟩

**lemma** *mset-factors-size-bound*:  
**assumes**  $n \in \text{mset-factors } m$   
**shows**  $\text{size } n \leq \text{size } m$   
 ⟨*proof*⟩

**lemma** *sets-to-inds-finite*:  
**assumes** *finite*  $I$   
**shows**  $\text{finite } S \implies \text{finite } (\text{Pi}_E S (\lambda\cdot. I))$   
 ⟨*proof*⟩

**end**

## 2.4.2 Finiteness of the Factor Set of a Monomial

This section shows that any monomial  $m$  has only finitely many factors. This is done by mapping the set of factors injectively into a finite extensional function set. Explicitly, a monomial is just mapped to its count function, restricted to the set of indices where the count is nonzero.

**definition** *mset-factors-to-fun* ::  
 $(\text{'a}, \text{'b}) \text{ ring-scheme} \Rightarrow \text{'c monomial} \Rightarrow \text{'c monomial} \Rightarrow (\text{'c} \Rightarrow \text{nat})$  **where**  
 $\text{mset-factors-to-fun } R \ m \ n = (\text{if } (n \in \text{mset-factors } m) \text{ then } (\text{restrict } (\text{count } n) (\text{set-mset } m)) \text{ else undefined})$

**context** *ring*  
**begin**

**lemma** *mset-factors-to-fun-prop*:  
**assumes**  $\text{size } m = n$   
**shows**  $\text{mset-factors-to-fun } R \ m \in (\text{mset-factors } m) \rightarrow (\text{Pi}_E (\text{set-mset } m) (\lambda\cdot. \{0.. n\}))$   
 ⟨*proof*⟩

**lemma** *mset-factors-to-fun-inj*:  
**shows** *inj-on*  $(\text{mset-factors-to-fun } R \ m) (\text{mset-factors } m)$   
 ⟨*proof*⟩

**lemma** *finite-target*:

*finite* (*PiE* (*set-mset* *m*) ( $\lambda$ -. {0..(*n*::*nat*)}))  
 ⟨*proof*⟩

A multiset has only finitely many factors:

**lemma** *mset-factors-finite*[*simp*]:  
*finite* (*mset-factors* *m*)  
 ⟨*proof*⟩

**end**

### 2.4.3 Definition of Indexed Polynomial Multiplication.

**context** *ring*  
**begin**

Monomial division:

**lemma** *monom-divide*:  
**assumes**  $n \in \text{mset-factors } m$   
**shows** (*THE*  $k. n + k = m$ ) =  $m - n$   
 ⟨*proof*⟩

A monomial is a factor of itself:

**lemma** *m-factor*[*simp*]:  
 $m \in \text{mset-factors } m$   
 ⟨*proof*⟩

**end**

The definition of polynomial multiplication:

**definition** *P-ring-mult* :: (*'a*, *'b*) *ring-scheme*  $\Rightarrow$  (*'a*, *'c*) *mvar-poly*  $\Rightarrow$  (*'a*, *'c*) *mvar-poly*  
 $\Rightarrow$  *'c monomial*  $\Rightarrow$  *'a*

**where**  
 $P\text{-ring-mult } R P Q m = (\text{finsum } R (\lambda x. (P x) \otimes_R (Q (m - x)))) (\text{mset-factors } m)$

**abbreviation**(*in ring*) *P-ring-mult-in-ring* (**infixl**  $\otimes_p$  65)**where**  
 $P\text{-ring-mult-in-ring} \equiv P\text{-ring-mult } R$

### 2.4.4 Distributivity Laws for Polynomial Multiplication

**context** *ring*  
**begin**

**lemma** *P-ring-rdistr*:  
**assumes** *carrier-coeff* *a*  
**assumes** *carrier-coeff* *b*  
**assumes** *carrier-coeff* *c*  
**shows**  $a \otimes_p (b \oplus c) = (a \otimes_p b) \oplus (a \otimes_p c)$   
 ⟨*proof*⟩

**lemma** *P-ring-ldistr*:  
**assumes** *carrier-coeff a*  
**assumes** *carrier-coeff b*  
**assumes** *carrier-coeff c*  
**shows**  $(b \oplus c) \otimes_p a = (b \otimes_p a) \oplus (c \otimes_p a)$   
 $\langle proof \rangle$   
**end**

#### 2.4.5 Multiplication Commutes with indexed\_pmult

**context** *ring*  
**begin**

This lemma shows how we can write the factors of a monomial  $m$  times a variable  $i$  in terms of the factors of  $m$ :

**lemma** *mset-factors-add-mset*:  
*mset-factors (add-mset i m) = mset-factors m  $\cup$  add-mset i ' (mset-factors m)*  
 $\langle proof \rangle$

**end**

#### 2.4.6 Associativity of Polynomial Multiplication.

**context** *ring*  
**begin**

**lemma** *finsum-eq*:  
**assumes**  $f \in S \rightarrow carrier R$   
**assumes**  $g \in S \rightarrow carrier R$   
**assumes**  $(\lambda x \in S. f x) = (\lambda x \in S. g x)$   
**shows**  $finsum R f S = finsum R g S$   
 $\langle proof \rangle$

**lemma** *finsum-eq-induct*:  
**assumes**  $f \in S \rightarrow carrier R$   
**assumes**  $g \in T \rightarrow carrier R$   
**assumes** *finite S*  
**assumes** *finite T*  
**assumes** *bij-betw h S T*  
**assumes**  $\bigwedge s. s \in S \implies f s = g (h s)$   
**shows**  $finite U \implies U \subseteq S \implies finsum R f U = finsum R g (h ' U)$   
 $\langle proof \rangle$

**lemma** *finsum-bij-eq*:  
**assumes**  $f \in S \rightarrow carrier R$   
**assumes**  $g \in T \rightarrow carrier R$   
**assumes** *finite S*  
**assumes** *bij-betw h S T*  
**assumes**  $\bigwedge s. s \in S \implies f s = g (h s)$

**shows**  $\text{finsum } R \ f \ S = \text{finsum } R \ g \ T$   
 ⟨proof⟩

**lemma** *monom-comp*:  
**assumes**  $x \subseteq\# m$   
**assumes**  $y \subseteq\# m - x$   
**shows**  $x \subseteq\# m - y$   
 ⟨proof⟩

**lemma** *monom-comp'*:  
**assumes**  $x \subseteq\# m$   
**assumes**  $y = m - x$   
**shows**  $x = m - y$   
 ⟨proof⟩

This lemma turns iterated sums into sums over a product set. The first lemma is only a technical phrasing of `double_finsum'` to facilitate an inductive proof, and likely can and should be simplified.

**lemma** *double-finsum-induct*:  
**assumes** *finite*  $S$   
**assumes**  $\bigwedge s. s \in S \implies \text{finite } (F \ s)$   
**assumes**  $P = (\lambda S. \{(s, t). s \in S \wedge t \in (F \ s)\})$   
**assumes**  $\bigwedge s \ y. s \in S \implies y \in (F \ s) \implies g \ s \ y \in \text{carrier } R$   
**shows**  $\text{finite } T \implies T \subseteq S \implies \text{finsum } R \ (\lambda s. \text{finsum } R \ (g \ s) \ (F \ s)) \ T =$   
 $\text{finsum } R \ (\lambda c. g \ (\text{fst } c) \ (\text{snd } c)) \ (P \ T)$   
 ⟨proof⟩

**lemma** *double-finsum*:  
**assumes** *finite*  $S$   
**assumes**  $\bigwedge s. s \in S \implies \text{finite } (F \ s)$   
**assumes**  $P = \{(s, t). s \in S \wedge t \in (F \ s)\}$   
**assumes**  $\bigwedge s \ y. s \in S \implies y \in (F \ s) \implies g \ s \ y \in \text{carrier } R$   
**shows**  $\text{finsum } R \ (\lambda s. \text{finsum } R \ (g \ s) \ (F \ s)) \ S =$   
 $\text{finsum } R \ (\lambda p. g \ (\text{fst } p) \ (\text{snd } p)) \ P$   
 ⟨proof⟩

**end**

The product index set for the double sums in the coefficients of the  $((a \otimes_p b) \otimes_p c)$ . It is the set of pairs  $(x, y)$  of monomials, such that  $x$  is a factor of monomial  $m$ , and  $y$  is a factor of monomial  $x$ .

**definition** *right-cuts* :: '*c monomial*  $\implies$  ('*c monomial*  $\times$  '*c monomial*) set **where**  
*right-cuts*  $m = \{(x, y). x \subseteq\# m \wedge y \subseteq\# x\}$

**context** *ring*  
**begin**

**lemma** *right-cuts-alt-def*:  
*right-cuts*  $m = \{(x, y). x \in \text{mset-factors } m \wedge y \in \text{mset-factors } x\}$

*<proof>*

**lemma** *right-cuts-finite:*

*finite (right-cuts m)*

*<proof>*

**lemma** *assoc-aux0:*

**assumes** *carrier-coeff a*

**assumes** *carrier-coeff b*

**assumes** *carrier-coeff c*

**assumes**  $g = (\lambda x y. a x \otimes (b y \otimes c (m - x - y)))$

**shows**  $\bigwedge s y. s \in \text{mset-factors } m \implies y \in \text{mset-factors } (m - x)$   
 $\implies g s y \in \text{carrier } R$

*<proof>*

**lemma** *assoc-aux1:*

**assumes** *carrier-coeff a*

**assumes** *carrier-coeff b*

**assumes** *carrier-coeff c*

**assumes**  $g = (\lambda x y. (a y \otimes b (x - y)) \otimes c (m - x))$

**shows**  $\bigwedge s y. s \in \text{mset-factors } m \implies y \in \text{mset-factors } x \implies g s y \in \text{carrier } R$

*<proof>*

**end**

The product index set for the double sums in the coefficients of the  $(a \otimes_p (b \otimes_p c))$ . It is the set of pairs  $(x, y)$  such that  $x$  is a factor of  $m$  and  $y$  is a factor of  $m/x$ .

**definition** *left-cuts* :: *'c monomial*  $\Rightarrow$  *('c monomial*  $\times$  *'c monomial)* *set* **where**  
*left-cuts m* =  $\{(x, y). x \subseteq \#m \wedge y \subseteq \#(m - x)\}$

**context** *ring*

**begin**

**lemma** *left-cuts-alt-def:*

*left-cuts m* =  $\{(x, y). x \in \text{mset-factors } m \wedge y \in \text{mset-factors } (m - x)\}$

*<proof>*

This lemma witnesses the bijection between left and right cuts for the proof of associativity:

**lemma** *left-right-cuts-bij:*

*bij-betw*  $(\lambda (x, y). (x + y, x))$  *(left-cuts m)* *(right-cuts m)*

*<proof>*

**lemma** *left-cuts-sum:*

**assumes** *carrier-coeff a*

**assumes** *carrier-coeff b*

**assumes** *carrier-coeff c*

**shows**  $(a \otimes_p (b \otimes_p c)) m = (\bigoplus p \in \text{left-cuts } m. a (\text{fst } p) \otimes (b (\text{snd } p) \otimes c (m - (\text{fst } p) - (\text{snd } p))))$

*<proof>*

**lemma** *right-cuts-sum*:

**assumes** *carrier-coeff a*

**assumes** *carrier-coeff b*

**assumes** *carrier-coeff c*

**shows**  $(a \otimes_p b \otimes_p c) m = (\bigoplus p \in \text{right-cuts } m. a (\text{snd } p) \otimes (b ((\text{fst } p) - (\text{snd } p)) \otimes c (m - (\text{fst } p))))$

*<proof>*

The Associativity of Polynomial Multiplication:

**lemma** *P-ring-mult-assoc*:

**assumes** *carrier-coeff a*

**assumes** *carrier-coeff b*

**assumes** *carrier-coeff c*

**shows**  $a \otimes_p (b \otimes_p c) = (a \otimes_p b) \otimes_p c$

*<proof>*

**end**

## 2.4.7 Commutativity of Polynomial Multiplication

**context** *ring*

**begin**

**lemma** *mset-factors-bij*:

*bij-betw*  $(\lambda x. m - x)$   $(\text{mset-factors } m)$   $(\text{mset-factors } m)$

*<proof>*

**lemma**(**in** *cring*) *P-ring-mult-comm*:

**assumes** *carrier-coeff a*

**assumes** *carrier-coeff b*

**shows**  $a \otimes_p b = b \otimes_p a$

*<proof>*

## 2.4.8 Closure properties for multiplication

Building monomials from polynomials:

**lemma** *indexed-const-P-mult-eq*:

**assumes**  $a \in \text{carrier } R$

**assumes**  $b \in \text{carrier } R$

**shows**  $(\text{indexed-const } a) \otimes_p (\text{indexed-const } b) = \text{indexed-const } (a \otimes b)$

*<proof>*

**lemma** *indexed-const-P-multE*:

**assumes**  $P \in \text{indexed-pset } I (\text{carrier } R)$

**assumes**  $c \in \text{carrier } R$

**shows**  $(P \otimes_p (\text{indexed-const } c)) m = (P m) \otimes c$

*<proof>*

**lemma** *indexed-const-var-mult:*

**assumes**  $P \in \text{indexed-pset } I \text{ (carrier } R)$

**assumes**  $c \in \text{carrier } R$

**assumes**  $i \in I$

**shows**  $P \otimes i \otimes_p \text{indexed-const } c = (P \otimes_p (\text{indexed-const } c)) \otimes i$

*<proof>*

**lemma** *indexed-const-P-mult-closed:*

**assumes**  $a \in \text{indexed-pset } I \text{ (carrier } R)$

**assumes**  $c \in \text{carrier } R$

**shows**  $a \otimes_p (\text{indexed-const } c) \in \text{indexed-pset } I \text{ (carrier } R)$

*<proof>*

**lemma** *monom-add-mset:*

$\text{mset-to-IP } R \text{ (add-mset } i \ m) = \text{mset-to-IP } R \ m \otimes i$

*<proof>*

Monomials are closed under multiplication:

**lemma** *monom-mult:*

$\text{mset-to-IP } R \ m \otimes_p \text{mset-to-IP } R \ n = \text{mset-to-IP } R \ (m + n)$

*<proof>*

**lemma** *poly-index-mult:*

**assumes**  $a \in \text{indexed-pset } I \text{ (carrier } R)$

**assumes**  $i \in I$

**shows**  $a \otimes i = a \otimes_p \text{mset-to-IP } R \ \{\#i\# \}$

*<proof>*

**lemma** *mset-to-IP-mult-closed:*

**assumes**  $a \in \text{indexed-pset } I \text{ (carrier } R)$

**shows**  $\text{set-mset } m \subseteq I \implies a \otimes_p \text{mset-to-IP } R \ m \in \text{indexed-pset } I \text{ (carrier } R)$

*<proof>*

A predicate which identifies when the variables used in a given polynomial  $P$  are only drawn from a fixed variable set  $I$ .

**abbreviation** *monoms-in where*

$\text{monoms-in } I \ P \equiv (\forall m \in \text{monomials-of } R \ P. \ \text{set-mset } m \subseteq I)$

**lemma** *monoms-of-const:*

$\text{monomials-of } R \ (\text{indexed-const } k) = (\text{if } k = \mathbf{0} \text{ then } \{\} \text{ else } \{\{\#\}\})$

*<proof>*

**lemma** *const-monoms-in:*

$\text{monoms-in } I \ (\text{indexed-const } k)$

*<proof>*

**lemma** *mset-to-IP-indices:*

**shows**  $P \in \text{indexed-pset } I \text{ (carrier } R) \implies \text{monoms-in } I \ P$

*<proof>*

**lemma** *mset-to-IP-indices'*:

**assumes**  $P \in \text{indexed-pset } I \text{ (carrier } R)$

**assumes**  $m \in \text{monomials-of } R \ P$

**shows**  $\text{set-mset } m \subseteq I$

*<proof>*

**lemma** *one-mset-to-IP*:

$\text{mset-to-IP } R \ \{\#\} = \text{indexed-const } \mathbf{1}$

*<proof>*

**lemma** *mset-to-IP-closed*:

**shows**  $\text{set-mset } m \subseteq I \implies \text{mset-to-IP } R \ m \in \text{indexed-pset } I \text{ (carrier } R)$

*<proof>*

**lemma** *mset-to-IP-closed'*:

**assumes**  $P \in \text{indexed-pset } I \text{ (carrier } R)$

**assumes**  $m \in \text{monomials-of } R \ P$

**shows**  $\text{mset-to-IP } R \ m \in \text{indexed-pset } I \text{ (carrier } R)$

*<proof>*

This lemma expresses closure under multiplication for indexed polynomials.

**lemma** *P-ring-mult-closed*:

**assumes** *carrier-coeff*  $P$

**assumes** *carrier-coeff*  $Q$

**shows** *carrier-coeff*  $(P\text{-ring-mult } R \ P \ Q)$

*<proof>*

## 2.5 Multivariable Polynomial Induction

**lemma** *mpoly-induct*:

**assumes**  $\bigwedge Q. Q \in \text{indexed-pset } I \text{ (carrier } R) \wedge \text{card (monomials-of } R \ Q) = 0$   
 $\implies P \ Q$

**assumes**  $\bigwedge n. (\bigwedge Q. Q \in \text{indexed-pset } I \text{ (carrier } R) \wedge \text{card (monomials-of } R \ Q)$   
 $\leq n \implies P \ Q)$

$\implies (\bigwedge Q. Q \in \text{indexed-pset } I \text{ (carrier } R) \wedge \text{card (monomials-of } R \ Q) \leq$   
 $(\text{Suc } n) \implies P \ Q)$

**assumes**  $Q \in \text{indexed-pset } I \text{ (carrier } R)$

**shows**  $P \ Q$

*<proof>*

**lemma** *monomials-of-card-zero*:

**assumes**  $Q \in \text{indexed-pset } I \text{ (carrier } R) \wedge \text{card (monomials-of } R \ Q) = 0$

**shows**  $Q = \text{indexed-const } \mathbf{0}$

*<proof>*

Polynomial induction on the number of monomials with nonzero coefficient:

**lemma** *mpoly-induct'*:

**assumes**  $P$  (*indexed-const*  $\mathbf{0}$ )  
**assumes**  $\bigwedge n. (\bigwedge Q. Q \in \text{indexed-pset } I (\text{carrier } R) \wedge \text{card } (\text{monomials-of } R \ Q) \leq n \implies P \ Q)$   
 $\implies (\bigwedge Q. Q \in \text{indexed-pset } I (\text{carrier } R) \wedge \text{card } (\text{monomials-of } R \ Q) = (\text{Suc } n) \implies P \ Q)$   
**assumes**  $Q \in \text{indexed-pset } I (\text{carrier } R)$   
**shows**  $P \ Q$   
 $\langle \text{proof} \rangle$

**lemma** *monomial-poly-split*:  
**assumes**  $P \in \text{indexed-pset } I (\text{carrier } R)$   
**assumes**  $m \in \text{monomials-of } R \ P$   
**shows**  $(\text{restrict-poly-to-monom-set } R \ P ((\text{monomials-of } R \ P) - \{m\})) \oplus ((\text{mset-to-IP } R \ m) \otimes_p (\text{indexed-const } (P \ m))) = P$   
 $\langle \text{proof} \rangle$

**lemma** *restrict-not-in-monom*s:  
**assumes**  $a \notin \text{monomials-of } R \ P$   
**shows**  $\text{restrict-poly-to-monom-set } R \ P \ A = \text{restrict-poly-to-monom-set } R \ P (\text{insert } a \ A)$   
 $\langle \text{proof} \rangle$

**lemma** *restriction-closed'*:  
**assumes**  $P \in \text{indexed-pset } I (\text{carrier } R)$   
**assumes** *finite*  $ms$   
**shows**  $(\text{restrict-poly-to-monom-set } R \ P \ ms) \in \text{indexed-pset } I (\text{carrier } R)$   
 $\langle \text{proof} \rangle$

**lemma** *restriction-restrict*:  
 $\text{restrict-poly-to-monom-set } R \ P \ ms = \text{restrict-poly-to-monom-set } R \ P (ms \cap \text{monomials-of } R \ P)$   
 $\langle \text{proof} \rangle$

**lemma** *restriction-closed*:  
**assumes**  $P \in \text{indexed-pset } I (\text{carrier } R)$   
**assumes**  $Q = \text{restrict-poly-to-monom-set } R \ P \ ms$   
**shows**  $Q \in \text{indexed-pset } I (\text{carrier } R)$   
 $\langle \text{proof} \rangle$

**lemma** *monomial-split-card*:  
**assumes**  $P \in \text{indexed-pset } I (\text{carrier } R)$   
**assumes**  $m \in \text{monomials-of } R \ P$   
**shows**  $\text{card } (\text{monomials-of } R \ (\text{restrict-poly-to-monom-set } R \ P ((\text{monomials-of } R \ P) - \{m\}))) = \text{card } (\text{monomials-of } R \ P) - 1$   
 $\langle \text{proof} \rangle$

**lemma** *P-ring-mult-closed'*:  
**assumes**  $a \in \text{indexed-pset } I (\text{carrier } R)$

**assumes**  $b \in \text{indexed-pset } I \text{ (carrier } R)$   
**shows**  $a \otimes_p b \in \text{indexed-pset } I \text{ (carrier } R)$   
 ⟨proof⟩

**end**

## 2.6 Subtraction of Polynomials

**definition**  $P\text{-ring-uminus} :: ('a, 'b) \text{ ring-scheme} \Rightarrow ('a, 'c) \text{ mvar-poly} \Rightarrow ('a, 'c) \text{ mvar-poly}$  **where**  
 $P\text{-ring-uminus } R P = (\lambda m. \ominus_R (P m))$

**context** *ring*

**begin**

**lemma**  $P\text{-ring-uminus-eq}$ :

**assumes**  $a \in \text{indexed-pset } I \text{ (carrier } R)$   
**shows**  $P\text{-ring-uminus } R a = a \otimes_p (\text{indexed-const } (\ominus \mathbf{1}))$   
 ⟨proof⟩

**lemma**  $P\text{-ring-uminus-closed}$ :

**assumes**  $a \in \text{indexed-pset } I \text{ (carrier } R)$   
**shows**  $P\text{-ring-uminus } R a \in \text{indexed-pset } I \text{ (carrier } R)$   
 ⟨proof⟩

**lemma**  $P\text{-ring-uminus-add}$ :

**assumes**  $a \in \text{indexed-pset } I \text{ (carrier } R)$   
**shows**  $P\text{-ring-uminus } R a \oplus a = \text{indexed-const } \mathbf{0}$   
 ⟨proof⟩

multiplication by 1

**lemma**  $\text{one-mult-left}$ :

**assumes**  $a \in \text{indexed-pset } I \text{ (carrier } R)$   
**shows**  $(\text{indexed-const } \mathbf{1}) \otimes_p a = a$   
 ⟨proof⟩

**end**

## 2.7 The Carrier of the Ring of Indexed Polynomials

**abbreviation**(*input*)  $P\text{ring-set}$  **where**

$P\text{ring-set } R I \equiv \text{ring.indexed-pset } R I \text{ (carrier } R)$

**context** *ring*

**begin**

**lemma** *Pring-set-zero*:  
**assumes**  $f \in \text{Pring-set } R \ I$   
**assumes**  $\neg \text{set-mset } m \subseteq I$   
**shows**  $f \ m = \mathbf{0}_R$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *ring*) *Pring-cfs-closed*:  
**assumes**  $P \in \text{Pring-set } R \ I$   
**shows**  $P \ m \in \text{carrier } R$   
 $\langle \text{proof} \rangle$

**lemma** *indexed-pset-mono-aux*:  
**assumes**  $P \in \text{indexed-pset } I \ S$   
**shows**  $S \subseteq T \implies P \in \text{indexed-pset } I \ T$   
 $\langle \text{proof} \rangle$

**lemma** *indexed-pset-mono*:  
**assumes**  $S \subseteq T$   
**shows**  $\text{indexed-pset } I \ S \subseteq \text{indexed-pset } I \ T$   
 $\langle \text{proof} \rangle$

**end**

## 2.8 Scalar Multiplication

**definition** *poly-scalar-mult* ::  $('a, 'b) \text{ ring-scheme} \Rightarrow 'a \Rightarrow ('a, 'c) \text{ mvar-poly} \Rightarrow ('a, 'c) \text{ mvar-poly}$  **where**  
*poly-scalar-mult*  $R \ c \ P = (\lambda m. c \otimes_R P \ m)$

**lemma**(**in** *cring*) *poly-scalar-mult-eq*:  
**assumes**  $c \in \text{carrier } R$   
**shows**  $P \in \text{Pring-set } R \ (I :: 'c \ \text{set}) \implies \text{poly-scalar-mult } R \ c \ P = \text{indexed-const } c \ \otimes_p \ P$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *cring*) *poly-scalar-mult-const*:  
**assumes**  $c \in \text{carrier } R$   
**assumes**  $k \in \text{carrier } R$   
**shows**  $\text{poly-scalar-mult } R \ k \ (\text{indexed-const } c) = \text{indexed-const } (k \ \otimes \ c)$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *cring*) *poly-scalar-mult-closed*:  
**assumes**  $c \in \text{carrier } R$   
**assumes**  $P \in \text{Pring-set } R \ I$   
**shows**  $\text{poly-scalar-mult } R \ c \ P \in \text{Pring-set } R \ I$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *cring*) *poly-scalar-mult-zero*:  
**assumes**  $P \in \text{Pring-set } R \ I$

**shows**  $\text{poly-scalar-mult } R \mathbf{0} P = \text{indexed-const } \mathbf{0}$   
 $\langle \text{proof} \rangle$

**lemma**(*in cring*) *poly-scalar-mult-one*:  
**assumes**  $P \in \text{Pring-set } R I$   
**shows**  $\text{poly-scalar-mult } R \mathbf{1} P = P$   
 $\langle \text{proof} \rangle$

**lemma**(*in cring*) *times-poly-scalar-mult*:  
**assumes**  $P \in \text{Pring-set } R I$   
**assumes**  $Q \in \text{Pring-set } R I$   
**assumes**  $k \in \text{carrier } R$   
**shows**  $P \otimes_p (\text{poly-scalar-mult } R k Q) = \text{poly-scalar-mult } R k (P \otimes_p Q)$   
 $\langle \text{proof} \rangle$

**lemma**(*in cring*) *poly-scalar-mult-times*:  
**assumes**  $P \in \text{Pring-set } R I$   
**assumes**  $Q \in \text{Pring-set } R I$   
**assumes**  $k \in \text{carrier } R$   
**shows**  $\text{poly-scalar-mult } R k (Q \otimes_p P) = (\text{poly-scalar-mult } R k Q) \otimes_p P$   
 $\langle \text{proof} \rangle$

## 2.9 Defining the Ring of Indexed Polynomials

**definition**  $\text{Pring} :: ('b, 'e) \text{ring-scheme} \Rightarrow 'a \text{ set} \Rightarrow ('b, ('b, 'a) \text{mvar-poly}) \text{module}$   
**where**

$\text{Pring } R I \equiv (\mid \text{carrier} = \text{Pring-set } R I,$   
 $\text{Group.monoid.mult} = \text{P-ring-mult } R ,$   
 $\text{one} = \text{ring.indexed-const } R \mathbf{1}_R,$   
 $\text{zero} = \text{ring.indexed-const } R \mathbf{0}_R,$   
 $\text{add} = \text{ring.indexed-padd } R,$   
 $\text{smult} = \text{poly-scalar-mult } R)$

**context** *ring*

**begin**

**lemma** *Pring-car*:  
 $\text{carrier } (\text{Pring } R I) = \text{Pring-set } R I$   
 $\langle \text{proof} \rangle$

Definitions of the operations and constants:

**lemma** *Pring-mult*:  
 $a \otimes_{\text{Pring } R I} b = a \otimes_p b$   
 $\langle \text{proof} \rangle$

**lemma** *Pring-add*:  
 $a \oplus_{\text{Pring } R I} b = a \oplus b$

$\langle \text{proof} \rangle$

**lemma** *Pring-zero*:

$\mathbf{0}_{\text{Pring } R \ I} = \text{indexed-const } \mathbf{0}$

$\langle \text{proof} \rangle$

**lemma** *Pring-one*:

$\mathbf{1}_{\text{Pring } R \ I} = \text{indexed-const } \mathbf{1}$

$\langle \text{proof} \rangle$

**lemma** *Pring-smult*:

$(\odot_{\text{Pring } R \ I}) = (\text{poly-scalar-mult } R)$

$\langle \text{proof} \rangle$

**lemma** *Pring-carrier-coeff*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $\text{carrier-coeff } a$

$\langle \text{proof} \rangle$

**lemma** *Pring-carrier-coeff'[simp]*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \ m \in \text{carrier } R$

$\langle \text{proof} \rangle$

**lemma** *Pring-add-closed*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $b \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \oplus_{\text{Pring } R \ I} b \in \text{carrier } (\text{Pring } R \ I)$

$\langle \text{proof} \rangle$

**lemma** *Pring-mult-closed*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $b \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \otimes_{\text{Pring } R \ I} b \in \text{carrier } (\text{Pring } R \ I)$

$\langle \text{proof} \rangle$

**lemma** *Pring-one-closed*:

$\mathbf{1}_{\text{Pring } R \ I} \in \text{carrier } (\text{Pring } R \ I)$

$\langle \text{proof} \rangle$

**lemma** *Pring-zero-closed*:

$\mathbf{0}_{\text{Pring } R \ I} \in \text{carrier } (\text{Pring } R \ I)$

$\langle \text{proof} \rangle$

**lemma** *Pring-var-closed*:

**assumes**  $i \in I$

**shows**  $\text{var-to-IP } R \ i \in \text{carrier } (\text{Pring } R \ I)$

$\langle \text{proof} \rangle$

Properties of addition:

**lemma** *Pring-add-assoc*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $b \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $c \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \oplus_{\text{Pring } R \ I} (b \oplus_{\text{Pring } R \ I} c) = (a \oplus_{\text{Pring } R \ I} b) \oplus_{\text{Pring } R \ I} c$

*<proof>*

**lemma** *Pring-add-comm*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $b \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \oplus_{\text{Pring } R \ I} b = b \oplus_{\text{Pring } R \ I} a$

*<proof>*

**lemma** *Pring-add-zero*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \oplus_{\text{Pring } R \ I} \mathbf{0}_{\text{Pring } R \ I} = a$

$\mathbf{0}_{\text{Pring } R \ I} \oplus_{\text{Pring } R \ I} a = a$

*<proof>*

Properties of multiplication

**lemma** *Pring-mult-assoc*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $b \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $c \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \otimes_{\text{Pring } R \ I} (b \otimes_{\text{Pring } R \ I} c) = (a \otimes_{\text{Pring } R \ I} b) \otimes_{\text{Pring } R \ I} c$

*<proof>*

**lemma** *Pring-mult-comm*:

**assumes**  $\text{cring } R$

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $b \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \otimes_{\text{Pring } R \ I} b = b \otimes_{\text{Pring } R \ I} a$

*<proof>*

**lemma** *Pring-mult-one*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $a \otimes_{\text{Pring } R \ I} \mathbf{1}_{\text{Pring } R \ I} = a$

*<proof>*

**lemma** *Pring-mult-one'*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**shows**  $\mathbf{1}_{\text{Pring } R \ I} \otimes_{\text{Pring } R \ I} a = a$

*<proof>*

Distributive laws

**lemma** *Pring-mult-rdistr*:

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$

**assumes**  $b \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $c \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $a \otimes_{\text{Pring } R \ I} (b \oplus_{\text{Pring } R \ I} c) = (a \otimes_{\text{Pring } R \ I} b) \oplus_{\text{Pring } R \ I} (a \otimes_{\text{Pring } R \ I} c)$   
 <proof>

**lemma** *Pring-mult-ldistr:*

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $b \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $c \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $(b \oplus_{\text{Pring } R \ I} c) \otimes_{\text{Pring } R \ I} a = (b \otimes_{\text{Pring } R \ I} a) \oplus_{\text{Pring } R \ I} (c \otimes_{\text{Pring } R \ I} a)$   
 <proof>

Properties of subtraction:

**lemma** *Pring-uminus:*

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $P\text{-ring-uminus } R \ a \in \text{carrier } (\text{Pring } R \ I)$   
 <proof>

**lemma** *Pring-subtract:*

**assumes**  $a \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $P\text{-ring-uminus } R \ a \oplus_{\text{Pring } R \ I} a = \mathbf{0}_{\text{Pring } R \ I}$   
 $a \oplus_{\text{Pring } R \ I} P\text{-ring-uminus } R \ a = \mathbf{0}_{\text{Pring } R \ I}$   
 <proof>

Pring R I is a ring

**lemma** *Pring-is-abelian-group:*

**shows**  $\text{abelian-group } (\text{Pring } R \ I)$   
 <proof>

**lemma** *Pring-is-monoid:*

$\text{Group.monoid } (\text{Pring } R \ I)$   
 <proof>

**lemma** *Pring-is-ring:*

**shows**  $\text{ring } (\text{Pring } R \ I)$   
 <proof>

**lemma** *Pring-is-cring:*

**assumes**  $\text{cring } R$   
**shows**  $\text{cring } (\text{Pring } R \ I)$   
 <proof>

**lemma** *Pring-a-inv:*

**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $\ominus_{\text{Pring } R \ I} P = P\text{-ring-uminus } R \ P$   
 <proof>

end

## 2.10 Defining the R-Algebra of Indexed Polynomials

context *cring*

begin

**lemma** *Pring-smult-cfs*:

assumes  $a \in \text{carrier } R$

assumes  $P \in \text{carrier } (\text{Pring } R \ I)$

shows  $(a \odot_{\text{Pring } R \ I} P) \ m = a \otimes (P \ m)$

*<proof>*

**lemma** *Pring-smult-closed*:

$\bigwedge a \ x. [\![ a \in \text{carrier } R; x \in \text{carrier } (\text{Pring } R \ I) ]\!] \implies a \odot_{(\text{Pring } R \ I)} x \in \text{carrier } (\text{Pring } R \ I)$

*<proof>*

**lemma** *Pring-smult-l-distr*:

$!!a \ b \ x. [\![ a \in \text{carrier } R; b \in \text{carrier } R; x \in \text{carrier } (\text{Pring } R \ I) ]\!] \implies$

$(a \oplus b) \odot_{(\text{Pring } R \ I)} x = (a \odot_{(\text{Pring } R \ I)} x) \oplus_{(\text{Pring } R \ I)} (b \odot_{(\text{Pring } R \ I)} x)$

*<proof>*

**lemma** *Pring-smult-r-distr*:

$!!a \ x \ y. [\![ a \in \text{carrier } R; x \in \text{carrier } (\text{Pring } R \ I); y \in \text{carrier } (\text{Pring } R \ I) ]\!] \implies$

$a \odot_{(\text{Pring } R \ I)} (x \oplus_{(\text{Pring } R \ I)} y) = (a \odot_{(\text{Pring } R \ I)} x) \oplus_{(\text{Pring } R \ I)} (a \odot_{(\text{Pring } R \ I)} y)$

*<proof>*

**lemma** *Pring-smult-assoc1*:

$!!a \ b \ x. [\![ a \in \text{carrier } R; b \in \text{carrier } R; x \in \text{carrier } (\text{Pring } R \ I) ]\!] \implies$

$(a \otimes b) \odot_{\text{Pring } R \ I} x = a \odot_{\text{Pring } R \ I} (b \odot_{\text{Pring } R \ I} x)$

*<proof>*

**lemma** *Pring-smult-one*:

$!!x. x \in \text{carrier } (\text{Pring } R \ I) \implies (\text{one } R) \odot_{\text{Pring } R \ I} x = x$

*<proof>*

**lemma** *Pring-smult-assoc2*:

$!!a \ x \ y. [\![ a \in \text{carrier } R; x \in \text{carrier } (\text{Pring } R \ I); y \in \text{carrier } (\text{Pring } R \ I) ]\!] \implies$

$(a \odot_{\text{Pring } R \ I} x) \otimes_{\text{Pring } R \ I} y = a \odot_{\text{Pring } R \ I} (x \otimes_{\text{Pring } R \ I} y)$

*<proof>*

**lemma** *Pring-algebra*:

*algebra R (Pring R I)*  
 ⟨proof⟩

**end**

## 2.11 Evaluation of Polynomials and Subring Structure

In this section the aim is to define the evaluation of a polynomial over its base ring. We define both total evaluation of a polynomial at all variables, and partial evaluation at only a subset of variables. The basic input for evaluation is a variable assignment function mapping variables to ring elements. Once we can evaluate a polynomial  $P$  in variables  $I$  over a ring  $R$  at an assignment  $f : I \rightarrow R$ , this can be generalized to evaluation of  $P$  in some other ring  $S$ , given a variable assignment  $f : I \rightarrow S$  and a ring homomorphism  $\phi : R \rightarrow S$ . We chose to define this by simply applying  $\phi$  to the coefficients of  $P$ , and then using the first evaluation function over  $S$ . This could also have been done the other way around: define general polynomial evaluation over any ring, given a ring hom  $\phi$ , and then defining evaluation over the base ring  $R$  as the specialization of this function to the case there  $\phi = id_R$ .

**definition** *remove-monom* ::

*('a,'b) ring-scheme  $\Rightarrow$  'c monomial  $\Rightarrow$  ('a, 'c) mvar-poly  $\Rightarrow$  ('a, 'c) mvar-poly*

**where**

*remove-monom R m P = ring.indexed-padd R P (poly-scalar-mult R ( $\ominus_R$  P m)*  
*(mset-to-IP R m))*

**context** *cring*

**begin**

**lemma** *remove-monom-alt-def:*

**assumes**  $P \in \text{Pring-set } R \ I$

**shows** *remove-monom R m P n = (if n = m then 0 else P n)*

⟨proof⟩

**lemma** *remove-monom-zero:*

**assumes**  $m \notin \text{monomials-of } R \ P$

**assumes**  $P \in \text{Pring-set } R \ I$

**shows** *remove-monom R m P = P*

⟨proof⟩

**lemma** *remove-monom-closed:*

**assumes**  $P \in \text{Pring-set } R \ I$

**shows** *remove-monom R m P  $\in$  Pring-set R I*

⟨proof⟩

**lemma** *remove-monom-monomials:*

**assumes**  $P \in \text{Pring-set } R \ I$

**shows**  $\text{monomials-of } R \ (\text{remove-monom } R \ m \ P) = \text{monomials-of } R \ P - \{m\}$   
 $\langle \text{proof} \rangle$

The additive decomposition of a polynomial by a monomial

**lemma** *remove-monom-eq:*

**assumes**  $P \in \text{Pring-set } R \ I$

**shows**  $P = (\text{remove-monom } R \ a \ P) \oplus \text{poly-scalar-mult } R \ (P \ a) \ (\text{mset-to-IP } R \ a)$   
 $\langle \text{proof} \rangle$

**lemma** *remove-monom-restrict-poly-to-monom-set:*

**assumes**  $P \in \text{Pring-set } R \ I$

**assumes**  $\text{monomials-of } R \ P = \text{insert } a \ M$

**assumes**  $a \notin M$

**shows**  $(\text{remove-monom } R \ a \ P) = \text{restrict-poly-to-monom-set } R \ P \ M$   
 $\langle \text{proof} \rangle$

**end**

### 2.11.1 Nesting of Polynomial Rings According to Nesting of Index Sets

**lemma**(*in ring*) *Pring-carrier-subset:*

**assumes**  $J \subseteq I$

**shows**  $(\text{Pring-set } R \ J) \subseteq (\text{Pring-set } R \ I)$   
 $\langle \text{proof} \rangle$

**lemma**(*in cring*) *Pring-set-restrict-induct:*

**shows**  $\text{finite } S \implies \forall P. \text{monomials-of } R \ P = S \wedge P \in \text{Pring-set } R \ I \wedge (\forall m \in \text{monomials-of } R \ P. \text{set-mset } m \subseteq J) \longrightarrow P \in \text{Pring-set } R \ J$   
 $\langle \text{proof} \rangle$

**lemma**(*in cring*) *Pring-set-restrict:*

**assumes**  $P \in \text{Pring-set } R \ I$

**assumes**  $(\bigwedge m. m \in \text{monomials-of } R \ P \implies \text{set-mset } m \subseteq J)$

**shows**  $P \in \text{Pring-set } R \ J$   
 $\langle \text{proof} \rangle$

**lemma**(*in ring*) *Pring-mult-eq:*

**fixes**  $I:: 'c \ \text{set}$

**fixes**  $J:: 'c \ \text{set}$

**shows**  $(\otimes_{\text{Pring}} R \ I) = (\otimes_{\text{Pring}} R \ J)$   
 $\langle \text{proof} \rangle$

**lemma**(*in ring*) *Pring-add-eq:*

**fixes**  $I:: 'c \ \text{set}$

**fixes**  $J:: 'c \ \text{set}$

**shows**  $(\oplus_{Pring\ R\ I}) = (\oplus_{Pring\ R\ J})$   
 $\langle proof \rangle$

**lemma**(in *ring*) *Pring-one-eq*:  
**fixes**  $I:: 'c\ set$   
**fixes**  $J:: 'c\ set$   
**shows**  $(\mathbf{1}_{Pring\ R\ I}) = (\mathbf{1}_{Pring\ R\ J})$   
 $\langle proof \rangle$

**lemma**(in *ring*) *Pring-zero-eq*:  
**fixes**  $I:: 'c\ set$   
**fixes**  $J:: 'c\ set$   
**shows**  $(\mathbf{0}_{Pring\ R\ I}) = (\mathbf{0}_{Pring\ R\ J})$   
 $\langle proof \rangle$

**lemma**(in *ring*) *index-subset-Pring-subring*:  
**assumes**  $J \subseteq I$   
**shows** *subring* (*carrier* (*Pring R J*)) (*Pring R I*)  
 $\langle proof \rangle$

### 2.11.2 Inclusion Maps

**definition** *Pring-inc* ::  $('a, 'c)\ mvar\ poly \Rightarrow ('a, 'c)\ mvar\ poly$  **where**  
*Pring-inc*  $\equiv (\lambda P. P)$

**lemma**(in *ring*) *Princ-inc-is-ring-hom*:  
**assumes**  $J \subseteq I$   
**shows** *ring-hom-ring* (*Pring R J*) (*Pring R I*) *Pring-inc*  
 $\langle proof \rangle$

### 2.11.3 Restricting a Multiset to a Subset of Variables

**definition** *restrict-to-indices* ::  $'c\ monomial \Rightarrow 'c\ set \Rightarrow 'c\ monomial$  **where**  
*restrict-to-indices*  $m\ S = filter\ mset\ (\lambda i. i \in S)\ m$

**lemma** *restrict-to-indicesE*:  
**assumes**  $i \in \#\ restrict\ to\ indices\ m\ S$   
**shows**  $i \in S$   
 $\langle proof \rangle$

**lemma** *restrict-to-indicesI[simp]*:  
**assumes**  $i \in \#\ m$   
**assumes**  $i \in S$   
**shows**  $i \in \#\ restrict\ to\ indices\ m\ S$   
 $\langle proof \rangle$

**lemma** *restrict-to-indices-not-in[simp]*:  
**assumes**  $i \in \#\ m$   
**assumes**  $i \notin S$

**shows**  $i \notin \# \text{ restrict-to-indices } m \ S$   
 ⟨proof⟩

**lemma** *restrict-to-indices-submultiset[simp]*:  
 $\text{restrict-to-indices } m \ S \subseteq \# \ m$   
 ⟨proof⟩

**lemma** *restrict-to-indices-add-element*:  
 $\text{restrict-to-indices } (\text{add-mset } x \ m) \ S = (\text{if } x \in S \text{ then } (\text{add-mset } x \ (\text{restrict-to-indices } m \ S)) \text{ else } (\text{restrict-to-indices } m \ S))$   
 ⟨proof⟩

**lemma** *restrict-to-indices-count[simp]*:  
 $\text{count } (\text{restrict-to-indices } m \ S) \ i = (\text{if } (i \in S) \text{ then } (\text{count } m \ i) \text{ else } 0)$   
 ⟨proof⟩

**lemma** *restrict-to-indices-subset*:  
 $\text{restrict-to-indices } m \ S = \text{restrict-to-indices } m \ (\text{set-mset } m \cap S)$   
 ⟨proof⟩

`Restrict_to_indices` only depends on the intersection of the index set with the set of indices in  $m$ :

**lemma** *restrict-to-indices-subset'*:  
**assumes**  $(\text{set-mset } m) \cap S = (\text{set-mset } m) \cap S'$   
**shows**  $\text{restrict-to-indices } m \ S = \text{restrict-to-indices } m \ S'$   
 ⟨proof⟩

**lemma** *mset-add-plus*:  
**assumes**  $m = n + k$   
**shows**  $\text{add-mset } x \ m = (\text{add-mset } x \ n) + k$   
 ⟨proof⟩

Restricting to  $S$  and the complement of  $S$  partitions  $m$ :

**lemma** *restrict-to-indices-decomp*:  
 $m = (\text{restrict-to-indices } m \ S) + (\text{restrict-to-indices } m \ ((\text{set-mset } m) - S))$   
 ⟨proof⟩

**definition** *remove-indices* ::  $'c \ \text{monomial} \Rightarrow 'c \ \text{set} \Rightarrow 'c \ \text{monomial}$  **where**  
 $\text{remove-indices } m \ S = (\text{restrict-to-indices } m \ (\text{set-mset } m - S))$

**lemma** *remove-indices-decomp*:  
 $m = (\text{restrict-to-indices } m \ S) + (\text{remove-indices } m \ S)$   
 ⟨proof⟩

**lemma** *remove-indices-indices[simp]*:  
**assumes**  $\text{set-mset } m \subseteq I$   
**shows**  $\text{set-mset } (\text{remove-indices } m \ S) \subseteq I - S$   
 ⟨proof⟩

#### 2.11.4 Total evaluation of a monomial

We define total evaluation of a monomial first, and then define the partial evaluation of a monomial in terms of this.

**abbreviation** *(input) closed-fun where*  
*closed-fun*  $R$   $g \equiv g \in UNIV \rightarrow carrier\ R$

**definition** *monom-eval*  $:: ('a, 'b)$  ring-scheme  $\Rightarrow 'c$  monomial  $\Rightarrow ('c \Rightarrow 'a) \Rightarrow 'a$   
**where**  
*monom-eval*  $R$  ( $m :: 'c$  monomial)  $g = fold-mset (\lambda x . \lambda y. \text{if } y \in carrier\ R \text{ then } (g\ x) \otimes_R y \text{ else } \mathbf{0}_R) \mathbf{1}_R\ m$

**context** *cring*  
**begin**

**lemma** *closed-fun-simp*:  
**assumes** *closed-fun*  $R$   $g$   
**shows**  $g\ n \in carrier\ R$   
 $\langle proof \rangle$

**lemma** *closed-funI*:  
**assumes**  $\bigwedge x. g\ x \in carrier\ R$   
**shows** *closed-fun*  $R$   $g$   
 $\langle proof \rangle$

The following are necessary technical lemmas to prove properties of about folds over multisets:

**lemma** *monom-eval-comp-fun*:  
**fixes**  $g :: 'c \Rightarrow 'a$   
**assumes** *closed-fun*  $R$   $g$   
**shows** *comp-fun-commute*  $(\lambda x . \lambda y. \text{if } y \in carrier\ R \text{ then } (g\ x) \otimes y \text{ else } \mathbf{0})$   
 $\langle proof \rangle$

**lemma** *monom-eval-car*:  
**assumes** *closed-fun*  $R$   $g$   
**shows** *monom-eval*  $R$  ( $m :: 'c$  monomial)  $g \in carrier\ R$   
 $\langle proof \rangle$

Formula for recursive (total) evaluation of a monomial:

**lemma** *monom-eval-add*:  
**assumes** *closed-fun*  $R$   $g$   
**shows** *monom-eval*  $R$  (*add-mset*  $x$   $M$ )  $g = (g\ x) \otimes (monom-eval\ R\ M\ g)$   
 $\langle proof \rangle$

**end**

This function maps a polynomial  $P$  to the set of monomials in  $P$  which, after evaluating all variables in the set  $S$  to values in the ring  $R$ , reduce to the monomial  $n$ .

**definition** *monomials-reducing-to* ::

$(\text{'a, 'b})$  ring-scheme  $\Rightarrow$   $\text{'c}$  monomial  $\Rightarrow$   $(\text{'a, 'c})$  mvar-poly  $\Rightarrow$   $\text{'c}$  set  $\Rightarrow$   $(\text{'c}$  monomial) set **where**

*monomials-reducing-to*  $R$   $n$   $P$   $S = \{m \in \text{monomials-of } R$   $P. \text{remove-indices } m$   $S = n\}$

**lemma** *monomials-reducing-to-subset*[simp]:

*monomials-reducing-to*  $R$   $n$   $P$   $s \subseteq \text{monomials-of } R$   $P$   
 $\langle$ proof $\rangle$

**context** *cring*  
**begin**

**lemma** *monomials-reducing-to-finite*:

**assumes**  $P \in \text{Pring-set } R$   $I$   
**shows** *finite* (*monomials-reducing-to*  $R$   $n$   $P$   $s$ )  
 $\langle$ proof $\rangle$

**lemma** *monomials-reducing-to-disjoint*:

**assumes**  $n1 \neq n2$   
**shows** *monomials-reducing-to*  $R$   $n1$   $P$   $S \cap \text{monomials-reducing-to } R$   $n2$   $P$   $S = \{\}$   
 $\langle$ proof $\rangle$

**lemma** *monomials-reducing-to-submset*:

**assumes**  $n \subset \# m$   
**shows**  $n \notin \text{monomials-reducing-to } R$   $m$   $P$   $S$   
 $\langle$ proof $\rangle$

**end**

### 2.11.5 Partial Evaluation of a Polynomial

This function takes as input a set  $S$  of variables, an evaluation function  $g$ , and a polynomial to evaluate  $P$ . The output is a polynomial which is the result of evaluating the variables from the set  $S$  which occur in  $P$ , according to the evaluation function  $g$ .

**definition** *poly-eval* ::

$(\text{'a, 'b})$  ring-scheme  $\Rightarrow$   $\text{'c}$  set  $\Rightarrow$   $(\text{'c} \Rightarrow \text{'a}) \Rightarrow$   $(\text{'a, 'c})$  mvar-poly  $\Rightarrow$   $(\text{'a, 'c})$  mvar-poly **where**  
*poly-eval*  $R$   $S$   $g$   $P$   $m = (\text{finsum } R (\lambda n. \text{monom-eval } R (\text{restrict-to-indices } n$   $S) g \otimes_R (P$   $n)) (\text{monomials-reducing-to } R$   $m$   $P$   $S))$

**context** *cring*  
**begin**

**lemma** *finsum-singleton*:

**assumes**  $S = \{s\}$

**assumes**  $f s \in \text{carrier } R$   
**shows**  $\text{finsum } R f S = f s$   
 ⟨proof⟩

**lemma** *poly-eval-constant*:  
**assumes**  $k \in \text{carrier } R$   
**shows**  $\text{poly-eval } R S g (\text{indexed-const } k) = (\text{indexed-const } k)$   
 ⟨proof⟩

**lemma** *finsum-partition*:  
**assumes** *finite*  $S$   
**assumes**  $f \in S \rightarrow \text{carrier } R$   
**assumes**  $T \subseteq S$   
**shows**  $\text{finsum } R f S = \text{finsum } R f T \oplus \text{finsum } R f (S - T)$   
 ⟨proof⟩

**lemma** *finsum-eq-partition*:  
**assumes** *finite*  $S$   
**assumes**  $f \in S \rightarrow \text{carrier } R$   
**assumes**  $T \subseteq S$   
**assumes**  $\bigwedge x. x \in S - T \implies f x = \mathbf{0}$   
**shows**  $\text{finsum } R f S = \text{finsum } R f T$   
 ⟨proof⟩

**lemma** *poly-eval-scalar-mult*:  
**assumes**  $k \in \text{carrier } R$   
**assumes** *closed-fun*  $R g$   
**assumes**  $P \in \text{Pring-set } R I$   
**shows**  $\text{poly-eval } R S g (\text{poly-scalar-mult } R k P) =$   
 $(\text{poly-scalar-mult } R k (\text{poly-eval } R S g P))$   
 ⟨proof⟩

**lemma** *poly-eval-monomial*:  
**assumes** *closed-fun*  $R g$   
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**shows**  $\text{poly-eval } R S g (\text{mset-to-IP } R m) =$   
 $\text{poly-scalar-mult } R (\text{monom-eval } R (\text{restrict-to-indices } m S) g)$   
 $(\text{mset-to-IP } R (\text{remove-indices } m S))$   
 ⟨proof⟩

**lemma**(in *cring*) *poly-eval-monomial-closed*:  
**assumes** *closed-fun*  $R g$   
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**assumes** *set-mset*  $m \subseteq I$   
**shows**  $\text{poly-eval } R S g (\text{mset-to-IP } R m) \in \text{Pring-set } R (I - S)$   
 ⟨proof⟩

**lemma** *poly-scalar-mult-iter*:

**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $k \in \text{carrier } R$   
**assumes**  $n \in \text{carrier } R$   
**shows**  $\text{poly-scalar-mult } R \ k \ (\text{poly-scalar-mult } R \ n \ P) = \text{poly-scalar-mult } R \ (k \otimes n) \ P$   
 $\langle \text{proof} \rangle$

**lemma** *poly-scalar-mult-comm:*

**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{carrier } R$   
**shows**  $\text{poly-scalar-mult } R \ a \ (\text{poly-scalar-mult } R \ b \ P) = \text{poly-scalar-mult } R \ b \ (\text{poly-scalar-mult } R \ a \ P)$   
 $\langle \text{proof} \rangle$

**lemma** *poly-eval-monomial-term:*

**assumes** *closed-fun*  $R \ g$   
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**assumes** *set-mset*  $m \subseteq I$   
**assumes**  $k \in \text{carrier } R$   
**shows**  $\text{poly-eval } R \ S \ g \ (\text{poly-scalar-mult } R \ k \ (\text{mset-to-IP } R \ m)) = \text{poly-scalar-mult } R \ (k \otimes (\text{monom-eval } R \ (\text{restrict-to-indices } m \ S) \ g)) \ (\text{mset-to-IP } R \ (\text{remove-indices } m \ S))$   
 $\langle \text{proof} \rangle$

**lemma** *poly-eval-monomial-term-closed:*

**assumes** *closed-fun*  $R \ g$   
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**assumes** *set-mset*  $m \subseteq I$   
**assumes**  $k \in \text{carrier } R$   
**shows**  $\text{poly-eval } R \ S \ g \ (\text{poly-scalar-mult } R \ k \ (\text{mset-to-IP } R \ m)) \in \text{Pring-set } R \ (I - S)$   
 $\langle \text{proof} \rangle$

**lemma** *finsum-split:*

**assumes** *finite*  $S$   
**assumes**  $f \in S \rightarrow \text{carrier } R$   
**assumes**  $g \in S \rightarrow \text{carrier } R$   
**assumes**  $k \in \text{carrier } R$   
**assumes**  $c \in S$   
**assumes**  $\bigwedge s. s \in S \wedge s \neq c \implies f \ s = g \ s$   
**assumes**  $g \ c = f \ c \oplus k$   
**shows**  $\text{finsum } R \ g \ S = k \oplus \text{finsum } R \ f \ S$   
 $\langle \text{proof} \rangle$

**lemma** *poly-monom-induction:*

**assumes**  $P \ (\text{indexed-const } \mathbf{0})$

**assumes**  $\bigwedge m k. \text{set-mset } m \subseteq I \wedge k \in \text{carrier } R \implies P (\text{poly-scalar-mult } R k (\text{mset-to-IP } R m))$   
**assumes**  $\bigwedge Q m k. Q \in \text{Pring-set } R I \wedge (P Q) \wedge \text{set-mset } m \subseteq I \wedge k \in \text{carrier } R \implies P (Q \oplus (\text{poly-scalar-mult } R k (\text{mset-to-IP } R m)))$   
**shows**  $\bigwedge Q. Q \in \text{Pring-set } R I \implies P Q$   
 $\langle \text{proof} \rangle$

**lemma** *Pring-car-induct:*

**assumes**  $q \in \text{carrier } (\text{Pring } R I)$   
**assumes**  $P \mathbf{0}_{\text{Pring } R I}$   
**assumes**  $\bigwedge m k. \text{set-mset } m \subseteq I \wedge k \in \text{carrier } R \implies P (k \odot_{\text{Pring } R I} (\text{mset-to-IP } R m))$   
**assumes**  $\bigwedge Q m k. Q \in \text{carrier } (\text{Pring } R I) \wedge (P Q) \wedge \text{set-mset } m \subseteq I \wedge k \in \text{carrier } R \implies$   
 $P (Q \oplus (k \odot_{\text{Pring } R I} (\text{mset-to-IP } R m)))$   
**shows**  $P q$   
 $\langle \text{proof} \rangle$

**lemma** *poly-monom-induction2:*

**assumes**  $P (\text{indexed-const } \mathbf{0})$   
**assumes**  $\bigwedge m k. \text{set-mset } m \subseteq I \wedge k \in \text{carrier } R \implies P (\text{poly-scalar-mult } R k (\text{mset-to-IP } R m))$   
**assumes**  $\bigwedge Q m k. Q \in \text{Pring-set } R I \wedge (P Q) \wedge \text{set-mset } m \subseteq I \wedge k \in \text{carrier } R \implies P (Q \oplus (\text{poly-scalar-mult } R k (\text{mset-to-IP } R m)))$   
**assumes**  $Q \in \text{Pring-set } R I$   
**shows**  $P Q$   
 $\langle \text{proof} \rangle$

**lemma** *poly-monom-induction3:*

**assumes**  $Q \in \text{Pring-set } R I$   
**assumes**  $P (\text{indexed-const } \mathbf{0})$   
**assumes**  $\bigwedge m k. \text{set-mset } m \subseteq I \wedge k \in \text{carrier } R \implies P (\text{poly-scalar-mult } R k (\text{mset-to-IP } R m))$   
**assumes**  $\bigwedge p q. p \in \text{Pring-set } R I \implies (P p) \implies q \in \text{Pring-set } R I \implies (P q) \implies P (p \oplus q)$   
**shows**  $P Q$   
 $\langle \text{proof} \rangle$

**lemma** *Pring-car-induct':*

**assumes**  $Q \in \text{carrier } (\text{Pring } R I)$   
**assumes**  $P \mathbf{0}_{\text{Pring } R I}$   
**assumes**  $\bigwedge m k. \text{set-mset } m \subseteq I \wedge k \in \text{carrier } R \implies P (k \odot_{\text{Pring } R I} (\text{mset-to-IP } R m))$   
**assumes**  $\bigwedge p q. p \in \text{carrier } (\text{Pring } R I) \implies (P p) \implies q \in \text{carrier } (\text{Pring } R I) \implies (P q) \implies P (p \oplus_{\text{Pring } R I} q)$   
**shows**  $P Q$   
 $\langle \text{proof} \rangle$

**lemma** *poly-eval-mono:*

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $\text{closed-fun } R \ g$   
**assumes**  $\text{finite } F$   
**assumes**  $\text{monomials-reducing-to } R \ m \ P \ S \subseteq F$   
**assumes**  $\bigwedge n. n \in F \implies \text{remove-indices } n \ S = m$   
**shows**  $\text{poly-eval } R \ S \ g \ P \ m = (\bigoplus_{n \in F. \text{monom-eval } R \ (\text{restrict-to-indices } n \ S)}$   
 $g \otimes P \ n)$   
 $\langle \text{proof} \rangle$

**lemma** *finsum-group*:

**assumes**  $\bigwedge n. f \ n \in \text{carrier } R$   
**assumes**  $\bigwedge n. g \ n \in \text{carrier } R$   
**shows**  $\text{finite } S \implies \text{finsum } R \ f \ S \oplus \text{finsum } R \ g \ S = \text{finsum } R \ (\lambda n. f \ n \oplus g \ n) \ S$   
 $\langle \text{proof} \rangle$

**lemma** *poly-eval-add*:

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $Q \in \text{Pring-set } R \ I$   
**assumes**  $\text{closed-fun } R \ g$   
**shows**  $\text{poly-eval } R \ S \ g \ (P \oplus Q) = \text{poly-eval } R \ S \ g \ P \oplus \text{poly-eval } R \ S \ g \ Q$   
 $\langle \text{proof} \rangle$

**lemma** *poly-eval-Pring-add*:

**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $Q \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $\text{closed-fun } R \ g$   
**shows**  $\text{poly-eval } R \ S \ g \ (P \oplus_{\text{Pring } R \ I} Q) = \text{poly-eval } R \ S \ g \ P \oplus_{\text{Pring } R \ I}$   
 $\text{poly-eval } R \ S \ g \ Q$   
 $\langle \text{proof} \rangle$

Closure of partial evaluation maps:

**lemma**(in *cring*) *poly-eval-closed*:

**assumes**  $\text{closed-fun } R \ g$   
**assumes**  $P \in \text{Pring-set } R \ I$   
**shows**  $\text{poly-eval } R \ S \ g \ P \in \text{Pring-set } R \ (I - S)$   
 $\langle \text{proof} \rangle$

**lemma** *poly-scalar-mult-indexed-pmult*:

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $k \in \text{carrier } R$   
**shows**  $\text{poly-scalar-mult } R \ k \ (P \otimes i) = (\text{poly-scalar-mult } R \ k \ P) \otimes i$   
 $\langle \text{proof} \rangle$

**lemma** *remove-indices-add-mset*:

**assumes**  $i \notin S$   
**shows**  $\text{remove-indices } (\text{add-mset } i \ m) \ S = \text{add-mset } i \ (\text{remove-indices } m \ S)$   
 $\langle \text{proof} \rangle$

**lemma** *poly-eval-monom-insert*:

**assumes** *closed-fun R g*  
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**assumes**  $i \in S$   
**shows**  $\text{poly-eval } R \ S \ g \ (\text{mset-to-IP } R \ (\text{add-mset } i \ m))$   
 $= \text{poly-scalar-mult } R \ (g \ i)$   
 $(\text{poly-eval } R \ S \ g \ (\text{mset-to-IP } R \ m))$   
 ⟨*proof*⟩

**lemma** *poly-eval-monom-insert'*:  
**assumes** *closed-fun R g*  
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**assumes**  $i \notin S$   
**shows**  $\text{poly-eval } R \ S \ g \ (\text{mset-to-IP } R \ (\text{add-mset } i \ m))$   
 $= (\text{poly-eval } R \ S \ g \ (\text{mset-to-IP } R \ m)) \otimes i$   
 ⟨*proof*⟩

**lemma** *poly-eval-indexed-pmult-monomial*:  
**assumes** *closed-fun R g*  
**assumes**  $k \in \text{carrier } R$   
**assumes**  $i \in S$   
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**shows**  $\text{poly-eval } R \ S \ g \ (\text{poly-scalar-mult } R \ k \ (\text{mset-to-IP } R \ m) \otimes i) =$   
 $\text{poly-scalar-mult } R \ (g \ i) \ (\text{poly-eval } R \ S \ g \ (\text{poly-scalar-mult } R \ k \ (\text{mset-to-IP}$   
 $R \ m)))$   
 ⟨*proof*⟩

**lemma** *poly-eval-indexed-pmult-monomial'*:  
**assumes** *closed-fun R g*  
**assumes**  $k \in \text{carrier } R$   
**assumes**  $i \notin S$   
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**shows**  $\text{poly-eval } R \ S \ g \ (\text{poly-scalar-mult } R \ k \ (\text{mset-to-IP } R \ m) \otimes i) =$   
 $(\text{poly-eval } R \ S \ g \ (\text{poly-scalar-mult } R \ k \ (\text{mset-to-IP } R \ m))) \otimes i$   
 ⟨*proof*⟩

**lemma** *indexed-pmult-add*:  
**assumes**  $p \in \text{Pring-set } R \ I$   
**assumes**  $q \in \text{Pring-set } R \ I$   
**shows**  $p \oplus q \otimes i = (p \otimes i) \oplus (q \otimes i)$   
 ⟨*proof*⟩

**lemma** *poly-eval-indexed-pmult*:  
**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes** *closed-fun R g*  
**shows**  $\text{poly-eval } R \ S \ g \ (P \otimes i) = (\text{if } i \in S \text{ then } \text{poly-scalar-mult } R \ (g \ i)$   
 $(\text{poly-eval } R \ S \ g \ P) \text{ else } (\text{poly-eval } R \ S \ g \ P) \otimes i)$   
 ⟨*proof*⟩

**lemma** *poly-eval-index*:

**assumes**  $1 \neq 0$   
**assumes**  $\text{closed-fun } R \ g$   
**shows**  $\text{poly-eval } R \ S \ g \ (\text{mset-to-IP } R \ \{\#i\#}) = (\text{if } i \in S \text{ then } (\text{indexed-const } (g \ i)) \text{ else } \text{mset-to-IP } R \ \{\#i\#})$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{poly-eval-indexed-pmult}'$ :

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $\text{closed-fun } R \ g$   
**assumes**  $i \in I$   
**shows**  $\text{poly-eval } R \ S \ g \ (P \otimes_p (\text{mset-to-IP } R \ \{\#i\#})) = \text{poly-eval } R \ S \ g \ P \otimes_p \text{poly-eval } R \ S \ g \ (\text{mset-to-IP } R \ \{\#i\#})$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{poly-eval-monom-mult}$ :

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $\text{closed-fun } R \ g$   
**shows**  $\text{poly-eval } R \ S \ g \ (P \otimes_p (\text{mset-to-IP } R \ m)) = \text{poly-eval } R \ S \ g \ P \otimes_p \text{poly-eval } R \ S \ g \ (\text{mset-to-IP } R \ m)$   
 $\langle \text{proof} \rangle$

**abbreviation**  $\text{mon-term } (Mt) \ \text{where}$

$Mt \ k \ m \equiv \text{poly-scalar-mult } R \ k \ (\text{mset-to-IP } R \ m)$

**lemma**  $\text{poly-eval-monom-term-mult}$ :

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $\text{closed-fun } R \ g$   
**assumes**  $k \in \text{carrier } R$   
**shows**  $\text{poly-eval } R \ S \ g \ (P \otimes_p (Mt \ k \ m)) = \text{poly-eval } R \ S \ g \ P \otimes_p \text{poly-eval } R \ S \ g \ (Mt \ k \ m)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{poly-eval-mult}$ :

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $Q \in \text{Pring-set } R \ I$   
**assumes**  $\text{closed-fun } R \ g$   
**shows**  $\text{poly-eval } R \ S \ g \ (P \otimes_p Q) = \text{poly-eval } R \ S \ g \ P \otimes_p \text{poly-eval } R \ S \ g \ Q$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{poly-eval-Pring-mult}$ :

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $Q \in \text{Pring-set } R \ I$   
**assumes**  $\text{closed-fun } R \ g$   
**shows**  $\text{poly-eval } R \ S \ g \ (P \otimes_{\text{Pring } R \ I} Q) = \text{poly-eval } R \ S \ g \ P \otimes_{\text{Pring } R \ I} \text{poly-eval } R \ S \ g \ Q$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{poly-eval-smult}$ :

**assumes**  $P \in \text{Pring-set } R \ I$

**assumes**  $a \in \text{carrier } R$   
**assumes**  $\text{closed-fun } R \ g$   
**shows**  $\text{poly-eval } R \ S \ g \ (a \odot_{\text{Pring } R \ I} P) = a \odot_{\text{Pring } R \ I} \text{poly-eval } R \ S \ g \ P$   
 $\langle \text{proof} \rangle$

### 2.11.6 Partial Evaluation is a Homomorphism

**lemma**  $\text{poly-eval-ring-hom}$ :  
**assumes**  $I \subseteq J$   
**assumes**  $\text{closed-fun } R \ g$   
**assumes**  $J - S \subseteq I$   
**shows**  $\text{ring-hom-ring } (\text{Pring } R \ J) \ (\text{Pring } R \ I) \ (\text{poly-eval } R \ S \ g)$   
 $\langle \text{proof} \rangle$

$\text{poly\_eval } R$  at the zero function is an inverse to the inclusion of polynomial rings

**lemma**  $\text{poly-eval-zero-function}$ :  
**assumes**  $g = (\lambda n. \mathbf{0})$   
**assumes**  $J - S = I$   
**shows**  $P \in \text{Pring-set } R \ I \implies \text{poly-eval } R \ S \ g \ P = P$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{poly-eval-eval-function-eq}$ :  
**assumes**  $\text{closed-fun } R \ g$   
**assumes**  $\text{closed-fun } R \ g'$   
**assumes**  $\text{restrict } g \ S = \text{restrict } g' \ S$   
**assumes**  $P \in \text{Pring-set } R \ I$   
**shows**  $\text{poly-eval } R \ S \ g \ P = \text{poly-eval } R \ S \ g' \ P$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{poly-eval-eval-set-eq}$ :  
**assumes**  $\text{closed-fun } R \ g$   
**assumes**  $S \cap I = S' \cap I$   
**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $\mathbf{1} \neq \mathbf{0}$   
**shows**  $\text{poly-eval } R \ S \ g \ P = \text{poly-eval } R \ S' \ g \ P$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{poly-eval-trivial}$ :  
**assumes**  $\text{closed-fun } R \ g$   
**assumes**  $P \in \text{Pring-set } R \ (I - S)$   
**shows**  $\text{poly-eval } R \ S \ g \ P = P$   
 $\langle \text{proof} \rangle$

### 2.11.7 Total Evaluation of a Polynomial

**lemma**  $\text{zero-fun-closed}$ :  
 $\text{closed-fun } R \ (\lambda n. \mathbf{0})$   
 $\langle \text{proof} \rangle$

**lemma** *deg-zero-cf-eval*:

**shows**  $P \in \text{Pring-set } R \ I \implies \text{poly-eval } R \ I \ (\lambda n. \mathbf{0}) \ P = \text{indexed-const } (P \ \{\#\})$   
*<proof>*

**lemma** *deg-zero-cf-mult*:

**assumes**  $P \in \text{Pring-set } R \ I$   
**assumes**  $Q \in \text{Pring-set } R \ I$   
**shows**  $(P \otimes_P Q) \ \{\#\} = P \ \{\#\} \otimes Q \ \{\#\}$   
*<proof>*

**definition** *deg-zero-cf* ::  $('a, 'c) \text{ mvar-poly} \Rightarrow 'a$  **where**  
*deg-zero-cf*  $P = P \ \{\#\}$

**lemma** *deg-zero-cf-ring-hom*:

**shows** *ring-hom-ring*  $(\text{Pring } R \ I) \ R \ (\text{deg-zero-cf})$   
*<proof>*

**end**

**definition** *eval-in-ring* ::

$('a, 'b) \text{ ring-scheme} \Rightarrow 'c \text{ set} \Rightarrow ('c \Rightarrow 'a) \Rightarrow ('a, 'c) \text{ mvar-poly} \Rightarrow 'a$  **where**  
*eval-in-ring*  $R \ S \ g \ P = (\text{poly-eval } R \ S \ g \ P) \ \{\#\}$

**definition** *total-eval* ::

$('a, 'b) \text{ ring-scheme} \Rightarrow ('c \Rightarrow 'a) \Rightarrow ('a, 'c) \text{ mvar-poly} \Rightarrow 'a$  **where**  
*total-eval*  $R \ g \ P = \text{eval-in-ring } R \ \text{UNIV } g \ P$

**context** *cring*

**begin**

**lemma** *eval-in-ring-ring-hom*:

**assumes** *closed-fun*  $R \ g$   
**shows** *ring-hom-ring*  $(\text{Pring } R \ I) \ R \ (\text{eval-in-ring } R \ S \ g)$   
*<proof>*

**lemma** *eval-in-ring-smult*:

**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $a \in \text{carrier } R$   
**assumes** *closed-fun*  $R \ g$   
**shows**  $\text{eval-in-ring } R \ S \ g \ (a \odot_{\text{Pring } R \ I} P) = a \otimes \text{eval-in-ring } R \ S \ g \ P$   
*<proof>*

**lemma** *total-eval-ring-hom*:

**assumes** *closed-fun*  $R \ g$   
**shows** *ring-hom-ring*  $(\text{Pring } R \ I) \ R \ (\text{total-eval } R \ g)$   
*<proof>*

**lemma** *total-eval-smult*:

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

**assumes**  $a \in \text{carrier } R$

**assumes** *closed-fun*  $R \ g$

**shows**  $\text{total-eval } R \ g \ (a \odot_{\text{Pring } R \ I} P) = a \otimes \text{total-eval } R \ g \ P$

*<proof>*

**lemma** *total-eval-const*:

**assumes**  $k \in \text{carrier } R$

**shows**  $\text{total-eval } R \ g \ (\text{indexed-const } k) = k$

*<proof>*

**lemma** *total-eval-var*:

**assumes** *closed-fun*  $R \ g$

**shows**  $(\text{total-eval } R \ g \ (\text{mset-to-IP } R \ \{\#i\})) = g \ i$

*<proof>*

**lemma** *total-eval-indexed-pmult*:

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

**assumes**  $i \in I$

**assumes** *closed-fun*  $R \ g$

**shows**  $\text{total-eval } R \ g \ (P \otimes i) = \text{total-eval } R \ g \ P \otimes_R g \ i$

*<proof>*

**lemma** *total-eval-mult*:

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

**assumes**  $Q \in \text{carrier } (\text{Pring } R \ I)$

**assumes** *closed-fun*  $R \ g$

**shows**  $\text{total-eval } R \ g \ (P \otimes_{\text{Pring } R \ I} Q) = (\text{total-eval } R \ g \ P) \otimes_R (\text{total-eval } R \ g \ Q)$

*<proof>*

**lemma** *total-eval-add*:

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

**assumes**  $Q \in \text{carrier } (\text{Pring } R \ I)$

**assumes** *closed-fun*  $R \ g$

**shows**  $\text{total-eval } R \ g \ (P \oplus_{\text{Pring } R \ I} Q) = (\text{total-eval } R \ g \ P) \oplus_R (\text{total-eval } R \ g \ Q)$

*<proof>*

**lemma** *total-eval-one*:

**assumes** *closed-fun*  $R \ g$

**shows**  $\text{total-eval } R \ g \ \mathbf{1}_{\text{Pring } R \ I} = \mathbf{1}$

*<proof>*

**lemma** *total-eval-zero*:

**assumes** *closed-fun*  $R \ g$

**shows**  $\text{total-eval } R \ g \ \mathbf{0}_{\text{Pring } R \ I} = \mathbf{0}$

*<proof>*

**lemma** *total-eval-closed*:  
**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**assumes** *closed-fun*  $R \ g$   
**shows** *total-eval*  $R \ g \ P \in \text{carrier } R$   
 $\langle \text{proof} \rangle$

## 2.12 Constructing Homomorphisms from Indexed Polynomial Rings and a Universal Property

The inclusion of  $R$  into its polynomial ring

**lemma** *indexed-const-ring-hom*:  
*ring-hom-ring*  $R \ (\text{Pring } R \ I) \ (\text{indexed-const})$   
 $\langle \text{proof} \rangle$

**lemma** *indexed-const-inj-on*:  
*inj-on*  $(\text{indexed-const}) \ (\text{carrier } R)$   
 $\langle \text{proof} \rangle$

**end**

### 2.12.1 Mapping $R[x] \rightarrow S[x]$ along a homomorphism $R \rightarrow S$

**definition** *ring-hom-to-IP-ring-hom* ::  
 $(\ 'a, \ 'e) \text{ ring-hom} \Rightarrow (\ 'a, \ 'c) \text{ mvar-poly} \Rightarrow \ 'c \text{ monomial} \Rightarrow \ 'e \text{ where}$   
*ring-hom-to-IP-ring-hom*  $\varphi \ P \ m = \varphi \ (P \ m)$

**context** *cring*  
**begin**

**lemma** *ring-hom-to-IP-ring-hom-one*:  
**assumes** *cring*  $S$   
**assumes** *ring-hom-ring*  $R \ S \ \varphi$   
**shows** *ring-hom-to-IP-ring-hom*  $\varphi \ \mathbf{1}_{\text{Pring } R \ I} = \mathbf{1}_{\text{Pring } S \ I}$   
 $\langle \text{proof} \rangle$

**lemma** *ring-hom-to-IP-ring-hom-constant*:  
**assumes** *cring*  $S$   
**assumes** *ring-hom-ring*  $R \ S \ \varphi$   
**assumes**  $a \in \text{carrier } R$   
**shows** *ring-hom-to-IP-ring-hom*  $\varphi \ ((\text{indexed-const } a):: \ 'c \text{ monomial} \Rightarrow \ 'a) =$   
*ring.indexed-const*  $S \ (\varphi \ a)$   
 $\langle \text{proof} \rangle$

**lemma** *ring-hom-to-IP-ring-hom-add*:  
**assumes** *cring*  $S$   
**assumes** *ring-hom-ring*  $R \ S \ \varphi$   
**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $Q \in \text{carrier } (\text{Pring } R \ I)$

**shows** *ring-hom-to-IP-ring-hom*  $\varphi (P \oplus_{Pring R I} Q) =$   
 $(ring-hom-to-IP-ring-hom \varphi P) \oplus_{Pring S I} (ring-hom-to-IP-ring-hom \varphi Q)$   
 $\langle proof \rangle$

**lemma** *ring-hom-to-IP-ring-hom-closed*:  
**assumes** *cring*  $S$   
**assumes** *ring-hom-ring*  $R S \varphi$   
**assumes**  $P \in carrier (Pring R I)$   
**shows** *ring-hom-to-IP-ring-hom*  $\varphi P \in carrier (Pring S I)$   
 $\langle proof \rangle$

**lemma** *ring-hom-to-IP-ring-hom-monom*:  
**assumes** *cring*  $S$   
**assumes** *ring-hom-ring*  $R S \varphi$   
**shows** *ring-hom-to-IP-ring-hom*  $\varphi (mset-to-IP R m) = mset-to-IP S m$   
 $\langle proof \rangle$

**lemma** *Pring-morphism*:  
**assumes** *cring*  $S$   
**assumes**  $\varphi \in (carrier (Pring R I)) \rightarrow (carrier S)$   
**assumes**  $\varphi \mathbf{1}_{Pring R I} = \mathbf{1}_S$   
**assumes**  $\varphi \mathbf{0}_{Pring R I} = \mathbf{0}_S$   
**assumes**  $\bigwedge P Q. P \in carrier (Pring R I) \implies Q \in carrier (Pring R I) \implies$   
 $\varphi (P \oplus_{Pring R I} Q) = (\varphi P) \oplus_S (\varphi Q)$   
**assumes**  $\bigwedge i. \bigwedge P. i \in I \implies P \in carrier (Pring R I) \implies \varphi (P \otimes i) = (\varphi$   
 $P) \otimes_S (\varphi (mset-to-IP R \{i\}))$   
**assumes**  $\bigwedge k Q. k \in carrier R \implies Q \in carrier (Pring R I) \implies \varphi (poly-scalar-mult$   
 $R k Q) =$   
 $(\varphi (indexed-const k)) \otimes_S (\varphi Q)$   
**shows** *ring-hom-ring*  $(Pring R I) S \varphi$   
 $\langle proof \rangle$

**lemma**(*in cring*) *indexed-const-Pring-mult*:  
**assumes**  $k \in carrier R$   
**assumes**  $P \in carrier (Pring R I)$   
**shows**  $(indexed-const k \otimes_{Pring R I} P) m = k \otimes_R (P m)$   
 $(P \otimes_{Pring R I} indexed-const k) m = k \otimes_R (P m)$   
 $\langle proof \rangle$

**lemma**(*in cring*) *ring-hom-to-IP-ring-hom-is-hom*:  
**assumes** *cring*  $S$   
**assumes** *ring-hom-ring*  $R S \varphi$   
**shows** *ring-hom-ring*  $(Pring R I) (Pring S I) (ring-hom-to-IP-ring-hom \varphi)$   
 $\langle proof \rangle$

**lemma** *ring-hom-to-IP-ring-hom-smult*:  
**assumes** *cring*  $S$   
**assumes** *ring-hom-ring*  $R S \varphi$   
**assumes**  $P \in carrier (Pring R I)$

**assumes**  $a \in \text{carrier } R$   
**shows**  $\text{ring-hom-to-IP-ring-hom } \varphi (a \odot_{\text{Pring } R} I^P) =$   
 $\varphi a \odot_{\text{Pring } S} I (\text{ring-hom-to-IP-ring-hom } \varphi P)$   
 ⟨proof⟩

## 2.12.2 A Universal Property for Indexed Polynomial Rings

**lemma** *Pring-universal-prop-0*:  
**assumes**  $a\text{-cring}: \text{cring } S$   
**assumes**  $\text{index-map}: \text{closed-fun } S g$   
**assumes**  $\text{ring-hom}: \text{ring-hom-ring } R S \varphi$   
**assumes**  $\psi = (\text{total-eval } S g) \circ (\text{ring-hom-to-IP-ring-hom } \varphi)$   
**shows**  $(\text{ring-hom-ring } (\text{Pring } R I) S \psi)$   
 $(\forall i \in I. \psi (\text{mset-to-IP } R \{\#i\}) = g i)$   
 $(\forall a \in \text{carrier } R. \psi (\text{indexed-const } a) = \varphi a)$   
 $\forall \varrho. (\text{ring-hom-ring } (\text{Pring } R I) S \varrho) \wedge$   
 $(\forall i \in I. \varrho (\text{mset-to-IP } R \{\#i\}) = g i) \wedge$   
 $(\forall a \in \text{carrier } R. \varrho (\text{indexed-const } a) = \varphi a) \longrightarrow$   
 $(\forall x \in \text{carrier } (\text{Pring } R I). \varrho x = \psi x)$   
 ⟨proof⟩

**end**

**definition**  $\text{close-fun} :: 'c \text{ set} \Rightarrow ('e, 'f) \text{ ring-scheme} \Rightarrow ('c \Rightarrow 'e) \Rightarrow ('c \Rightarrow 'e)$   
**where**  
 $\text{close-fun } I S g = (\lambda i. (\text{if } i \in I \text{ then } g i \text{ else } \mathbf{0}_S))$

**context** *cring*  
**begin**

**lemma** *close-funE*:  
**assumes**  $\text{cring } S$   
**assumes**  $g \in I \rightarrow \text{carrier } S$   
**shows**  $\text{closed-fun } S (\text{close-fun } I S g)$   
 ⟨proof⟩

**end**

**definition** *indexed-poly-induced-morphism* ::  
 $'c \text{ set} \Rightarrow ('e, 'f) \text{ ring-scheme} \Rightarrow ('a, 'e) \text{ ring-hom} \Rightarrow ('c \Rightarrow 'e) \Rightarrow (( 'a, 'c) \text{ mvar-poly},$   
 $'e) \text{ ring-hom}$  **where**  
 $\text{indexed-poly-induced-morphism } I S \varphi g = (\text{total-eval } S (\text{close-fun } I S g)) \circ (\text{ring-hom-to-IP-ring-hom } \varphi)$

**context** *cring*  
**begin**

**lemma** *Pring-universal-prop*:  
**assumes**  $a\text{-cring}: \text{cring } S$

**assumes** *index-map*:  $g \in I \rightarrow \text{carrier } S$   
**assumes** *ring-hom*: *ring-hom-ring*  $R S \varphi$   
**assumes**  $\psi = \text{indexed-poly-induced-morphism } I S \varphi g$   
**shows** (*ring-hom-ring* (*Pring*  $R I$ )  $S \psi$ )  
 $(\forall i \in I. \psi (\text{mset-to-IP } R \{\#i\}) = g i)$   
 $(\forall a \in \text{carrier } R. \psi (\text{indexed-const } a) = \varphi a)$   
 $\forall \varrho. (\text{ring-hom-ring } (\text{Pring } R I) S \varrho) \wedge$   
 $(\forall i \in I. \varrho (\text{mset-to-IP } R \{\#i\}) = g i) \wedge$   
 $(\forall a \in \text{carrier } R. \varrho (\text{indexed-const } a) = \varphi a) \longrightarrow$   
 $(\forall x \in \text{carrier } (\text{Pring } R I). \varrho x = \psi x)$   
 <proof>

## 2.13 Mapping Multivariate Polynomials over a Single Variable to Univariate Polynomials

Constructor for multisets which have one distinct element

**definition** *nat-to-mset* ::  $'c \Rightarrow \text{nat} \Rightarrow 'c \text{ monomial}$  **where**  
 $\text{nat-to-mset } i n = \text{Abs-multiset } (\lambda j. \text{if } (j = i) \text{ then } n \text{ else } 0)$

**lemma** *nat-to-msetE*:  $\text{count } (\text{nat-to-mset } i n) i = n$   
 <proof>

**lemma** *nat-to-msetE'*:  
**assumes**  $j \neq i$   
**shows**  $\text{count } (\text{nat-to-mset } i n) j = 0$   
 <proof>

**lemma** *nat-to-mset-add*:  $\text{nat-to-mset } i (n + m) = (\text{nat-to-mset } i n) + (\text{nat-to-mset } i m)$   
 <proof>

**lemma** *nat-to-mset-inj*:  
**assumes**  $n \neq m$   
**shows**  $(\text{nat-to-mset } i n) \neq (\text{nat-to-mset } i m)$   
 <proof>

**lemma** *nat-to-mset-zero*:  $\text{nat-to-mset } i 0 = \{\#\}$   
 <proof>

**lemma** *nat-to-mset-Suc*:  $\text{nat-to-mset } i (\text{Suc } n) = \text{add-mset } i (\text{nat-to-mset } i n)$   
 <proof>

**lemma** *nat-to-mset-Pring-singleton*:  
**assumes** *cring*  $R$   
**assumes**  $P \in \text{carrier } (\text{Pring } R \{i\})$   
**assumes**  $m \in \text{monomials-of } R P$   
**shows**  $m = \text{nat-to-mset } i (\text{count } m i)$   
 <proof>

**definition** *IP-to-UP* :: 'd ⇒ ('e, 'd) mvar-poly ⇒ 'e u-poly **where**  
*IP-to-UP* i P = (λ (n::nat). P (nat-to-mset i n))

**lemma** *IP-to-UP-closed*:

**assumes** cring R

**assumes** P ∈ carrier (Pring R {i::'c})

**shows** *IP-to-UP* i P ∈ carrier (UP R)

⟨proof⟩

**lemma** *IP-to-UP-var*:

**shows** *IP-to-UP* i (mset-to-IP R {#i#}) = X-poly R

⟨proof⟩

**end**

**context** UP-cring

**begin**

**lemma** *IP-to-UP-monom*:

**shows** *IP-to-UP* i (mset-to-IP R (nat-to-mset i n)) = ((X-poly R)[ $\uparrow$ ]<sub>UP R</sub><sup>n</sup>)

⟨proof⟩

**lemma** *IP-to-UP-one*:

*IP-to-UP* i **1**<sub>Pring R {i}</sub> = **1**<sub>UP R</sub>

⟨proof⟩

**lemma** *IP-to-UP-zero*:

*IP-to-UP* i **0**<sub>Pring R {i}</sub> = **0**<sub>UP R</sub>

⟨proof⟩

**lemma** *IP-to-UP-add*:

**assumes** x ∈ carrier (Pring R {i})

**assumes** y ∈ carrier (Pring R {i})

**shows** *IP-to-UP* i (x ⊕<sub>Pring R {i}</sub> y) =

*IP-to-UP* i x ⊕<sub>UP R</sub> *IP-to-UP* i y

⟨proof⟩

**lemma** *IP-to-UP-indexed-const*:

**assumes** k ∈ carrier R

**shows** *IP-to-UP* i (ring.indexed-const R k) = to-polynomial R k

⟨proof⟩

**lemma** *IP-to-UP-indexed-pmult*:

**assumes** p ∈ carrier (Pring R {i})

**shows** *IP-to-UP* i (ring.indexed-pmult R p i) = (*IP-to-UP* i p) ⊗<sub>UP R</sub> (X-poly R)

⟨proof⟩

**lemma** *IP-to-UP-ring-hom*:

**shows** *ring-hom-ring* (*Pring*  $R \{i\}$ ) (*UP*  $R$ ) (*IP-to-UP*  $i$ )  
 ⟨*proof*⟩

**lemma** *IP-to-UP-ring-hom-inj*:

**shows** *inj-on* (*IP-to-UP*  $i$ ) (*carrier* (*Pring*  $R \{i\}$ ))  
 ⟨*proof*⟩

**lemma** *IP-to-UP-scalar-mult*:

**assumes**  $a \in \text{carrier } R$

**assumes**  $p \in \text{carrier } (\text{Pring } R \{i\})$

**shows** (*IP-to-UP*  $i$  ( $a \odot_{\text{Pring } R \{i\}} p$ )) =  $a \odot_{\text{UP } R} (\text{IP-to-UP } i p)$

⟨*proof*⟩

**end**

Evaluation of indexed polynomials commutes with evaluation of univariate polynomials:

**lemma** *pvar-closed*:

**assumes** *cring*  $R$

**assumes**  $i \in I$

**shows** (*pvar*  $R i$ )  $\in \text{carrier } (\text{Pring } R I)$

⟨*proof*⟩

**context** *UP-cring*

**begin**

**lemma** *pvar-mult*:

**assumes**  $i \in I$

**assumes**  $j \in I$

**shows** (*pvar*  $R i$ )  $\otimes_{\text{Pring } R I}$  (*pvar*  $R j$ ) = *mset-to-IP*  $R \{\#i, j\#}$

⟨*proof*⟩

**lemma** *pvar-pow*:

**assumes**  $i \in I$

**shows** (*pvar*  $R i$ )  $[\bigwedge]_{\text{Pring } R I} (n::\text{nat})$  = *mset-to-IP*  $R (\text{nat-to-mset } i n)$

⟨*proof*⟩

**lemma** *IP-to-UP-poly-eval*:

**assumes**  $p \in \text{Pring-set } R \{i\}$

**assumes** *closed-fun*  $R g$

**shows** *total-eval*  $R g p$  = *to-function*  $R (\text{IP-to-UP } i p)$  ( $g i$ )

⟨*proof*⟩

**end**

## 2.14 Mapping Univariate Polynomials to Multivariate Polynomials over a Singleton Variable Set

**definition**  $UP\text{-to-IP} :: ('a, 'b) \text{ ring-scheme} \Rightarrow 'c \Rightarrow 'a \text{ u-poly} \Rightarrow ('a, 'c) \text{ mvar-poly}$   
**where**  
 $UP\text{-to-IP } R \ i \ P = (\lambda m. \text{if } (\text{set-mset } m) \subseteq \{i\} \text{ then } P \ (\text{count } m \ i) \text{ else } \mathbf{0}_R)$

**context**  $UP\text{-cring}$   
**begin**

**lemma**  $UP\text{-to-IP-inv}$ :  
**assumes**  $p \in \text{Pring-set } R \ \{i\}$   
**shows**  $UP\text{-to-IP } R \ i \ (\text{IP-to-UP } i \ p) = p$   
 $\langle \text{proof} \rangle$

**lemma**  $UP\text{-to-IP-const}$ :  
**assumes**  $a \in \text{carrier } R$   
**shows**  $UP\text{-to-IP } R \ i \ (\text{to-polynomial } R \ a) = \text{ring.indexed-const } R \ a$   
 $\langle \text{proof} \rangle$

**lemma**  $UP\text{-to-IP-add}$ :  
**assumes**  $p \in \text{carrier } (UP \ R)$   
**assumes**  $Q \in \text{carrier } (UP \ R)$   
**shows**  $UP\text{-to-IP } R \ i \ (p \oplus_{UP \ R} Q) =$   
 $UP\text{-to-IP } R \ i \ p \oplus_{\text{Pring } R \ \{i\}} UP\text{-to-IP } R \ i \ Q$   
 $\langle \text{proof} \rangle$

**lemma**  $UP\text{-to-IP-var}$ :  
**shows**  $UP\text{-to-IP } R \ i \ (X\text{-poly } R) = \text{pvar } R \ i$   
 $\langle \text{proof} \rangle$

**lemma**  $UP\text{-to-IP-var-pow}$ :  
**shows**  $UP\text{-to-IP } R \ i \ ((X\text{-poly } R) [\overset{\wedge}{\wedge}]_{UP \ R} (n::\text{nat})) = (\text{pvar } R \ i) [\overset{\wedge}{\wedge}]_{\text{Pring } R \ \{i\}} n$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{one-var-indexed-poly-monom-simp}$ :  
**assumes**  $a \in \text{carrier } R$   
**shows**  $(a \odot_{\text{Pring } R \ \{i\}} ((\text{pvar } R \ i) [\overset{\wedge}{\wedge}]_{\text{Pring } R \ \{i\}} n)) \ x = (\text{if } x = (\text{nat-to-mset } i \ n) \text{ then } a \text{ else } \mathbf{0})$   
 $\langle \text{proof} \rangle$

**lemma**  $UP\text{-to-IP-monom}$ :  
**assumes**  $a \in \text{carrier } R$   
**shows**  $UP\text{-to-IP } R \ i \ (\text{up-ring.monom } (UP \ R) \ a \ n) = a \odot_{\text{Pring } R \ \{i\}} ((\text{pvar } R \ i) [\overset{\wedge}{\wedge}]_{\text{Pring } R \ \{i\}} n)$   
 $\langle \text{proof} \rangle$

**lemma**  $UP\text{-to-IP-monom}'$ :

**assumes**  $a \in \text{carrier } R$   
**shows**  $UP\text{-to-IP } R \ i \ (\text{up-ring.monom } (UP \ R) \ a \ n) = a \odot_{Pring \ R \ \{i\}} ((\text{pvar } R \ i)[\bigwedge_{Pring \ R \ \{i\}} n])$   
 $\langle \text{proof} \rangle$

**lemma** *UP-to-IP-closed:*

**assumes**  $p \in \text{carrier } P$   
**shows**  $(UP\text{-to-IP } R \ i \ p) \in \text{carrier } (Pring \ R \ \{i\})$   
 $\langle \text{proof} \rangle$

**lemma** *IP-to-UP-inv:*

**assumes**  $p \in \text{carrier } P$   
**shows**  $IP\text{-to-UP } i \ (UP\text{-to-IP } R \ i \ p) = p$   
 $\langle \text{proof} \rangle$

**lemma** *UP-to-IP-mult:*

**assumes**  $p \in \text{carrier } (UP \ R)$   
**assumes**  $Q \in \text{carrier } (UP \ R)$   
**shows**  $UP\text{-to-IP } R \ i \ (p \otimes_{UP \ R} Q) =$   
 $UP\text{-to-IP } R \ i \ p \otimes_{Pring \ R \ \{i\}} UP\text{-to-IP } R \ i \ Q$   
 $\langle \text{proof} \rangle$

**lemma** *UP-to-IP-ring-hom:*

**shows**  $\text{ring-hom-ring } (UP \ R) \ (Pring \ R \ \{i\}) \ (UP\text{-to-IP } R \ i)$   
 $\langle \text{proof} \rangle$

**end**

### 2.14.1 The isomorphism $R[I \cup J] \sim R[I][J]$ , where $I$ and $J$ are disjoint variable sets

Given a ring  $R$  and variable sets  $I$  and  $J$ , we'd like to construct the canonical (iso)morphism  $R[I \cup J] \rightarrow R[I][J]$ . This can be done with the universal property of the previous section. Let  $\phi : R \rightarrow R[J]$  be the inclusion of constants, and  $f : J \rightarrow R[I]$  be the map which sends the variable  $i$  to the polynomial variable  $i$  over the ring  $R[I][J]$ . Then these are the two basic pieces of input required to give us a canonical homomorphism  $R[I \cup J] \rightarrow R[I][J]$  with the universal property. The first map  $\phi$  will be "dist\_varset\_morphsim" below, and the second map will be "dist\_varset\_var\_ass". The desired induced isomorphism will be called "var\_factor".

**definition**(in *ring*) *dist-varset-morphism*

$:: 'd \ \text{set} \Rightarrow 'd \ \text{set} \Rightarrow$   
 $( 'a, (('a, 'd) \ \text{mvar-poly}, 'd) \ \text{mvar-poly}) \ \text{ring-hom} \ \mathbf{where}$   
 $\text{dist-varset-morphism } (I :: 'd \ \text{set}) \ (J :: 'd \ \text{set}) =$   
 $(\text{ring.indexed-const } (Pring \ R \ J) :: ('d \ \text{multiset} \Rightarrow 'a) \Rightarrow 'd \ \text{multiset} \Rightarrow ('d$   
 $\text{multiset} \Rightarrow 'a)) \circ (\text{ring.indexed-const } R :: 'a \Rightarrow 'd \ \text{multiset} \Rightarrow 'a)$

**definition**(in ring) *dist-varset-var-ass*  
 :: 'd set  $\Rightarrow$  'd set  $\Rightarrow$  'd  $\Rightarrow$  (('a, 'd) mvar-poly, 'd) mvar-poly  
**where**  
*dist-varset-var-ass* (I:: 'd set) (J:: 'd set) = ( $\lambda i$ . if  $i \in J$  then ring.indexed-const  
 (Pring R J) (pvar R i) else  
 pvar (Pring R J) i )

**context** cring  
**begin**

**lemma** *dist-varset-morphism-is-morphism*:  
 assumes (I:: 'd set)  $\subseteq$  J0  $\cup$  J1  
 assumes J1  $\subseteq$  I  
 assumes  $\varphi = \text{dist-varset-morphism } I \ J0$   
 shows ring-hom-ring R (Pring (Pring R J0) J1)  $\varphi$   
 <proof>

**definition** *var-factor* ::  
 'd set  $\Rightarrow$  'd set  $\Rightarrow$  'd set  $\Rightarrow$   
 (('a, 'd) mvar-poly, (('a, 'd) mvar-poly, 'd) mvar-poly) ring-hom **where**  
*var-factor* (I:: 'd set) (J0:: 'd set) (J1:: 'd set) = indexed-poly-induced-morphism  
 I (Pring (Pring R J0) J1)  
 (dist-varset-morphism I J0)  
 (dist-varset-var-ass I J0)

**lemma** *indexed-const-closed*:  
 assumes  $x \in \text{carrier } R$   
 shows indexed-const  $x \in \text{carrier } (\text{Pring } R \ I)$   
 <proof>

**lemma** *var-factor-morphism*:  
 assumes (I:: 'd set)  $\subseteq$  J0  $\cup$  J1  
 assumes J1  $\subseteq$  I  
 assumes J1  $\cap$  J0 = {}  
 assumes  $g = \text{dist-varset-var-ass } I \ J0$   
 assumes  $\varphi = \text{dist-varset-morphism } I \ J0$   
 assumes  $\psi = (\text{var-factor } I \ J0 \ J1)$   
 shows ring-hom-ring (Pring R I) (Pring (Pring R J0) J1)  $\psi$   
 $\bigwedge i. i \in J0 \cap I \Longrightarrow \psi (\text{pvar } R \ i) = \text{ring.indexed-const } (\text{Pring } R \ J0) (\text{pvar}$   
 R i)  
 $\bigwedge i. i \in J1 \Longrightarrow \psi (\text{pvar } R \ i) = \text{pvar } (\text{Pring } R \ J0) \ i$   
 $\bigwedge a. a \in \text{carrier } (\text{Pring } R \ (J0 \cap I)) \Longrightarrow \psi \ a = \text{ring.indexed-const } (\text{Pring } R$   
 J0) a  
 <proof>

**lemma** *var-factor-morphism'*:  
 assumes  $I = J0 \cup J1$   
 assumes J1  $\subseteq$  I  
 assumes J1  $\cap$  J0 = {}

**assumes**  $\psi = (\text{var-factor } I \ J0 \ J1)$   
**shows**  $\text{ring-hom-ring } (\text{Pring } R \ I) \ (\text{Pring } (\text{Pring } R \ J0) \ J1) \ \psi$   
 $\bigwedge i. i \in J1 \implies \psi (\text{pvar } R \ i) = \text{pvar } (\text{Pring } R \ J0) \ i$   
 $\bigwedge a. a \in \text{carrier } (\text{Pring } R \ (J0 \cap I)) \implies \psi \ a = \text{ring.indexed-const } (\text{Pring } R \ J0) \ a$   
 $\langle \text{proof} \rangle$

Constructing the inverse morphism for `var_factor_morphism`

**lemma** *pvar-ass-closed*:

**assumes**  $J1 \subseteq I$   
**shows**  $\text{pvar } R \in J1 \rightarrow \text{carrier } (\text{Pring } R \ I)$   
 $\langle \text{proof} \rangle$

The following function gives us the inverse morphism  $R[I][J] \rightarrow R[I \cup J]$ :

**definition** *var-factor-inv* ::  $'d \ \text{set} \Rightarrow 'd \ \text{set} \Rightarrow 'd \ \text{set} \Rightarrow$   
 $((('a, 'd) \ \text{mvar-poly}, 'd) \ \text{mvar-poly}, ('a, 'd) \ \text{mvar-poly}) \ \text{ring-hom}$  **where**  
 $\text{var-factor-inv } (I:: 'd \ \text{set}) \ (J0:: 'd \ \text{set}) \ (J1:: 'd \ \text{set}) = \text{indexed-poly-induced-morphism}$   
 $J1 \ (\text{Pring } R \ I)$   
 $(\text{id}:: ('d \ \text{multiset} \Rightarrow 'a) \Rightarrow 'd \ \text{multiset})$   
 $\Rightarrow 'a)$   
 $(\text{pvar } R)$

**lemma** *var-factor-inv-morphism*:

**assumes**  $I = J0 \cup J1$   
**assumes**  $J1 \subseteq I$   
**assumes**  $J1 \cap J0 = \{\}$   
**assumes**  $\psi = (\text{var-factor-inv } I \ J0 \ J1)$   
**shows**  $\text{ring-hom-ring } (\text{Pring } (\text{Pring } R \ J0) \ J1) \ (\text{Pring } R \ I) \ \psi$   
 $\bigwedge i. i \in J1 \implies \psi (\text{pvar } (\text{Pring } R \ J0) \ i) = \text{pvar } R \ i$   
 $\bigwedge a. a \in \text{carrier } (\text{Pring } R \ J0) \implies \psi (\text{ring.indexed-const } (\text{Pring } R \ J0) \ a) =$   
 $a$   
 $\langle \text{proof} \rangle$

**lemma** *var-factor-inv-inverse*:

**assumes**  $I = J0 \cup J1$   
**assumes**  $J1 \subseteq I$   
**assumes**  $J1 \cap J0 = \{\}$   
**assumes**  $\psi1 = (\text{var-factor-inv } I \ J0 \ J1)$   
**assumes**  $\psi0 = (\text{var-factor } I \ J0 \ J1)$   
**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $\psi1 \ (\psi0 \ P) = P$   
 $\langle \text{proof} \rangle$

**lemma** *var-factor-total-eval*:

**assumes**  $I = J0 \cup J1$   
**assumes**  $J1 \subseteq I$   
**assumes**  $J1 \cap J0 = \{\}$   
**assumes**  $\psi = (\text{var-factor } I \ J0 \ J1)$   
**assumes**  $\text{closed-fun } R \ g$

**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $\text{total-eval } R \ g \ P = \text{total-eval } R \ g \ (\text{total-eval } (\text{Pring } R \ J0) \ (\text{indexed-const } \circ \ g) \ (\psi \ P))$   
 $\langle \text{proof} \rangle$

**lemma** *var-factor-closed*:

**assumes**  $I = J0 \cup J1$   
**assumes**  $J1 \subseteq I$   
**assumes**  $J1 \cap J0 = \{\}$   
**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $\text{var-factor } I \ J0 \ J1 \ P \in \text{carrier } (\text{Pring } (\text{Pring } R \ J0) \ J1)$   
 $\langle \text{proof} \rangle$

**lemma** *var-factor-add*:

**assumes**  $I = J0 \cup J1$   
**assumes**  $J1 \subseteq I$   
**assumes**  $J1 \cap J0 = \{\}$   
**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $Q \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $\text{var-factor } I \ J0 \ J1 \ (P \oplus_{\text{Pring } R \ I} Q) = \text{var-factor } I \ J0 \ J1 \ P \oplus_{\text{Pring } (\text{Pring } R \ J0) \ J1} \text{var-factor } I \ J0 \ J1 \ Q$   
 $\langle \text{proof} \rangle$

**lemma** *var-factor-mult*:

**assumes**  $I = J0 \cup J1$   
**assumes**  $J1 \subseteq I$   
**assumes**  $J1 \cap J0 = \{\}$   
**assumes**  $P \in \text{carrier } (\text{Pring } R \ I)$   
**assumes**  $Q \in \text{carrier } (\text{Pring } R \ I)$   
**shows**  $\text{var-factor } I \ J0 \ J1 \ (P \otimes_{\text{Pring } R \ I} Q) = \text{var-factor } I \ J0 \ J1 \ P \otimes_{\text{Pring } (\text{Pring } R \ J0) \ J1} \text{var-factor } I \ J0 \ J1 \ Q$   
 $\langle \text{proof} \rangle$

## 2.14.2 Viewing a Multivariable Polynomial as a Univariate Polynomial over a Multivariate Polynomial Base Ring

**definition** *multivar-poly-to-univ-poly* ::

$'c \ \text{set} \Rightarrow 'c \Rightarrow ('a, 'c) \ \text{mvar-poly} \Rightarrow$   
 $((('a, 'c) \ \text{mvar-poly}) \ \text{u-poly} \ \mathbf{where}$   
 $\text{multivar-poly-to-univ-poly } I \ i \ P = ((\text{IP-to-UP } i) \circ (\text{var-factor } I \ (I - \{i\}) \ \{i\})) \ P$

**definition** *univ-poly-to-multivar-poly* ::

$'c \ \text{set} \Rightarrow 'c \Rightarrow (('a, 'c) \ \text{mvar-poly}) \ \text{u-poly} \Rightarrow$   
 $('a, 'c) \ \text{mvar-poly} \ \mathbf{where}$   
 $\text{univ-poly-to-multivar-poly } I \ i \ P = ((\text{var-factor-inv } I \ (I - \{i\}) \ \{i\}) \circ (\text{UP-to-IP } (\text{Pring } R \ (I - \{i\})) \ i)) \ P$

**lemma** *multivar-poly-to-univ-poly-is-hom*:

**assumes**  $i \in I$

**shows** *multivar-poly-to-univ-poly*  $I i \in \text{ring-hom} (\text{Pring } R I) (UP (\text{Pring } R (I - \{i\})))$   
 ⟨*proof*⟩

**lemma** *multivar-poly-to-univ-poly-inverse*:  
**assumes**  $i \in I$   
**assumes**  $\psi 0 = \text{multivar-poly-to-univ-poly } I i$   
**assumes**  $\psi 1 = \text{univ-poly-to-multivar-poly } I i$   
**assumes**  $P \in \text{carrier} (\text{Pring } R I)$   
**shows**  $\psi 1 (\psi 0 P) = P$   
 ⟨*proof*⟩

**lemma** *multivar-poly-to-univ-poly-total-eval*:  
**assumes**  $i \in I$   
**assumes**  $\psi = \text{multivar-poly-to-univ-poly } I i$   
**assumes**  $P \in \text{carrier} (\text{Pring } R I)$   
**assumes** *closed-fun*  $R g$   
**shows**  $\text{total-eval } R g P = \text{total-eval } R g (\text{to-function} (\text{Pring } R (I - \{i\})) (\psi P))$   
 (*indexed-const*  $(g i)$ )  
 ⟨*proof*⟩

Induction for polynomial rings. Basically just `indexed_pset.induct` with some boilerplate translations removed

**lemma**(*in ring*) *Pring-car-induct''*:  
**assumes**  $Q \in \text{carrier} (\text{Pring } R I)$   
**assumes**  $\bigwedge c. c \in \text{carrier } R \implies P (\text{indexed-const } c)$   
**assumes**  $\bigwedge p q. p \in \text{carrier} (\text{Pring } R I) \implies q \in \text{carrier} (\text{Pring } R I) \implies P p \implies P q \implies P (p \oplus_{\text{Pring } R I} q)$   
**assumes**  $\bigwedge p i. p \in \text{carrier} (\text{Pring } R I) \implies i \in I \implies P p \implies P (p \otimes_{\text{Pring } R I} \text{pvar } R i)$   
**shows**  $P Q$   
 ⟨*proof*⟩

### 2.14.3 Application: A Polynomial Ring over a Domain is a Domain

In this section, we use the fact the  $UP R$  is a domain when  $R$  is a domain to show the analogous result for indexed polynomial rings. We first prove it inductively for rings with a finite variable set, and then generalize to infinite variable sets using the fact that any two multivariable polynomials over an indexed polynomial ring must also lie in a finitely indexed polynomial subring.

**lemma** *indexed-const-mult*:  
**assumes**  $a \in \text{carrier } R$   
**assumes**  $b \in \text{carrier } R$   
**shows**  $\text{indexed-const } a \otimes_{\text{Pring } R I} \text{indexed-const } b = \text{indexed-const } (a \otimes b)$   
 ⟨*proof*⟩

**lemma**(in *domain*) *Pring-fin-vars-is-domain*:

**assumes** *finite* ( $I :: 'c$  set)  
**shows** *domain* (*Pring*  $R$   $I$ )

$\langle$ *proof* $\rangle$

**lemma** *locally-finite*:

**assumes**  $a \in \text{carrier } (\text{Pring } R I)$   
**shows**  $\exists J. J \subseteq I \wedge \text{finite } J \wedge a \in \text{carrier } (\text{Pring } R J)$

$\langle$ *proof* $\rangle$

**lemma**(in *domain*) *Pring-is-domain*:

*domain* (*Pring*  $R$   $I$ )

$\langle$ *proof* $\rangle$

#### 2.14.4 Relabelling of Variables for Indexed Polynomial Rings

**definition** *relabel-vars* ::  $'d$  set  $\Rightarrow$   $'e$  set  $\Rightarrow$  ( $'d \Rightarrow 'e$ )  $\Rightarrow$

( $'a, 'd$ ) *mvar-poly*  $\Rightarrow$  ( $'a, 'e$ ) *mvar-poly* **where**

*relabel-vars*  $I$   $J$   $g = \text{indexed-poly-induced-morphism } I$  (*Pring*  $R$   $J$ ) *indexed-const*  
( $\lambda i. \text{pvar } R (g i)$ )

**lemma** *relabel-vars-is-morphism*:

**assumes**  $g \in I \rightarrow J$

**shows** *ring-hom-ring* (*Pring*  $R$   $I$ ) (*Pring*  $R$   $J$ ) (*relabel-vars*  $I$   $J$   $g$ )

$\bigwedge i. i \in I \implies \text{relabel-vars } I J g (\text{pvar } R i) = \text{pvar } R (g i)$

$\bigwedge c. c \in \text{carrier } R \implies \text{relabel-vars } I J g (\text{indexed-const } c) = \text{indexed-const } c$

$\langle$ *proof* $\rangle$

**lemma** *relabel-vars-add*:

**assumes**  $g \in I \rightarrow J$

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

**assumes**  $Q \in \text{carrier } (\text{Pring } R I)$

**shows** *relabel-vars*  $I$   $J$   $g$  ( $P \oplus_{\text{Pring } R I} Q$ ) = *relabel-vars*  $I$   $J$   $g$   $P \oplus_{\text{Pring } R J}$

*relabel-vars*  $I$   $J$   $g$   $Q$

$\langle$ *proof* $\rangle$

**lemma** *relabel-vars-mult*:

**assumes**  $g \in I \rightarrow J$

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

**assumes**  $Q \in \text{carrier } (\text{Pring } R I)$

**shows** *relabel-vars*  $I$   $J$   $g$  ( $P \otimes_{\text{Pring } R I} Q$ ) = *relabel-vars*  $I$   $J$   $g$   $P \otimes_{\text{Pring } R J}$

*relabel-vars*  $I$   $J$   $g$   $Q$

$\langle$ *proof* $\rangle$

**lemma** *relabel-vars-closed*:

**assumes**  $g \in I \rightarrow J$

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

**shows** *relabel-vars*  $I$   $J$   $g$   $P \in \text{carrier } (\text{Pring } R J)$

*<proof>*

**lemma** *relabel-vars-smult*:

**assumes**  $g \in I \rightarrow J$

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

**assumes**  $a \in \text{carrier } R$

**shows**  $\text{relabel-vars } I \ J \ g \ (a \odot_{\text{Pring } R} I^P) = a \odot_{\text{Pring } R} J \ \text{relabel-vars } I \ J \ g \ P$

*<proof>*

**lemma** *relabel-vars-inverse*:

**assumes**  $g \in I \rightarrow J$

**assumes**  $h \in J \rightarrow I$

**assumes**  $\bigwedge i. i \in I \implies h (g \ i) = i$

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

**shows**  $\text{relabel-vars } J \ I \ h \ (\text{relabel-vars } I \ J \ g \ P) = P$

*<proof>*

**lemma** *relabel-vars-total-eval*:

**assumes**  $g \in I \rightarrow J$

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

**assumes** *closed-fun*  $R \ f$

**shows**  $\text{total-eval } R \ (f \circ g) \ P = \text{total-eval } R \ f \ (\text{relabel-vars } I \ J \ g \ P)$

*<proof>*

**end**

**end**

**theory** *Indices*

**imports** *Main*

**begin**

### 3 Basic Lemmas for Manipulating Indices and Lists

**fun** *index-list* **where**

*index-list*  $0 = []$

*index-list*  $(\text{Suc } n) = \text{index-list } n \ @ \ [n]$

**lemma** *index-list-length*:

*length*  $(\text{index-list } n) = n$

*<proof>*

**lemma** *index-list-indices*:

$k < n \implies (\text{index-list } n)!k = k$

*<proof>*

**lemma** *index-list-set*:

*set*  $(\text{index-list } n) = \{..<n\}$

*<proof>*

**fun** *flat-map* :: ('a => 'b list) => 'a list => 'b list **where**  
*flat-map* f [] = []  
|*flat-map* f (h#t) = (f h)@(flat-map f t)

**abbreviation**(*input*) *project-at-indices* ( $\pi$ .) **where**  
*project-at-indices* S as  $\equiv$  *nths* as S

**fun** *insert-at-index* :: 'a list => 'a => nat => 'a list **where**  
*insert-at-index* as a n = (take n as) @ (a#(drop n as))

**lemma** *insert-at-index-length*:  
**shows** *length* (insert-at-index as a n) = *length* as + 1  
<proof>

**lemma** *insert-at-index-eq[simp]*:  
**assumes**  $n \leq \text{length } as$   
**shows** (insert-at-index as a n)!n = a  
<proof>

**lemma** *insert-at-index-eq'[simp]*:  
**assumes**  $n \leq \text{length } as$   
**assumes**  $k < n$   
**shows** (insert-at-index as a n)!k = as ! k  
<proof>

**lemma** *insert-at-index-eq''[simp]*:  
**assumes**  $n < \text{length } as$   
**assumes**  $k \leq n$   
**shows** (insert-at-index as a k)!(Suc n) = as ! n  
<proof>

Correctness of *project\_at\_indices*

**definition** *indices-of* :: 'a list => nat set **where**  
*indices-of* as = {..*length* as}

**lemma** *proj-at-index-list-length[simp]*:  
**assumes**  $S \subseteq \text{indices-of } as$   
**shows** *length* (project-at-indices S as) = *card* S  
<proof>

A function which enumerates finite sets

**abbreviation**(*input*) *set-to-list* :: nat set => nat list **where**  
*set-to-list* S  $\equiv$  *sorted-list-of-set* S

**lemma** *set-to-list-set*:  
**assumes** *finite* S  
**shows** *set* (set-to-list S) = S  
<proof>

**lemma** *set-to-list-length*:  
**assumes** *finite S*  
**shows**  $\text{length } (\text{set-to-list } S) = \text{card } S$   
 $\langle \text{proof} \rangle$

**lemma** *set-to-list-empty*:  
**assumes**  $\text{card } S = 0$   
**shows**  $\text{set-to-list } S = []$   
 $\langle \text{proof} \rangle$

**lemma** *set-to-list-first*:  
**assumes**  $\text{card } S > 0$   
**shows**  $\text{Min } S = \text{set-to-list } S ! 0$   
 $\langle \text{proof} \rangle$

**lemma** *set-to-list-last*:  
**assumes**  $\text{card } S > 0$   
**shows**  $\text{Max } S = \text{last } (\text{set-to-list } S)$   
 $\langle \text{proof} \rangle$

**lemma** *set-to-list-insert-Max*:  
**assumes** *finite S*  
**assumes**  $\bigwedge s. s \in S \implies a > s$   
**shows**  $\text{set-to-list } (\text{insert } a \ S) = \text{set-to-list } S @ [a]$   
 $\langle \text{proof} \rangle$

**lemma** *set-to-list-insert-Min*:  
**assumes** *finite S*  
**assumes**  $\bigwedge s. s \in S \implies a < s$   
**shows**  $\text{set-to-list } (\text{insert } a \ S) = a \# \text{set-to-list } S$   
 $\langle \text{proof} \rangle$

**fun** *nth-elem* **where**  
 $\text{nth-elem } S \ n = \text{set-to-list } S ! n$

**lemma** *nth-elem-closed*:  
**assumes**  $i < \text{card } S$   
**shows**  $\text{nth-elem } S \ i \in S$   
 $\langle \text{proof} \rangle$

**lemma** *nth-elem-Min*:  
**assumes**  $\text{card } S > 0$   
**shows**  $\text{nth-elem } S \ 0 = \text{Min } S$   
 $\langle \text{proof} \rangle$

**lemma** *nth-elem-Max*:  
**assumes**  $\text{card } S > 0$   
**shows**  $\text{nth-elem } S \ (\text{card } S - 1) = \text{Max } S$

*<proof>*

**lemma** *nth-elem-Suc*:

**assumes**  $\text{card } S > \text{Suc } n$

**shows**  $\text{nth-elem } S (\text{Suc } n) > \text{nth-elem } S n$

*<proof>*

**lemma** *nth-elem-insert-Min*:

**assumes**  $\text{card } S > 0$

**assumes**  $a < \text{Min } S$

**shows**  $\text{nth-elem } (\text{insert } a \ S) (\text{Suc } i) = \text{nth-elem } S \ i$

*<proof>*

**lemma** *set-to-list-Suc-map*:

**assumes** *finite*  $S$

**shows**  $\text{set-to-list } (\text{Suc } ` \ S) = \text{map } \text{Suc } (\text{set-to-list } S)$

*<proof>*

**lemma** *nth-elem-Suc-im*:

**assumes**  $i < \text{card } S$

**shows**  $\text{nth-elem } (\text{Suc } ` \ S) \ i = \text{Suc } (\text{nth-elem } S \ i)$

*<proof>*

**lemma** *set-to-list-upto*:

$\text{set-to-list } \{.. $n\} = [0.. $n]$$$

*<proof>*

**lemma** *nth-elem-upto*:

**assumes**  $i < n$

**shows**  $\text{nth-elem } \{.. $n\} \ i = i$$

*<proof>*

Characterizing the entries of `project_at_indices`

**lemma** *project-at-indices-append*:

$\text{project-at-indices } S \ (as@bs) = \text{project-at-indices } S \ as \ @ \ \text{project-at-indices } \{j. j + \text{length } as \in S\} \ bs$

*<proof>*

**lemma** *project-at-indices-nth*:

**assumes**  $S \subseteq \text{indices-of } as$

**assumes**  $\text{card } S > i$

**shows**  $\text{project-at-indices } S \ as \ ! \ i = as \ ! \ (\text{nth-elem } S \ i)$

*<proof>*

An inverse for `nth_elem`

**definition** *set-rank* **where**

$\text{set-rank } S \ x = (\text{THE } i. i < \text{card } S \wedge x = \text{nth-elem } S \ i)$

**lemma** *set-rank-exist*:

**assumes** *finite S*  
**assumes**  $x \in S$   
**shows**  $\exists i. i < \text{card } S \wedge x = \text{nth-elem } S \ i$   
 ⟨*proof*⟩

**lemma** *set-rank-unique*:

**assumes** *finite S*  
**assumes**  $x \in S$   
**assumes**  $i < \text{card } S \wedge x = \text{nth-elem } S \ i$   
**assumes**  $j < \text{card } S \wedge x = \text{nth-elem } S \ j$   
**shows**  $i = j$   
 ⟨*proof*⟩

**lemma** *nth-elem-set-rank-inv*:

**assumes** *finite S*  
**assumes**  $x \in S$   
**shows**  $\text{nth-elem } S \ (\text{set-rank } S \ x) = x$   
 ⟨*proof*⟩

**lemma** *set-rank-nth-elem-inv*:

**assumes** *finite S*  
**assumes**  $i < \text{card } S$   
**shows**  $\text{set-rank } S \ (\text{nth-elem } S \ i) = i$   
 ⟨*proof*⟩

**lemma** *set-rank-range*:

**assumes** *finite S*  
**assumes**  $x \in S$   
**shows**  $\text{set-rank } S \ x < \text{card } S$   
 ⟨*proof*⟩

**lemma** *project-at-indices-nth'*:

**assumes**  $S \subseteq \text{indices-of } as$   
**assumes**  $i \in S$   
**shows**  $as \ ! \ i = \text{project-at-indices } S \ as \ ! \ (\text{set-rank } S \ i)$   
 ⟨*proof*⟩

**fun** *proj-away-from-index* ::  $\text{nat} \Rightarrow 'a \ \text{list} \Rightarrow 'a \ \text{list} \ (\pi_{\neq})$  **where**  
*proj-away-from-index*  $n \ as = (\text{take } n \ as) @ (\text{drop } (\text{Suc } n) \ as)$

*proj\_away\_from\_index* is an inverse to *insert\_at\_index*

**lemma** *insert-at-index-project-away[simp]*:

**assumes**  $k < \text{length } as$   
**assumes**  $bs = (\text{insert-at-index } as \ a \ k)$   
**shows**  $\pi_{\neq} \ k \ bs = as$   
 ⟨*proof*⟩

**definition** *fibred-cell* ::  $'a \ \text{list} \ \text{set} \Rightarrow ('a \ \text{list} \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \ \text{list} \ \text{set}$  **where**  
*fibred-cell*  $C \ P = \{as \ . \ \exists x \ t. as = (t \# x) \wedge x \in C \wedge (P \ x \ t)\}$

**definition** *fibred-cell-at-ind* ::  $\text{nat} \Rightarrow 'a \text{ list set} \Rightarrow ('a \text{ list} \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \text{ list set}$  **where**  
*fibred-cell-at-ind*  $n \ C \ P = \{as . \exists x \ t. as = (\text{insert-at-index } x \ t \ n) \wedge x \in C \wedge (P \ x \ t)\}$

**lemma** *fibred-cell-lengths*:  
**assumes**  $\bigwedge k. k \in C \implies \text{length } k = n$   
**shows**  $k \in (\text{fibred-cell } C \ P) \implies \text{length } k = \text{Suc } n$   
 $\langle \text{proof} \rangle$

**lemma** *fibred-cell-at-ind-lengths*:  
**assumes**  $\bigwedge k. k \in C \implies \text{length } k = n$   
**assumes**  $k \leq n$   
**shows**  $c \in (\text{fibred-cell-at-ind } k \ C \ P) \implies \text{length } c = \text{Suc } n$   
 $\langle \text{proof} \rangle$

**lemma** *project-fibred-cell*:  
**assumes**  $\bigwedge k. k \in C \implies \text{length } k = n$   
**assumes**  $k < n$   
**assumes**  $\forall x \in C. \exists t. P \ x \ t$   
**shows**  $\pi_{\neq k} ' (\text{fibred-cell-at-ind } k \ C \ P) = C$   
 $\langle \text{proof} \rangle$

**definition** *list-segment* **where**  
*list-segment*  $i \ j \ as = \text{map } (\text{nth } as) \ [i..<j]$

**lemma** *list-segment-length*:  
**assumes**  $i \leq j$   
**assumes**  $j \leq \text{length } as$   
**shows**  $\text{length } (\text{list-segment } i \ j \ as) = j - i$   
 $\langle \text{proof} \rangle$

**lemma** *list-segment-drop*:  
**assumes**  $i < \text{length } as$   
**shows**  $(\text{list-segment } i \ (\text{length } as) \ as) = \text{drop } i \ as$   
 $\langle \text{proof} \rangle$

**lemma** *list-segment-concat*:  
**assumes**  $j \leq k$   
**assumes**  $i \leq j$   
**shows**  $(\text{list-segment } i \ j \ as) @ (\text{list-segment } j \ k \ as) = (\text{list-segment } i \ k \ as)$   
 $\langle \text{proof} \rangle$

**lemma** *list-segment-subset*:  
**assumes**  $j \leq k$   
**shows**  $\text{set } (\text{list-segment } i \ j \ as) \subseteq \text{set } (\text{list-segment } i \ k \ as)$   
 $\langle \text{proof} \rangle$

**lemma** *list-segment-subset-list-set*:  
**assumes**  $j \leq \text{length } as$   
**shows**  $\text{set } (\text{list-segment } i \ j \ as) \subseteq \text{set } as$   
 $\langle \text{proof} \rangle$

**definition** *fun-inv* **where**  
 $\text{fun-inv} = \text{inv}$

**end**  
**theory** *Ring-Powers*  
**imports** *HOL-Algebra.Chinese-Remainder HOL-Combinatorics.List-Permutation*  
*Padic-Ints.Function-Ring HOL-Algebra.Generated-Rings Cring-Multivariable-Poly*  
*Indices*  
**begin**

**type-synonym** *arity* = *nat*  
**type-synonym** *'a tuple* = *'a list*

## 4 Cartesian Powers of a Ring

### 4.1 Constructing the Cartesian Power of a Ring

Powers of a ring

**R\_list**  $n \ R$  produces the list  $[R, \dots, R]$  of length  $n$

**fun** *R-list* ::  $\text{nat} \Rightarrow ('a, 'b) \text{ ring-scheme} \Rightarrow (('a, 'b) \text{ ring-scheme}) \text{ list}$  **where**  
 $\text{R-list } n \ R = \text{map } (\lambda-. R) (\text{index-list } n)$

Cartesian powers of a ring

**definition** *cartesian-power* ::  $('a, 'b) \text{ ring-scheme} \Rightarrow \text{nat} \Rightarrow ('a \text{ list}) \text{ ring}$  (*- 80*)  
**where**  
 $R^n \equiv \text{RDirProd-list } (\text{R-list } n \ R)$

**lemma** *R-list-length*:  
 $\text{length } (\text{R-list } n \ R) = n$   
 $\langle \text{proof} \rangle$

**lemma** *R-list-nth*:  
 $i < n \implies \text{R-list } n \ R ! i = R$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-car-memI*:  
**assumes**  $\text{length } as = n$   
**assumes**  $\text{set } as \subseteq \text{carrier } R$   
**shows**  $as \in \text{carrier } (R^n)$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-car-memI'*:  
**assumes**  $\text{length } as = n$   
**assumes**  $\bigwedge i. i < n \implies as ! i \in \text{carrier } R$   
**shows**  $as \in \text{carrier } (R^n)$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-car-memE*:  
**assumes**  $as \in \text{carrier } (R^n)$   
**shows**  $\text{length } as = n$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-car-memE'*:  
**assumes**  $as \in \text{carrier } (R^n)$   
**assumes**  $i < n$   
**shows**  $as ! i \in \text{carrier } R$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-car-memE''*:  
**assumes**  $as \in \text{carrier } (R^n)$   
**shows**  $\text{set } as \subseteq \text{carrier } R$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-car-memI''*:  
**assumes**  $\text{length } as = n + k$   
**assumes**  $\text{take } n \text{ } as \in \text{carrier } (R^n)$   
**assumes**  $\text{drop } n \text{ } as \in \text{carrier } (R^k)$   
**shows**  $as \in \text{carrier } (R^{n+k})$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-cons*:  
**assumes**  $as \in \text{carrier } (R^n)$   
**assumes**  $a \in \text{carrier } R$   
**shows**  $a \# as \in \text{carrier } (R^{n+1})$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-append*:  
**assumes**  $as \in \text{carrier } (R^n)$   
**assumes**  $a \in \text{carrier } R$   
**shows**  $as @ [a] \in \text{carrier } (R^{n+1})$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-head*:  
**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$   
**shows**  $\text{hd } as \in \text{carrier } R$   
 $\langle \text{proof} \rangle$

**lemma** *cartesian-power-tail*:  
**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**shows**  $tl\ as \in carrier\ (R^n)$   
 $\langle proof \rangle$

**lemma** *insert-at-index-closed*:

**assumes**  $length\ as = n$   
**assumes**  $as \in carrier\ (R^n)$   
**assumes**  $a \in carrier\ R$   
**assumes**  $k \leq n$   
**shows**  $(insert-at-index\ as\ a\ k) \in carrier\ (R^{Suc\ n})$   
 $\langle proof \rangle$

**lemma** *insert-at-index-pow-not-car*:

**assumes**  $k \leq n$   
**assumes**  $length\ x = n$   
**assumes**  $(insert-at-index\ x\ a\ k) \in carrier\ (R^{Suc\ n})$   
**shows**  $x \in carrier\ (R^n)$   
 $\langle proof \rangle$

**lemma** *insert-at-index-pow-not-car'*:

**assumes**  $k \leq n$   
**assumes**  $length\ x = n$   
**assumes**  $x \notin carrier\ (R^n)$   
**shows**  $(insert-at-index\ x\ a\ n) \notin carrier\ (R^{Suc\ n})$   
 $\langle proof \rangle$

**lemma** *take-closed*:

**assumes**  $k \leq n$   
**assumes**  $x \in carrier\ (R^n)$   
**shows**  $take\ k\ x \in carrier\ (R^k)$   
 $\langle proof \rangle$

**lemma** *drop-closed*:

**assumes**  $k < n$   
**assumes**  $x \in carrier\ (R^n)$   
**shows**  $drop\ k\ x \in carrier\ (R^{n-k})$   
 $\langle proof \rangle$

**lemma** *last-closed*:

**assumes**  $n > 0$   
**assumes**  $x \in carrier\ (R^n)$   
**shows**  $last\ x \in carrier\ R$   
 $\langle proof \rangle$

**lemma** *cartesian-power-concat*:

**assumes**  $a \in carrier\ (R^n)$   
**assumes**  $b \in carrier\ (R^k)$   
**shows**  $a@b \in carrier\ (R^{n+k})$   
 $b@a \in carrier\ (R^{n+k})$

*<proof>*

**lemma** *cartesian-power-decomp*:

**assumes**  $a \in \text{carrier } (R^{n+k})$

**obtains**  $a0\ a1$  **where**  $a0 \in \text{carrier } (R^n) \wedge a1 \in \text{carrier } (R^k) \wedge a0@a1 = a$

*<proof>*

**lemma** *list-segment-pow*:

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $j \leq n$

**assumes**  $i \leq j$

**shows** *list-segment*  $i\ j\ as \in \text{carrier } (R^{j-i})$

*<proof>*

**lemma** *nth-list-segment*:

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $j \leq n$

**assumes**  $i \leq j$

**assumes**  $k < j - i$

**shows**  $(\text{list-segment } i\ j\ as) ! k = as ! (i + k)$

*<proof>*

## 4.2 Mapping the Carrier of a Ring to its 1-Dimensional Cartesian Power.

**context** *cring*

**begin**

**lemma** *R1-carI*:

**assumes**  $\text{length } as = 1$

**assumes**  $as!0 \in \text{carrier } R$

**shows**  $as \in \text{carrier } (R^1)$

*<proof>*

**abbreviation**(*input*) *to-R1* **where**

$to-R1\ a \equiv [a]$

**abbreviation**(*input*) *to-R* **::**  $'a\ \text{list} \Rightarrow 'a$  **where**

$to-R\ as \equiv as!0$

**lemma** *to-R1-to-R*:

**assumes**  $a \in \text{carrier } (R^1)$

**shows**  $to-R1\ (to-R\ a) = a$

*<proof>*

**lemma** *to-R-to-R1*:

**shows**  $to-R\ (to-R1\ a) = a$

*<proof>*

**lemma** *to-R1-closed*:

**assumes**  $a \in \text{carrier } R$

**shows**  $\text{to-R1 } a \in \text{carrier } (R^1)$

*<proof>*

**lemma** *to-R-pow-closed*:

**assumes**  $a \in \text{carrier } (R^1)$

**shows**  $\text{to-R } a \in \text{carrier } R$

*<proof>*

**lemma** *to-R1-intersection*:

**assumes**  $A \subseteq \text{carrier } R$

**assumes**  $B \subseteq \text{carrier } R$

**shows**  $\text{to-R1 } (A \cap B) = \text{to-R1 } A \cap \text{to-R1 } B$

*<proof>*

**lemma** *to-R1-finite*:

**assumes** *finite*  $A$

**shows** *finite*  $(\text{to-R1 } A)$

$\text{card } A = \text{card } (\text{to-R1 } A)$

*<proof>*

**lemma** *to-R1-carrier*:

$\text{to-R1 } (\text{carrier } R) = \text{carrier } (R^1)$

*<proof>*

**lemma** *to-R1-diff*:

$\text{to-R1 } (A - B) = \text{to-R1 } A - \text{to-R1 } B$

*<proof>*

**lemma** *to-R1-complement*:

**shows**  $\text{to-R1 } (\text{carrier } R - A) = \text{carrier } (R^1) - \text{to-R1 } A$

*<proof>*

**lemma** *to-R1-subset*:

**assumes**  $A \subseteq B$

**shows**  $\text{to-R1 } A \subseteq \text{to-R1 } B$

*<proof>*

**lemma** *to-R1-car-subset*:

**assumes**  $A \subseteq \text{carrier } R$

**shows**  $\text{to-R1 } A \subseteq \text{carrier } (R^1)$

*<proof>*

**end**

### 4.3 Simple Cartesian Products

**definition** *cartesian-product* :: ('a list) set  $\Rightarrow$  ('a list) set  $\Rightarrow$  ('a list) set **where**  
*cartesian-product*  $A B \equiv \{xs. \exists as \in A. \exists bs \in B. xs = as@bs\}$

**lemma** *cartesian-product-closed*:

**assumes**  $A \subseteq \text{carrier } (R^n)$

**assumes**  $B \subseteq \text{carrier } (R^m)$

**shows**  $\text{cartesian-product } A B \subseteq \text{carrier } (R^n + m)$

*<proof>*

**lemma** *cartesian-product-closed'*:

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $b \in \text{carrier } (R^m)$

**shows**  $(a@b) \in \text{carrier } (R^n + m)$

*<proof>*

**lemma** *cartesian-product-carrier*:

$\text{cartesian-product } (\text{carrier } (R^n)) (\text{carrier } (R^m)) = \text{carrier } (R^n + m)$

*<proof>*

**lemma** *cartesian-product-memI*:

**assumes**  $A \subseteq \text{carrier } (R^n)$

**assumes**  $B \subseteq \text{carrier } (R^m)$

**assumes**  $\text{take } n \ a \in A$

**assumes**  $\text{drop } n \ a \in B$

**shows**  $a \in \text{cartesian-product } A B$

*<proof>*

**lemma** *cartesian-product-memI'*:

**assumes**  $A \subseteq \text{carrier } (R^n)$

**assumes**  $B \subseteq \text{carrier } (R^m)$

**assumes**  $a \in A$

**assumes**  $b \in B$

**shows**  $a@b \in \text{cartesian-product } A B$

*<proof>*

**lemma** *cartesian-product-memE*:

**assumes**  $a \in \text{cartesian-product } A B$

**assumes**  $A \subseteq \text{carrier } (R^n)$

**shows**  $\text{take } n \ a \in A$

$\text{drop } n \ a \in B$

*<proof>*

**lemma** *cartesian-product-intersection*:

**assumes**  $A \subseteq \text{carrier } (R^n)$

**assumes**  $B \subseteq \text{carrier } (R^m)$

**assumes**  $C \subseteq \text{carrier } (R^n)$

**assumes**  $D \subseteq \text{carrier } (R^m)$

**shows**  $\text{cartesian-product } A B \cap \text{cartesian-product } C D = \text{cartesian-product } (A \cap C) (B \cap D)$

*<proof>*

**lemma** *cartesian-product-subsetI*:

**assumes**  $C \subseteq A$

**assumes**  $D \subseteq B$

**shows**  $\text{cartesian-product } C D \subseteq \text{cartesian-product } A B$

*<proof>*

**lemma** *cartesian-product-binary-union-right*:

**assumes**  $C \subseteq \text{carrier } (R^n)$

**assumes**  $D \subseteq \text{carrier } (R^n)$

**shows**  $\text{cartesian-product } A (C \cup D) = (\text{cartesian-product } A C) \cup (\text{cartesian-product } A D)$

*<proof>*

**lemma** *cartesian-product-binary-union-left*:

**assumes**  $C \subseteq \text{carrier } (R^n)$

**assumes**  $D \subseteq \text{carrier } (R^n)$

**shows**  $\text{cartesian-product } (C \cup D) A = (\text{cartesian-product } C A) \cup (\text{cartesian-product } D A)$

*<proof>*

**lemma** *cartesian-product-binary-intersection-right*:

**assumes**  $C \subseteq \text{carrier } (R^n)$

**assumes**  $D \subseteq \text{carrier } (R^n)$

**assumes**  $A \subseteq \text{carrier } (R^m)$

**shows**  $\text{cartesian-product } A (C \cap D) = (\text{cartesian-product } A C) \cap (\text{cartesian-product } A D)$

*<proof>*

**lemma** *cartesian-product-binary-intersection-left*:

**assumes**  $C \subseteq \text{carrier } (R^n)$

**assumes**  $D \subseteq \text{carrier } (R^n)$

**assumes**  $A \subseteq \text{carrier } (R^m)$

**shows**  $\text{cartesian-product } (C \cap D) A = (\text{cartesian-product } C A) \cap (\text{cartesian-product } D A)$

*<proof>*

**lemma** *cartesian-product-car-complement-right*:

**assumes**  $A \subseteq \text{carrier } (R^m)$

**shows**  $\text{carrier } (R^{n+m}) - \text{cartesian-product } (\text{carrier } (R^n)) A = \text{cartesian-product } (\text{carrier } (R^n)) ((\text{carrier } (R^m)) - A)$

*<proof>*

**lemma** *cartesian-product-car-complement-left*:

**assumes**  $A \subseteq \text{carrier } (R^n)$

**shows**  $\text{carrier } (R^{n+m}) - \text{cartesian-product } A (\text{carrier } (R^m)) = \text{cartesian-product } ((\text{carrier } (R^n)) - A) (\text{carrier } (R^m))$

*<proof>*

**lemma** *cartesian-product-complement-right*:

**assumes**  $B \subseteq \text{carrier } (R^m)$   
**assumes**  $A \subseteq \text{carrier } (R^n)$   
**shows**  $\text{cartesian-product } A (\text{carrier } (R^m)) - (\text{cartesian-product } A B) =$   
 $\text{cartesian-product } A ((\text{carrier } (R^m)) - B)$   
 ⟨proof⟩

**lemma** *cartesian-product-complement-left*:  
**assumes**  $B \subseteq \text{carrier } (R^m)$   
**assumes**  $A \subseteq \text{carrier } (R^n)$   
**shows**  $\text{cartesian-product } (\text{carrier } (R^m)) A - (\text{cartesian-product } B A) =$   
 $\text{cartesian-product } ((\text{carrier } (R^m)) - B) A$   
 ⟨proof⟩

**lemma** *cartesian-product-empty-right*:  
**assumes**  $A \subseteq \text{carrier } (R^n)$   
**assumes**  $B = \{\}\}$   
**shows**  $\text{cartesian-product } A B = A$   
 ⟨proof⟩

**lemma** *cartesian-product-empty-left*:  
**assumes**  $B \subseteq \text{carrier } (R^n)$   
**assumes**  $A = \{\}\}$   
**shows**  $\text{cartesian-product } A B = B$   
 ⟨proof⟩

#### 4.4 Cartesian Products at Arbitrary Indices

**definition**(in *ring*) *ring-pow-proj* ::  $\text{nat} \Rightarrow (\text{nat set}) \Rightarrow ('a \text{ list}) \Rightarrow ('a \text{ list})$   
 $(\pi, \cdot)$  **where**  
 $\text{ring-pow-proj } n S \equiv \text{restrict } (\text{project-at-indices } S) (\text{carrier } (R^n))$

The projection at an arbitrary index set

**lemma** *project-at-indices-closed*:  
**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $S \subseteq \text{indices-of } a$   
**shows**  $\pi_S a \in \text{carrier } (R^{\text{card } S})$   
 ⟨proof⟩

**lemma**(in *ring*) *ring-pow-proj-is-map*:  
**assumes**  $S \subseteq \{..<n\}$   
**shows**  $\pi_{n,S} \in \text{struct-maps } (R^n) (R^{\text{card } S})$   
 ⟨proof⟩

**lemma**(in *ring*) *project-at-indices-ring-pow-proj*:  
**assumes**  $x \in \text{carrier } (R^n)$   
**shows**  $\pi_S x = \pi_{n,S} x$   
 ⟨proof⟩

Cartesian products where the first factor  $A$  occurs at the entries of some ar-

bitrary index set. Note that this product isn't completely arbitrary because the entries of the factor of  $A$  still occurs in ascending order.

**definition** *twisted-cartesian-product* ( $Prod_{\cdot, \cdot}$ ,  $\cdot$ ) **where**  
*twisted-cartesian-product*  $S S' A B = \{a \mid \text{length } a = \text{card } S + \text{card } S' \wedge \pi_S a \in A \wedge \pi_{S'} a \in B\}$

**lemma** *twisted-cartesian-product-mem-length*:

**assumes**  $\text{card } S = n$   
**assumes**  $\text{card } S' = m$   
**assumes**  $a \in Prod_{S, S'} A B$   
**shows**  $\text{length } a = n + m$   
 $\langle \text{proof} \rangle$

**lemma** *twisted-cartesian-product-closed*:

**assumes**  $A \subseteq \text{carrier } (R^n)$   
**assumes**  $B \subseteq \text{carrier } (R^m)$   
**assumes**  $\text{card } S = n$   
**assumes**  $\text{card } S' = m$   
**assumes**  $S \cup S' = \{.. < n + m\}$   
**shows** *twisted-cartesian-product*  $S S' A B \subseteq \text{carrier } (R^{n + m})$   
 $\langle \text{proof} \rangle$

**lemma** *twisted-cartesian-product-memE*:

**assumes**  $a \in \text{twisted-cartesian-product } S S' A B$   
**shows**  $\pi_S a \in A \wedge \pi_{S'} a \in B$   
 $\langle \text{proof} \rangle$

**lemma** *twisted-cartesian-product-memI*:

**assumes**  $\pi_S a \in A$   
**assumes**  $\pi_{S'} a \in B$   
**assumes**  $\text{length } a = \text{card } S + \text{card } S'$   
**shows**  $a \in \text{twisted-cartesian-product } S S' A B$   
 $\langle \text{proof} \rangle$

**lemma** *twisted-cartesian-product-empty-left-factor*:

**assumes**  $A = \{\}$   
**shows** *twisted-cartesian-product*  $S S' A B = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *twisted-cartesian-product-empty-right-factor*:

**assumes**  $B = \{\}$   
**shows** *twisted-cartesian-product*  $S S' A B = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *twisted-cartesian-project-left*:

**assumes**  $A \subseteq \text{carrier } (R^n)$   
**assumes**  $B \subseteq \text{carrier } (R^m)$   
**assumes**  $A \neq \{\}$   
**assumes**  $B \neq \{\}$

**assumes**  $\text{card } S = n$   
**assumes**  $\text{card } S' = m$   
**assumes**  $S \cup S' = \{..<n + m\}$   
**shows**  $\pi_S ' (Prod_{S,S'} A B) = A$   
 <proof>

**lemma** *twisted-cartesian-product-swap*:  
**shows**  $(Prod_{S,S'} A B) = (Prod_{S',S} B A)$   
 <proof>

**lemma** *twisted-cartesian-project-right*:  
**assumes**  $A \subseteq \text{carrier } (R^n)$   
**assumes**  $B \subseteq \text{carrier } (R^m)$   
**assumes**  $A \neq \{\}$   
**assumes**  $B \neq \{\}$   
**assumes**  $\text{card } S = n$   
**assumes**  $\text{card } S' = m$   
**assumes**  $S \cup S' = \{..<n + m\}$   
**shows**  $\pi_{S'} ' (Prod_{S,S'} A B) = B$   
 <proof>

Cartesian products which send points  $a = (a_1, \dots, a_m)$  and  $b = (b_1, \dots, b_n)$   
 to the point  $(a_1, \dots, a_i, b_1, \dots, b_n, a_{i+1}, \dots, a_m)$

**definition** *splitting-permutation* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow$   
 $\text{nat} \Rightarrow \text{nat}$  **where**  
*splitting-permutation*  $l1\ l2\ i\ j =$  (if  $j < i$  then  $j$  else  
 (if  $i \leq j \wedge j < l1$  then  $(l2 + j)$  else  
 (if  $j < l1 + l2$  then  $j - l1 + i$  else  $j$ )))

**lemma** *splitting-permutation-case-1-unique*:  
**assumes**  $i \leq l1$   
**assumes**  $y < i$   
**assumes** *splitting-permutation*  $l1\ l2\ i\ j = y$   
**shows**  $j = y$   
 <proof>

**lemma** *splitting-permutation-case-1-exists*:  
**assumes**  $i \leq l1$   
**assumes**  $y < i$   
**shows** *splitting-permutation*  $l1\ l2\ i\ y = y$   
 <proof>

**lemma** *splitting-permutation-case-2-unique*:  
**assumes**  $i \leq l1$   
**assumes**  $i \leq y \wedge y < l2 + i$   
**assumes** *splitting-permutation*  $l1\ l2\ i\ j = y$   
**shows**  $j = y + l1 - i$   
 <proof>

**lemma** *splitting-permutation-case-2-exists*:

**assumes**  $i \leq l1$

**assumes**  $i \leq y \wedge y < l2 + i$

**shows** *splitting-permutation*  $l1\ l2\ i\ (y + l1 - i) = y$

*<proof>*

**lemma** *splitting-permutation-case-3-unique*:

**assumes**  $i \leq l1$

**assumes**  $l2 + i \leq y \wedge y < l1 + l2$

**assumes** *splitting-permutation*  $l1\ l2\ i\ j = y$

**shows**  $j = y - l2$

*<proof>*

**lemma** *splitting-permutation-case-3-exists*:

**assumes**  $i \leq l1$

**assumes**  $l2 + i \leq y \wedge y < l1 + l2$

**shows** *splitting-permutation*  $l1\ l2\ i\ (y - l2) = y$

*<proof>*

**lemma** *splitting-permutation-case-4-unique*:

**assumes**  $i \leq l1$

**assumes**  $l1 + l2 \leq y$

**assumes** *splitting-permutation*  $l1\ l2\ i\ j = y$

**shows**  $j = y$

*<proof>*

**lemma** *splitting-permutation-case-4-exists*:

**assumes**  $i \leq l1$

**assumes**  $l1 + l2 \leq y$

**shows** *splitting-permutation*  $l1\ l2\ i\ y = y$

*<proof>*

**lemma** *splitting-permutation-permutes*:

**assumes**  $i \leq l1$

**shows** (*splitting-permutation*  $l1\ l2\ i$ ) permutes  $\{.. < l1 + l2\}$

*<proof>*

**lemma** *splitting-permutation-action*:

**assumes**  $i \leq l1$

**assumes** *length*  $a1 = l1$

**assumes** *length*  $a2 = l2$

**shows** *permute-list* (*splitting-permutation*  $l1\ l2\ i$ ) ((*take*  $i\ a1$ ) @  $a2$  @ (*drop*  $i\ a1$ )) =

$a1@a2$

*<proof>*

**definition** *scp-permutation where*

*scp-permutation*  $l1\ l2\ i = \text{fun-inv } (\text{splitting-permutation } l1\ l2\ i)$

**lemma** *scp-permutation-action*:

**assumes**  $i \leq l1$

**assumes**  $\text{length } a1 = l1$

**assumes**  $\text{length } a2 = l2$

**shows**  $\text{permute-list } (\text{scp-permutation } l1 \ l2 \ i) \ (a1@a2) = ((\text{take } i \ a1) \ @ \ a2 \ @ \ (\text{drop } i \ a1))$   
*<proof>*

**lemma** *scp-permutes*:

**assumes**  $i \leq l1$

**shows**  $(\text{scp-permutation } l1 \ l2 \ i) \ \text{permutes } \{..<l1 + l2\}$

*<proof>*

**definition** *split-cartesian-product where*

*split-cartesian-product*  $l1 \ l2 \ i \ A \ B = \text{permute-list } (\text{scp-permutation } l1 \ l2 \ i) \ '(\text{cartesian-product } A \ B)$

**lemma** *split-cartesian-product-memI*:

**assumes**  $a1@a2 \in A$

**assumes**  $b \in B$

**assumes**  $A \subseteq \text{carrier } (R^{l1})$

**assumes**  $B \subseteq \text{carrier } (R^{l2})$

**assumes**  $\text{length } a1 = i$

**shows**  $a1@b@a2 \in \text{split-cartesian-product } l1 \ l2 \ i \ A \ B$

*<proof>*

**lemma** *split-cartesian-product-memI'*:

**assumes**  $a \in A$

**assumes**  $b \in B$

**assumes**  $A \subseteq \text{carrier } (R^{l1})$

**assumes**  $B \subseteq \text{carrier } (R^{l2})$

**assumes**  $i \leq l1$

**shows**  $(\text{take } i \ a)@b@(\text{drop } i \ a) \in \text{split-cartesian-product } l1 \ l2 \ i \ A \ B$

*<proof>*

**lemma** *split-cartesian-product-memE*:

**assumes**  $a \in \text{split-cartesian-product } l1 \ l2 \ i \ A \ B$

**assumes**  $A \subseteq \text{carrier } (R^{l1})$

**assumes**  $B \subseteq \text{carrier } (R^{l2})$

**assumes**  $i \leq l1$

**shows**  $(\text{take } i \ a)@(\text{drop } (i + l2) \ a) \in A$

$(\text{drop } i \ (\text{take } (i + l2) \ a)) \in B$

*<proof>*

**lemma** *split-cartesian-product-mem-length*:

**assumes**  $a \in \text{split-cartesian-product } l1 \ l2 \ i \ A \ B$

**assumes**  $A \subseteq \text{carrier } (R^{l1})$

**assumes**  $B \subseteq \text{carrier } (R^{l2})$

**assumes**  $i \leq l1$

**shows**  $\text{length } a = l1 + l2$   
 ⟨proof⟩

**lemma** *split-cartesian-product-memE'*:  
**assumes**  $a1@b@a2 \in \text{split-cartesian-product } l1 \ l2 \ i \ A \ B$   
**assumes**  $A \subseteq \text{carrier } (R^{l1})$   
**assumes**  $B \subseteq \text{carrier } (R^{l2})$   
**assumes**  $i \leq l1$   
**assumes**  $\text{length } a1 = i$   
**assumes**  $\text{length } b = l2$   
**assumes**  $\text{length } as = (l1 - i)$   
**shows**  $a1@a2 \in A$   
 $b \in B$   
 ⟨proof⟩

**lemma** *split-cartesian-product-closed*:  
**assumes**  $A \subseteq \text{carrier } (R^{l1})$   
**assumes**  $B \subseteq \text{carrier } (R^{l2})$   
**assumes**  $i \leq l1$   
**shows**  $\text{split-cartesian-product } l1 \ l2 \ i \ A \ B \subseteq \text{carrier } (R^{l1 + l2})$   
 ⟨proof⟩

General function for permuting the elements of a simple cartesian product:

**definition** *intersperse* ::  $(\text{nat} \Rightarrow \text{nat}) \Rightarrow 'a \ \text{tuple} \Rightarrow 'a \ \text{tuple} \Rightarrow 'a \ \text{tuple}$  **where**  
*intersperse*  $\sigma \ as \ bs = \text{permute-list } \sigma \ (as@bs)$

**lemma** *intersperseE*:  
**assumes**  $\sigma \ \text{permutes } (\{..<n\})$   
**assumes**  $\text{length } as + \text{length } bs = n$   
**shows**  $\text{length } (\text{intersperse } \sigma \ as \ bs) = n$   
 ⟨proof⟩

**lemma** *intersperseE'*:  
**assumes**  $\sigma \ \text{permutes } (\{..<n\})$   
**assumes**  $\text{length } as + \text{length } bs = n$   
**assumes**  $\text{length } as = k$   
**assumes**  $\sigma \ i < k$   
**shows**  $(\text{intersperse } \sigma \ as \ bs)! \ i = as ! \ \sigma \ i$   
 ⟨proof⟩

**lemma** *intersperseE''*:  
**assumes**  $\sigma \ \text{permutes } (\{..<n\})$   
**assumes**  $\text{length } as + \text{length } bs = n$   
**assumes**  $\text{length } as = k$   
**assumes**  $i < n$   
**assumes**  $\sigma \ i \geq k$   
**shows**  $(\text{intersperse } \sigma \ as \ bs)! \ i = bs ! \ ((\sigma \ i) - k)$   
 ⟨proof⟩

Some more lemmas about the `project_at_indices` function.

**lemma** *project-at-indices-consecutive-ind-length:*

**assumes**  $(i::nat) < j$   
**assumes**  $j \leq n$   
**assumes**  $length\ a = n$   
**shows**  $length\ (project-at-indices\ \{i..<j\}\ a) = j - i$   
 $\langle proof \rangle$

**lemma** *project-at-indices-consecutive-ind-length':*

**assumes**  $(i::nat) < j$   
**assumes**  $j \leq n$   
**assumes**  $a \in carrier\ (R^n)$   
**shows**  $length\ (project-at-indices\ \{i..<j\}\ a) = j - i$   
 $\langle proof \rangle$

**lemma** *sorted-list-of-set-from-up-to:*

**assumes**  $(i::nat) < j$   
**assumes**  $k < j - i$   
**shows**  $sorted-list-of-set\ \{i..<j\}\ !\ k = i + k$   
 $\langle proof \rangle$

**lemma** *nth-elem-consecutive-indices:*

**assumes**  $(i::nat) < j$   
**assumes**  $k < j - i$   
**shows**  $nth-elem\ \{i..<j\}\ k = i + k$   
 $\langle proof \rangle$

**lemma** *project-at-indices-consecutive-indices:*

**assumes**  $(i::nat) < j$   
**assumes**  $j \leq n$   
**assumes**  $length\ a = n$   
**assumes**  $k < j - i$   
**shows**  $(project-at-indices\ \{i..<j\}\ a)\ !\ k = a!\ (i + k)$   
 $\langle proof \rangle$

**lemma** *project-at-indices-consecutive-indices':*

**assumes**  $(i::nat) < j$   
**assumes**  $j \leq n$   
**assumes**  $a \in carrier\ (R^n)$   
**assumes**  $k < j - i$   
**shows**  $(project-at-indices\ \{i..<j\}\ a)\ !\ k = a!\ (i + k)$   
 $\langle proof \rangle$

**lemma** *tl-as-projection:*

**assumes**  $a \in carrier\ (R^n)$   
**shows**  $tl\ a = project-at-indices\ \{1::nat..<n\}\ a$   
 $\langle proof \rangle$

## 4.5 Function Rings on Cartesian Powers

Complement operator

**definition** *ring-pow-comp* :: ('a, 'b) ring-scheme  $\Rightarrow$  arity  $\Rightarrow$  'a tuple set  $\Rightarrow$  'a tuple set **where**

*ring-pow-comp*  $R$   $n$   $S \equiv \text{carrier } (R^n) - S$

**lemma** *ring-pow-comp-closed*:

*ring-pow-comp*  $R$   $n$   $S \subseteq \text{carrier } (R^n)$

*<proof>*

**lemma** *ring-pow-comp-disjoint*:

*ring-pow-comp*  $R$   $n$   $S \cap S = \{\}$

*<proof>*

**lemma** *ring-pow-comp-union*:

**assumes**  $S \subseteq \text{carrier } (R^n)$

**shows**  $(\text{ring-pow-comp } R \ n \ S) \cup S = \text{carrier } (R^n)$

*<proof>*

**lemma** *ring-pow-comp-carrier*:

*ring-pow-comp*  $R$   $n$   $(\text{carrier } (R^n)) = \{\}$

*<proof>*

**lemma** *ring-pow-comp-empty*:

*ring-pow-comp*  $R$   $n$   $\{\} = (\text{carrier } (R^n))$

*<proof>*

**lemma** *ring-pow-comp-demorgans*:

**assumes**  $A \subseteq \text{carrier } (R^n)$

**assumes**  $B \subseteq \text{carrier } (R^n)$

**shows**  $\text{ring-pow-comp } R \ n \ (A \cup B) = (\text{ring-pow-comp } R \ n \ A) \cap (\text{ring-pow-comp } R \ n \ B)$

*<proof>*

**lemma** *ring-pow-comp-demorgans'*:

**assumes**  $A \subseteq \text{carrier } (R^n)$

**assumes**  $B \subseteq \text{carrier } (R^n)$

**shows**  $\text{ring-pow-comp } R \ n \ (A \cap B) = (\text{ring-pow-comp } R \ n \ A) \cup (\text{ring-pow-comp } R \ n \ B)$

*<proof>*

**lemma** *ring-pow-comp-inv*:

**assumes**  $A \subseteq \text{carrier } (R^n)$

**shows**  $\text{ring-pow-comp } R \ n \ (\text{ring-pow-comp } R \ n \ A) = A$

*<proof>*

The function ring defined on the powers of a ring:

**abbreviation**(*input*) *ring-pow-function-ring* (*Fun.* -) **where**

*ring-pow-function-ring*  $n R \equiv \text{function-ring } (\text{carrier } (R^n)) R$

Partial function application. Given a function  $f(x_1, \dots, x_{n+1})$ , an index  $i$  and a point  $a \in \text{carrier } R$  returns the function  $(x_1, \dots, x_n) \mapsto f(x_1, \dots, x_{i-1}, a, x_i, \dots, x_n)$

**lemma** *ring-pow-function-ring-car-memE*:

**assumes**  $f \in \text{carrier } (\text{Fun}_n R)$   
**shows**  $f \in \text{extensional } (\text{carrier } (R^n))$   
 $f \in \text{carrier } (R^n) \rightarrow \text{carrier } R$   
*<proof>*

**definition** *partial-eval* ::  $('a, 'b) \text{ ring-scheme} \Rightarrow \text{arity} \Rightarrow \text{nat} \Rightarrow ('a \text{ list} \Rightarrow 'a) \Rightarrow 'a \Rightarrow ('a \text{ list} \Rightarrow 'a)$  **where**  
*partial-eval*  $R m n f c = \text{restrict } (\lambda \text{ as. } f (\text{insert-at-index as } c n)) (\text{carrier } (R^m))$

**context** *ring*  
**begin**

**lemma** *function-ring-car-mem-closed*:

**assumes**  $f \in \text{carrier } (\text{function-ring } S R)$   
**assumes**  $s \in S$   
**shows**  $f s \in \text{carrier } R$   
*<proof>*

**lemma** *function-ring-car-mem-closed'*:

**assumes**  $f \in \text{carrier } (\text{Fun}_{\text{Suc } k} R)$   
**assumes**  $s \in \text{carrier } (R^{\text{Suc } k})$   
**shows**  $f s \in \text{carrier } R$   
*<proof>*

**lemma**(**in** *ring*) *partial-eval-domain*:

**assumes**  $f \in \text{carrier } (\text{Fun}_{\text{Suc } k} R)$   
**assumes**  $a \in \text{carrier } R$   
**assumes**  $n \leq k$   
**shows**  $(\text{partial-eval } R k n f a) \in \text{carrier } (\text{Fun}_k R)$   
*<proof>*

Pullbacks preserve ring power functions

**lemma** *fun-struct-maps*:

*struct-maps*  $(R^n) R = \text{carrier } (\text{Fun}_n R)$   
*<proof>*

**lemma** *pullback-fun-closed*:

**assumes**  $f \in \text{struct-maps } (R^n) (R^m)$   
**assumes**  $g \in \text{carrier } (\text{Fun}_m R)$   
**shows**  $\text{pullback } (R^n) f g \in \text{carrier } (\text{Fun}_n R)$   
*<proof>*

**end**

Includes  $R^{|S|}$  into  $R^n$  by pulling back along the projection  $R^n \mapsto R^{|S|}$  at indices  $S$

**context** *ring*  
**begin**

**definition**(in *ring*) *ring-pow-inc* :: (*nat set*)  $\Rightarrow$  *arity*  $\Rightarrow$  (*'a tuple*  $\Rightarrow$  *'a*)  $\Rightarrow$  (*'a tuple*  $\Rightarrow$  *'a*) **where**  
*ring-pow-inc*  $S$   $n$   $f$  = *pullback* ( $R^n$ ) ( $\pi_{n,S}$ )  $f$

**lemma** *ring-pow-inc-is-fun*:  
**assumes**  $S \subseteq \{..<n\}$   
**assumes**  $f \in \text{carrier } (\text{Fun}_{\text{card } S} R)$   
**shows** *ring-pow-inc*  $S$   $n$   $f \in \text{carrier } (\text{Fun}_n R)$   
(*proof*)

The "standard" inclusion of powers of function rings into one another

**abbreviation**(*input*) *std-proj*:: *nat*  $\Rightarrow$  *nat*  $\Rightarrow$  (*'a list*)  $\Rightarrow$  (*'a list*) **where**  
*std-proj*  $n$   $m \equiv \text{ring-pow-proj } n$  ( $\{..<m\}$ )

**lemma** *std-proj-id*:  
**assumes**  $m \leq n$   
**assumes**  $as \in \text{carrier } (R^n)$   
**assumes**  $i < m$   
**shows** *std-proj*  $n$   $m$   $as ! i = as ! i$   
(*proof*)

**abbreviation**(*input*) *std-inc*:: *nat*  $\Rightarrow$  *nat*  $\Rightarrow$  (*'a list*  $\Rightarrow$  *'a*)  $\Rightarrow$  (*'a list*  $\Rightarrow$  *'a*)  
**where**  
*std-inc*  $n$   $m$   $f \equiv \text{ring-pow-inc } (\{..<m\})$   $n$   $f$

**lemma** *std-proj-is-map[simp]*:  
**assumes**  $m \leq n$   
**shows** *std-proj*  $n$   $m \in \text{struct-maps } (R^n) (R^m)$   
(*proof*)

**end**

## 4.6 Coordinate Functions

**definition** *var* :: (*'a*, *'b*) *ring-scheme*  $\Rightarrow$  *nat*  $\Rightarrow$  *nat*  $\Rightarrow$  (*'a list*  $\Rightarrow$  *'a*) **where**  
*var*  $R$   $n$   $i = \text{restrict } (\lambda x. x!i)$  (*carrier* ( $R^n$ ))

**context** *ring*  
**begin**

**lemma** *var-in-car*:  
**assumes**  $i < n$   
**shows** *var*  $R$   $n$   $i \in \text{carrier } (\text{Fun}_n R)$   
(*proof*)

**lemma** *varE[simp]*:

**assumes**  $i < n$

**assumes**  $x \in \text{carrier } (R^n)$

**shows**  $\text{var } R \ n \ i \ x = x \ ! \ i$

*<proof>*

**lemma** *std-inc-of-var*:

**assumes**  $i < n$

**assumes**  $n \leq m$

**shows**  $\text{std-inc } m \ n \ (\text{var } R \ n \ i) = (\text{var } R \ m \ i)$

*<proof>*

**abbreviation** *variable* (**v.**) **where**

*variable*  $n \ i \equiv \text{var } R \ n \ i$

**end**

**definition** *var-set* ::  $('a, 'b)$  ring-scheme  $\Rightarrow$  nat  $\Rightarrow$   $('a$  list  $\Rightarrow 'a)$  set **where**

*var-set*  $R \ n = \text{var } R \ n \ \{..\<n\}$

**lemma** *var-setE*:

**assumes**  $f \in \text{var-set } R \ n$

**obtains**  $i$  **where**  $f = \text{var } R \ n \ i \wedge i \in \{..\<n\}$

*<proof>*

**lemma** *var-setI*:

**assumes**  $i \in \{..\<n\}$

**assumes**  $f = \text{var } R \ n \ i$

**shows**  $f \in \text{var-set } R \ n$

*<proof>*

**context** *ring*

**begin**

**lemma** *var-set-in-fun-ring-car*:

**shows**  $\text{var-set } R \ n \subseteq \text{carrier } \text{Fun}_n \ R$

*<proof>*

**end**

## 4.7 Graphs of functions

**definition** *fun-graph*::  $('a, 'b)$  ring-scheme  $\Rightarrow$  nat  $\Rightarrow$   $('a$  list  $\Rightarrow 'a) \Rightarrow 'a$  list set **where**

*fun-graph*  $R \ n \ f = \{as. (\exists x \in \text{carrier } (R^n). as = x @ [f \ x])\}$

**context** *ring*  
**begin**

**lemma** *function-ring-car-memE*:  
**assumes**  $f \in \text{carrier } (\text{Fun}_n R)$   
**assumes**  $a \in \text{carrier } (R^n)$   
**shows**  $f a \in \text{carrier } R$   
 $\langle \text{proof} \rangle$

**lemma** *graph-range*:  
**assumes**  $f \in \text{carrier } (\text{Fun}_n R)$   
**shows**  $\text{fun-graph } R \ n \ f \subseteq \text{carrier } (R^{\text{Suc } n})$   
 $\langle \text{proof} \rangle$

**lemma** *fun-graph-memE*:  
**assumes**  $f \in \text{carrier } (\text{Fun}_n R)$   
**assumes**  $p \in \text{fun-graph } R \ n \ f$   
**shows**  $(\text{take } n \ p) \in \text{carrier } (R^n)$   
 $\langle \text{proof} \rangle$

**lemma** *fun-graph-memE'*:  
**assumes**  $f \in \text{carrier } (\text{Fun}_n R)$   
**assumes**  $p \in \text{fun-graph } R \ n \ f$   
**shows**  $f (\text{take } n \ p) = p!n$   
 $\langle \text{proof} \rangle$

apply a function  $f$  to the tuple consisting of the first  $n$  indices, leaving the remaining indices unchanged

**definition** *partial-image*  $:: \text{arity} \Rightarrow ('c \ \text{list} \Rightarrow 'c) \Rightarrow 'c \ \text{list} \Rightarrow 'c \ \text{list}$  **where**  
*partial-image*  $n \ f \ as = (f (\text{take } n \ as)) \# (\text{drop } n \ as)$

**lemma** *partial-image-range*:  
**assumes**  $f \in \text{carrier } (\text{Fun}_n R)$   
**assumes**  $m \geq n$   
**assumes**  $as \in \text{carrier } (R^m)$   
**shows**  $\text{partial-image } n \ f \ as \in \text{carrier } (R^{m - n + 1})$   
 $\langle \text{proof} \rangle$

**end**

## 5 Coordinate Rings on Cartesian Powers

### 5.1 Basic Facts and Definitions

**locale** *cring-coord-rings* = *UP-cring* +  
**assumes** *one-neq-zero*:  $\mathbf{1} \neq \mathbf{0}$

coordinate polynomial ring in  $n$  variables over a commutative ring

**definition** *coord-ring* :: ('a, 'b) ring-scheme  $\Rightarrow$  nat  $\Rightarrow$  ('a, ('a, nat) mvar-poly)  
 module

(- [ $\mathcal{X}$ ] 80) **where**  $R[\mathcal{X}_n] \equiv \text{Pring } R \{..< n::\text{nat}\}$

**sublocale** *cring-coord-rings* < *cring-functions*  $R$  carrier ( $R^n$ )  $\text{Fun}_n R$   
 <proof>

**sublocale** *cring-coord-rings* <  $MP?$ : *cring*  $R[\mathcal{X}_n]$   
 <proof>

**sublocale** *cring-coord-rings* <  $F?$ : *cring*  $\text{Fun}_n R$   
 <proof>

**context** *cring-coord-rings*  
**begin**

**lemma** *coord-cring-cring*:  
*cring* ( $R[\mathcal{X}_n]$ ) <proof>

coordinate constant functions

**abbreviation**(*input*) *coord-const* :: 'a  $\Rightarrow$  ('a, nat) mvar-poly **where**  
*coord-const*  $k \equiv \text{ring.indexed-const } R k$

**lemma** *coord-const-ring-hom*:  
*ring-hom-ring*  $R$  ( $R[\mathcal{X}_n]$ ) *coord-const*  
 <proof>

coordinate functions

**lemma** *pvar-closed*:  
**assumes**  $i < n$   
**shows**  $pvar R i \in \text{carrier } (R[\mathcal{X}_n])$   
 <proof>

relationship between multiplication by a variable and index multiplication

**lemma** *pvar-indexed-pmult*:  
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $(p \otimes i) = p \otimes_{R[\mathcal{X}_n]} pvar R i$   
 <proof>

**lemma** *coord-ring-cfs-closed*:  
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $p m \in \text{carrier } R$   
 <proof>

**lemma** *coord-ring-plus*:  
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $Q \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $(p \oplus_{R[\mathcal{X}_n]} Q) m = p m \oplus Q m$

*<proof>*

**lemma** *coord-ring-uminus*:  
assumes  $p \in \text{carrier } (R[\mathcal{X}_n])$   
shows  $(\ominus_{R[\mathcal{X}_n]} p) m = \ominus (p m)$   
*<proof>*

**lemma** *coord-ring-minus*:  
assumes  $p \in \text{carrier } (R[\mathcal{X}_n])$   
assumes  $Q \in \text{carrier } (R[\mathcal{X}_n])$   
shows  $(p \ominus_{R[\mathcal{X}_n]} Q) m = p m \ominus Q m$   
*<proof>*

**lemma** *coord-ring-one*:  
 $\mathbf{1}_{R[\mathcal{X}_n]} m = (\text{if } m = \{\#\} \text{ then } \mathbf{1} \text{ else } \mathbf{0})$   
*<proof>*

**lemma** *coord-ring-zero*:  
 $\mathbf{0}_{R[\mathcal{X}_n]} m = \mathbf{0}$   
*<proof>*

Evaluation of a polynomial at a point

**end**

**abbreviation**(*input*) *point-to-eval-map* **where**  
*point-to-eval-map*  $R \text{ as} \equiv (\lambda i. (\text{if } i < \text{length as then as ! } i \text{ else } \mathbf{0}_R))$

**definition** *eval-at-point* :: ('a, 'b) *ring-scheme*  $\Rightarrow$  'a *list*  $\Rightarrow$  ('a, nat) *mvar-poly*  $\Rightarrow$  'a **where**  
*eval-at-point*  $R \text{ as } p \equiv \text{total-eval } R (\lambda i. (\text{if } i < \text{length as then as ! } i \text{ else } \mathbf{0}_R)) p$

**lemma**(**in** *cring-coord-rings*) *eval-at-point-factored*:  
*eval-at-point*  $R \text{ as } p = \text{total-eval } R (\text{point-to-eval-map } R \text{ as}) p$   
*<proof>*

## 5.2 Total Evaluation of a Polynomial

**abbreviation**(*input*) *eval-at-poly* **where**  
*eval-at-poly*  $R p \text{ as} \equiv \text{eval-at-point } R \text{ as } p$

evaluation of coordinate polynomials

**context** *cring-coord-rings*  
**begin**

**lemma** *eval-at-point-closed*:  
assumes  $a \in \text{carrier } (R^n)$   
assumes  $p \in \text{carrier } (R[\mathcal{X}_n])$

**shows** *eval-at-point*  $R$   $a$   $p \in \text{carrier } R$   
*<proof>*

**lemma** *eval-pvar*:

**assumes**  $i < (n::\text{nat})$   
**assumes**  $a \in \text{carrier } (R^n)$   
**shows** *eval-at-point*  $R$   $a$  (*pvar*  $R$   $i$ ) =  $a!i$   
*<proof>*

**lemma** *eval-at-point-const*:

**assumes**  $k \in \text{carrier } R$   
**assumes**  $a \in \text{carrier } (R^n)$   
**shows** *eval-at-point*  $R$   $a$  (*R.indexed-const*  $k$ ) =  $k$   
*<proof>*

**lemma** *eval-at-point-add*:

**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $B \in \text{carrier } (\text{coord-ring } R \ n)$   
**shows** *eval-at-point*  $R$   $a$  ( $A \oplus_{\text{coord-ring } R \ n} B$ ) =  
*eval-at-point*  $R$   $a$   $A \oplus_R$  *eval-at-point*  $R$   $a$   $B$   
*<proof>*

**lemma** *eval-at-point-mult*:

**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $B \in \text{carrier } ((R[\mathcal{X}_n]))$   
**shows** *eval-at-point*  $R$   $a$  ( $A \otimes_{R[\mathcal{X}_n]} B$ ) =  
*eval-at-point*  $R$   $a$   $A \otimes_R$  *eval-at-point*  $R$   $a$   $B$   
*<proof>*

**lemma** *eval-at-point-indexed-pmult*:

**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $i < n$   
**shows** *eval-at-point*  $R$   $a$  ( $A \otimes i$ ) =  
*eval-at-point*  $R$   $a$   $A \otimes_R$  ( $a!i$ )  
*<proof>*

**lemma** *eval-at-point-ring-hom*:

**assumes**  $a \in \text{carrier } (R^n)$   
**shows** *ring-hom-ring* (*coord-ring*  $R$   $I$ )  $R$  (*eval-at-point*  $R$   $a$ )  
*<proof>*

**lemma** *eval-at-point-scalar-mult*:

**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $k \in \text{carrier } R$   
**shows** *eval-at-point*  $R$   $a$  (*poly-scalar-mult*  $R$   $k$   $A$ ) =  $k \otimes_R$  (*eval-at-point*  $R$   $a$   $A$ )

*<proof>*

**lemma** *eval-at-point-smult:*

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $k \in \text{carrier } R$

**shows**  $\text{eval-at-point } R \ a \ (k \odot_{R[\mathcal{X}_n]} A) = k \otimes_R (\text{eval-at-point } R \ a \ A)$

*<proof>*

**lemma** *eval-at-point-subtract:*

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $B \in \text{carrier } (\text{coord-ring } R \ n)$

**shows**  $\text{eval-at-point } R \ a \ (A \ominus_{\text{coord-ring } R \ n} B) =$   
 $\text{eval-at-point } R \ a \ A \ominus_R \text{eval-at-point } R \ a \ B$

*<proof>*

**lemma** *eval-at-point-a-inv:*

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $B \in \text{carrier } (\text{coord-ring } R \ n)$

**shows**  $\text{eval-at-point } R \ a \ (\ominus_{R[\mathcal{X}_n]} B) = \ominus_R \text{eval-at-point } R \ a \ B$

*<proof>*

**lemma** *eval-at-point-nat-pow:*

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{eval-at-point } R \ a \ (A[\uparrow]_{R[\mathcal{X}_n]}(k::\text{nat})) = (\text{eval-at-point } R \ a \ A)[\uparrow]k$

*<proof>*

**end**

### 5.3 Partial Evaluation of a Polynomial

**definition** *coord-partial-eval* ::

$('a, 'b) \text{ ring-scheme} \Rightarrow \text{nat set} \Rightarrow 'a \text{ list} \Rightarrow ('a, \text{nat}) \text{ mvar-poly} \Rightarrow ('a, \text{nat})$

*mvar-poly* **where**

$\text{coord-partial-eval } R \ S \ as = \text{poly-eval } R \ S \ (\text{point-to-eval-map } R \ as)$

**context** *cring-coord-rings*

**begin**

**lemma** *point-to-eval-map-closed:*

**assumes**  $as \in \text{carrier } (R^n)$

**shows**  $\text{closed-fun } R \ (\text{point-to-eval-map } R \ as)$

*<proof>*

**lemma** *coord-partial-eval-hom:*

**assumes**  $as \in \text{carrier } (R^n)$

**shows**  $\text{coord-partial-eval } R \ S \ as \in \text{ring-hom } (R[\mathcal{X}_n]) \ (R[\mathcal{X}_n])$

*<proof>*

**lemma** *coord-partial-eval-hom'*:

**assumes**  $as \in \text{carrier } (R^n)$

**shows**  $\text{coord-partial-eval } R \ S \ as \in \text{ring-hom } (R[\mathcal{X}_n]) \ (\text{Pring } R \ (\{..\<n\} - S))$

*<proof>*

**lemma** *coord-partial-eval-closed*:

**assumes**  $S \subseteq \{..\<n\}$

**assumes**  $\{..\<n\} - S \subseteq I$

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{coord-partial-eval } R \ S \ as \ p \in \text{carrier } (\text{Pring } R \ I)$

*<proof>*

**lemma** *coord-partial-eval-add*:

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $Q \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{coord-partial-eval } R \ S \ as \ (p \oplus_{(R[\mathcal{X}_n])} Q) =$

$(\text{coord-partial-eval } R \ S \ as \ p) \oplus_{(R[\mathcal{X}_n])} (\text{coord-partial-eval } R \ S \ as \ Q)$

*<proof>*

**lemma** *coord-partial-eval-mult*:

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $Q \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{coord-partial-eval } R \ S \ as \ (p \otimes_{(R[\mathcal{X}_n])} Q) =$

$(\text{coord-partial-eval } R \ S \ as \ p) \otimes_{(R[\mathcal{X}_n])} (\text{coord-partial-eval } R \ S \ as \ Q)$

*<proof>*

**lemma** *coord-partial-eval-pvar*:

**assumes**  $1 \neq 0$

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $i \in S \cap \{..\<n\}$

**shows**  $\text{coord-partial-eval } R \ S \ as \ (\text{pvar } R \ i) = \text{coord-const } (as!i)$

*<proof>*

**lemma** *coord-partial-eval-pvar'*:

**assumes**  $1 \neq 0$

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $i \notin S$

**shows**  $\text{coord-partial-eval } R \ S \ as \ (\text{pvar } R \ i) = (\text{pvar } R \ i)$

*<proof>*

## 5.4 An induction rule for coordinate rings

**lemma** *coord-ring-induct*:

**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $\bigwedge a. a \in \text{carrier } R \implies p (\text{coord-const } a)$   
**assumes**  $\bigwedge i Q. Q \in \text{carrier } (R[\mathcal{X}_n]) \implies p Q \implies i < n \implies p (Q \otimes_{R[\mathcal{X}_n]} \text{pvar } R i)$   
**assumes**  $\bigwedge Q0 Q1. Q0 \in \text{carrier } (R[\mathcal{X}_n]) \implies Q1 \in \text{carrier } (R[\mathcal{X}_n]) \implies p Q0 \implies p Q1 \implies p (Q0 \oplus_{R[\mathcal{X}_n]} Q1)$   
**shows**  $p A$   
 $\langle \text{proof} \rangle$

**end**

## 5.5 Algebraic Sets in Cartesian Powers

### 5.5.1 The Zero Set of a Single Polynomial

**definition** *zero-set* :: ('a, 'b) ring-scheme  $\Rightarrow$  nat  $\Rightarrow$  ('a, nat) mvar-poly  $\Rightarrow$  'a list set

(V1) **where**  
*zero-set*  $R$   $n$   $p = \{a \in \text{carrier } (R^n). \text{eval-at-point } R a p = \mathbf{0}_R\}$

**context** *cring-coord-rings*

**begin**

**lemma** *zero-setI*:

**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $\text{eval-at-point } R a p = \mathbf{0}_R$   
**shows**  $a \in \text{zero-set } R n p$   
 $\langle \text{proof} \rangle$

**lemma** *zero-setE*:

**assumes**  $a \in \text{zero-set } R n p$   
**shows**  $a \in \text{carrier } (R^n)$   
 $\text{eval-at-point } R a p = \mathbf{0}_R$   
 $\langle \text{proof} \rangle$

**lemma** *zero-set-closed*:

$\text{zero-set } R n p \subseteq \text{carrier } (R^n)$   
 $\langle \text{proof} \rangle$

**end**

### 5.5.2 The Zero Set of a Collection of Polynomials

**definition** *affine-alg-set* :: ('a, 'b) ring-scheme  $\Rightarrow$  nat  $\Rightarrow$  ('a, nat) mvar-poly set  $\Rightarrow$  'a list set

**where** *affine-alg-set*  $R$   $n$   $as = \{a \in \text{carrier } (R^n). \forall b \in as. a \in (\text{zero-set } R n b)\}$

**context** *cring-coord-rings*

**begin**

**lemma** *affine-alg-set-empty*:

*affine-alg-set*  $R$   $n$   $\{\}$  = *carrier*  $(R^n)$   
*<proof>*

**lemma** *affine-alg-set-subset-zero-set*:  
**assumes**  $b \in as$   
**shows** *affine-alg-set*  $R$   $n$   $as \subseteq (\text{zero-set } R$   $n$   $b)$   
*<proof>*

**lemma**(*in cring-coord-rings*) *affine-alg-set-memE*:  
**assumes**  $b \in as$   
**assumes**  $a \in \text{affine-alg-set } R$   $n$   $as$   
**shows** *eval-at-poly*  $R$   $b$   $a = \mathbf{0}$   
*<proof>*

**lemma** *affine-alg-set-subset*:  
**assumes**  $as \subseteq bs$   
**shows** *affine-alg-set*  $R$   $n$   $bs \subseteq \text{affine-alg-set } R$   $n$   $as$   
*<proof>*

**lemma** *affine-alg-set-empty-set*:  
**assumes**  $as = \{\}$   
**shows** *affine-alg-set*  $R$   $n$   $as = \text{carrier } (R^n)$   
*<proof>*

**lemma** *affine-alg-set-closed*:  
**shows** *affine-alg-set*  $R$   $n$   $as \subseteq \text{carrier } (R^n)$   
*<proof>*

**lemma** *affine-alg-set-singleton*:  
*affine-alg-set*  $R$   $n$   $\{a\} = \text{zero-set } R$   $n$   $a$   
*<proof>*

**lemma** *affine-alg-set-insert*:  
*affine-alg-set*  $R$   $n$  (*insert*  $a$   $A$ ) = *zero-set*  $R$   $n$   $a \cap (\text{affine-alg-set } R$   $n$   $A)$   
*<proof>*

**lemma** *affine-alg-set-intersect*:  
*affine-alg-set*  $R$   $n$   $(A \cup B) = (\text{affine-alg-set } R$   $n$   $A) \cap (\text{affine-alg-set } R$   $n$   $B)$   
*<proof>*

**lemma** *affine-alg-set-memI*:  
**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $\bigwedge p. p \in B \implies \text{eval-at-point } R$   $a$   $p = \mathbf{0}$   
**shows**  $a \in (\text{affine-alg-set } R$   $n$   $B)$   
*<proof>*

**lemma** *affine-alg-set-not-memE*:  
**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $a \notin (\text{affine-alg-set } R$   $n$   $B)$

**shows**  $\exists b \in B. \text{eval-at-poly } R \ b \ a \neq \mathbf{0}$   
 ⟨*proof*⟩

### 5.5.3 Finite Unions and Intersections of Algebraic Sets are Algebraic

The product set of two sets in an arbitrary ring. That is, the set  $\{xy \mid x \in A \wedge y \in B\}$  for two sets  $A, B$ .

**definition**(in *ring*) *prod-set* :: 'a set  $\Rightarrow$  'a set  $\Rightarrow$  'a set **where**  
*prod-set as bs* =  $(\lambda x. \text{fst } x \otimes \text{snd } x) \text{ ` } (as \times bs)$

**lemma**(in *ring*) *prod-setI*:  
**assumes**  $c \in \text{prod-set } as \ bs$   
**shows**  $\exists a \in as. \exists b \in bs. c = a \otimes b$   
 ⟨*proof*⟩

**lemma**(in *ring*) *prod-set-closed*:  
**assumes**  $as \subseteq \text{carrier } R$   
**assumes**  $bs \subseteq \text{carrier } R$   
**shows**  $\text{prod-set } as \ bs \subseteq \text{carrier } R$   
 ⟨*proof*⟩

The set of products of elements from two finite sets is again finite.

**lemma**(in *ring*) *prod-set-finite*:  
**assumes** *finite as*  
**assumes** *finite bs*  
**shows** *finite (prod-set as bs)*  $\text{card } (\text{prod-set } as \ bs) \leq \text{card } as * \text{card } bs$   
 ⟨*proof*⟩

**definition** *poly-prod-set* **where**  
*poly-prod-set n as bs* = *ring.prod-set*  $(R[\mathcal{X}_n]) \ as \ bs$

**lemma** *poly-prod-setE*:  
**assumes**  $c \in \text{poly-prod-set } n \ as \ bs$   
**shows**  $\exists a \in as. \exists b \in bs. c = a \otimes_{R[\mathcal{X}_n]} b$   
 ⟨*proof*⟩

**lemma** *poly-prod-setI*:  
**assumes**  $a \in as$   
**assumes**  $b \in bs$   
**shows**  $a \otimes_{R[\mathcal{X}_n]} b \in \text{poly-prod-set } n \ as \ bs$   
 ⟨*proof*⟩

**lemma** *poly-prod-set-closed*:  
**assumes**  $as \subseteq \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $bs \subseteq \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $\text{poly-prod-set } n \ as \ bs \subseteq \text{carrier } (R[\mathcal{X}_n])$   
 ⟨*proof*⟩

**lemma** *poly-prod-set-finite*:  
**assumes** *finite as*  
**assumes** *finite bs*  
**shows** *finite (poly-prod-set n as bs) card (poly-prod-set n as bs) ≤ card as \* card bs*

*<proof>*

**end**

**locale** *domain-coord-rings = cring-coord-rings + domain*

**lemma**(**in** *domain-coord-rings*) *poly-prod-set-algebraic-set*:  
**assumes** *as ⊆ carrier (R[X<sub>n</sub>])*  
**assumes** *bs ⊆ carrier (R[X<sub>n</sub>])*  
**shows** *affine-alg-set R n as ∪ affine-alg-set R n bs = affine-alg-set R n (poly-prod-set n as bs)*  
*<proof>*

**definition** *is-algebraic* :: (*'a, 'b*) *ring-scheme* ⇒ *nat* ⇒ *'a list set* ⇒ *bool* **where**  
*is-algebraic R n S = (∃ ps. finite ps ∧ ps ⊆ carrier (R[X<sub>n</sub>]) ∧ S = affine-alg-set R n ps)*

**context** *cring-coord-rings*  
**begin**

**lemma** *is-algebraicE*:  
**assumes** *is-algebraic R n S*  
**obtains** *ps* **where** *finite ps ps ⊆ carrier (R[X<sub>n</sub>]) S = affine-alg-set R n ps*  
*<proof>*

**lemma** *is-algebraicI*:  
**assumes** *finite ps*  
**assumes** *ps ⊆ carrier (R[X<sub>n</sub>])*  
**assumes** *S = affine-alg-set R n ps*  
**shows** *is-algebraic R n S*  
*<proof>*

**lemma** *is-algebraicI'*:  
**assumes** *p ∈ carrier (R[X<sub>n</sub>])*  
**assumes** *S = zero-set R n p*  
**shows** *is-algebraic R n S*  
*<proof>*

**end**

**definition** *alg-sets* :: *arity* ⇒ (*'a, 'b*) *ring-scheme* ⇒ (*'a list set*) *set* **where**  
*alg-sets n R = {S. is-algebraic R n S}*

**context** *cring-coord-rings*

**begin**

**lemma** *intersection-is-alg:*

**assumes** *is-algebraic*  $R$   $n$   $A$

**assumes** *is-algebraic*  $R$   $n$   $B$

**shows** *is-algebraic*  $R$   $n$   $(A \cap B)$

*<proof>*

**lemma**(**in** *domain-coord-rings*) *union-is-alg:*

**assumes** *is-algebraic*  $R$   $n$   $A$

**assumes** *is-algebraic*  $R$   $n$   $B$

**shows** *is-algebraic*  $R$   $n$   $(A \cup B)$

*<proof>*

**lemma** *zero-set-zero:*

*zero-set*  $R$   $n$   $\mathbf{0}_{R[\mathcal{X}_n]}$  = *carrier*  $(R^n)$

*<proof>*

**lemma** *affine-alg-set-set:*

*affine-alg-set*  $R$   $n$   $\{\mathbf{0}_{R[\mathcal{X}_n]}\}$  = *carrier*  $(R^n)$

*<proof>*

**lemma** *car-is-alg:*

*is-algebraic*  $R$   $n$  (*carrier*  $(R^n)$ )

*<proof>*

**lemma** *zero-set-nonzero-constant:*

**assumes**  $a \neq \mathbf{0}$

**assumes**  $a \in$  *carrier*  $R$

**shows** *zero-set*  $R$   $n$  (*coord-const*  $a$ ) =  $\{\}$

*<proof>*

**lemma** *zero-set-one:*

**assumes**  $a \neq \mathbf{0}$

**assumes**  $a \in$  *carrier*  $R$

**shows** *zero-set*  $R$   $n$   $\mathbf{1}_{R[\mathcal{X}_n]}$  =  $\{\}$

*<proof>*

**lemma** *empty-set-as-affine-alg-set:*

*affine-alg-set*  $R$   $n$   $\{\mathbf{1}_{R[\mathcal{X}_n]}\}$  =  $\{\}$

*<proof>*

**lemma** *empty-is-alg:*

*is-algebraic*  $R$   $n$   $\{\}$

*<proof>*

### 5.5.4 Finite Sets Are Algebraic

the function mapping a point in  $R^n$  to the unique linear polynomial vanishing exclusively at that point

**definition** *pvar-trans* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow 'a \Rightarrow ('a, \text{nat}) \text{mvar-poly}$  **where**  
*pvar-trans*  $n\ i\ a = (\text{pvar } R\ i) \ominus_{R[\mathcal{X}_n]} \text{coord-const } a$

**lemma** *pvar-trans-closed*:

**assumes**  $a \in \text{carrier } R$

**assumes**  $i < n$

**shows**  $\text{pvar-trans } n\ i\ a \in \text{carrier } (R[\mathcal{X}_n])$

*<proof>*

**lemma** *pvar-trans-eval*:

**assumes**  $a \in \text{carrier } R$

**assumes**  $b \in \text{carrier } (R^n)$

**assumes**  $i < n$

**shows**  $\text{eval-at-point } R\ b\ (\text{pvar-trans } n\ i\ a) = (b!i) \ominus a$

*<proof>*

**definition** *point-to-polys* ::  $'a \text{ list} \Rightarrow ('a, \text{nat}) \text{mvar-poly list}$  **where**

*point-to-polys*  $as = \text{map } (\lambda x. \text{pvar-trans } (\text{length } as) (\text{snd } x) (\text{fst } x)) (\text{zip } as\ (\text{index-list } (\text{length } as)))$

**lemma** *point-to-polys-length*:

$\text{length } (\text{point-to-polys } as) = \text{length } as$

*<proof>*

**lemma** *point-to-polysE*:

**assumes**  $i < \text{length } as$

**shows**  $(\text{point-to-polys } as) ! i = (\text{pvar-trans } (\text{length } as) i (as ! i))$

*<proof>*

**lemma** *point-to-polysE'*:

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $i < n$

**shows**  $\text{eval-at-point } R\ as\ ((\text{point-to-polys } as) ! i) = \mathbf{0}$

*<proof>*

**lemma** *point-to-polysE''*:

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $b \in \text{set } (\text{point-to-polys } as)$

**shows**  $\text{eval-at-point } R\ as\ b = \mathbf{0}$

*<proof>*

**lemma** *point-to-polys-zero-set*:

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $b \in \text{set } (\text{point-to-polys } as)$

**shows**  $as \in \text{zero-set } R\ n\ b$

*<proof>*

**lemma** *point-to-polys-closed*:

**assumes**  $as \in \text{carrier } (R^n)$

**shows**  $\text{set } (\text{point-to-polys } as) \subseteq \text{carrier } (R[\mathcal{X}_n])$

*<proof>*

**lemma** *point-to-polys-affine-alg-set*:

**assumes**  $as \in \text{carrier } (R^n)$

**shows**  $\text{affine-alg-set } R \ n \ (\text{set } (\text{point-to-polys } as)) = \{as\}$

*<proof>*

**lemma** *singleton-is-algebraic*:

**assumes**  $as \in \text{carrier } (R^n)$

**shows**  $\text{is-algebraic } R \ n \ \{as\}$

*<proof>*

**lemma**(in *domain-coord-rings*) *finite-sets-are-algebraic*:

**assumes** *finite*  $F$

**shows**  $F \subseteq \text{carrier } (R^n) \longrightarrow \text{is-algebraic } R \ n \ F$

*<proof>*

## 5.6 Polynomial Maps

## 5.7 The Action of Index Permutations on Polynomials

**definition** *permute-poly-args* ::

$\text{nat} \Rightarrow (\text{nat} \Rightarrow \text{nat}) \Rightarrow ('a, \text{nat}) \text{ mvar-poly} \Rightarrow ('a, \text{nat}) \text{ mvar-poly}$  **where**  
 $\text{permute-poly-args } (n::\text{nat}) \ \sigma \ p = \text{indexed-poly-induced-morphism } \{..\lt n\} \ (R[\mathcal{X}_n])$   
 $\text{coord-const } (\lambda i. \text{pvar } R \ (\sigma \ i)) \ p$

**lemma** *permute-poly-args-characterization*:

**assumes**  $\sigma \text{ permutes } \{..\lt n\}$

**shows**  $(\text{ring-hom-ring } (R[\mathcal{X}_n]) \ (R[\mathcal{X}_n]) \ (\text{permute-poly-args } (n::\text{nat}) \ \sigma))$

$(\forall i \in \{..\lt n\}. (\text{permute-poly-args } (n::\text{nat}) \ \sigma) (\text{pvar } R \ i) = \text{pvar } R \ (\sigma \ i))$

$(\forall a \in \text{carrier } R. \text{permute-poly-args } (n::\text{nat}) \ \sigma \ (\text{coord-const } a) = (\text{coord-const } a))$

*<proof>*

**lemma** *permute-poly-args-closed*:

**assumes**  $\sigma \text{ permutes } \{..\lt n\}$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{permute-poly-args } n \ \sigma \ p \in \text{carrier } (R[\mathcal{X}_n])$

*<proof>*

**lemma** *permute-poly-args-constant*:

**assumes**  $a \in \text{carrier } R$

**assumes**  $\sigma \text{ permutes } \{..\lt n\}$

**shows**  $\text{permute-poly-args } n \ \sigma \ (\text{coord-const } a) = (\text{coord-const } a)$

*<proof>*

**lemma** *permute-poly-args-add:*

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $q \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{permute-poly-args } n \ \sigma \ (p \oplus_{R[\mathcal{X}_n]} q) = (\text{permute-poly-args } n \ \sigma \ p) \oplus_{R[\mathcal{X}_n]} (\text{permute-poly-args } n \ \sigma \ q)$

*<proof>*

**lemma** *permute-poly-args-mult:*

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $q \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{permute-poly-args } n \ \sigma \ (p \otimes_{R[\mathcal{X}_n]} q) = (\text{permute-poly-args } n \ \sigma \ p) \otimes_{R[\mathcal{X}_n]} (\text{permute-poly-args } n \ \sigma \ q)$

*<proof>*

**lemma** *permute-poly-args-indexed-pmult:*

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $i \in \{..<n\}$

**shows**  $(\text{permute-poly-args } n \ \sigma \ (p \otimes i)) = (\text{permute-poly-args } n \ \sigma \ p) \otimes (\sigma \ i)$

*<proof>*

**lemma** *permute-list-closed:*

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**shows**  $(\text{permute-list } \sigma \ a) \in \text{carrier } (R^n)$

*<proof>*

**lemma** *permute-list-set:*

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**shows**  $\text{set } (\text{permute-list } \sigma \ a) = \text{set } a$

*<proof>*

**end**

**definition** *perm-map* :: ('a, 'b) ring-scheme  $\Rightarrow$  nat  $\Rightarrow$  (nat  $\Rightarrow$  nat)  $\Rightarrow$  'a list  $\Rightarrow$  'a list **where**

*perm-map*  $R \ n \ \sigma = \text{restrict } (\text{permute-list } \sigma) (\text{carrier } (R^n))$

**context** *cring-coord-rings*

**begin**

**lemma** *perm-map-is-struct-map:*

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**shows**  $\text{perm-map } R \ n \ \sigma \in \text{struct-maps } (R^n) (R^n)$

*<proof>*

**lemma** *permute-poly-args-eval*:

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $\sigma \text{ permutes } \{..<n\}$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{eval-at-point } R \ a \ (\text{permute-poly-args } n \ \sigma \ p) = \text{eval-at-point } R \ (\text{permute-list } \sigma \ a) \ p$

*<proof>*

## 5.8 Inverse Images of Sets by Tuples of Polynomials

**definition** *is-poly-tuple* ::  $\text{nat} \Rightarrow ('a, \text{nat}) \text{ mvar-poly list} \Rightarrow \text{bool}$  **where**  
*is-poly-tuple* ( $n::\text{nat}$ )  $fs = (\text{set } (fs) \subseteq \text{carrier } (R[\mathcal{X}_n]))$

**lemma** *is-poly-tupleE*:

**assumes** *is-poly-tuple*  $n \ fs$

**assumes**  $j < \text{length } fs$

**shows**  $fs ! j \in \text{carrier } (R[\mathcal{X}_n])$

*<proof>*

**lemma** *is-poly-tuple-Cons*:

**assumes** *is-poly-tuple*  $n \ fs$

**assumes**  $f \in \text{carrier } (R[\mathcal{X}_n])$

**shows** *is-poly-tuple*  $n \ (f\#fs)$

*<proof>*

**lemma** *is-poly-tuple-append*:

**assumes** *is-poly-tuple*  $n \ fs$

**assumes**  $f \in \text{carrier } (R[\mathcal{X}_n])$

**shows** *is-poly-tuple*  $n \ (fs@[f])$

*<proof>*

**definition** *poly-tuple-eval* ::  $('a, \text{nat}) \text{ mvar-poly list} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list}$  **where**  
*poly-tuple-eval*  $fs \ as = \text{map } (\lambda f. \text{eval-at-poly } R \ f \ as) \ fs$

**lemma** *poly-tuple-evalE*:

**assumes** *is-poly-tuple*  $n \ fs$

**assumes**  $\text{length } fs = m$

**assumes**  $as \in \text{carrier } (R^n)$

**assumes**  $j < m$

**shows**  $(\text{poly-tuple-eval } fs \ as) ! j \in \text{carrier } R$

*<proof>*

**lemma** *poly-tuple-evalE'*:

**shows**  $\text{length } (\text{poly-tuple-eval } fs \ as) = \text{length } fs$

*<proof>*

**lemma** *poly-tuple-evalE''*:

**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes** *length*  $fs = m$   
**assumes**  $as \in \text{carrier } (R^n)$   
**assumes**  $j < m$   
**shows**  $(\text{poly-tuple-eval } fs \ as)!\j = (\text{eval-at-poly } R \ (fs!\j) \ as)$   
 $\langle \text{proof} \rangle$

**lemma** *poly-tuple-eval-closed*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes** *length*  $fs = m$   
**assumes**  $as \in \text{carrier } (R^n)$   
**shows**  $(\text{poly-tuple-eval } fs \ as) \in \text{carrier } (R^m)$   
 $\langle \text{proof} \rangle$

**lemma** *poly-tuple-eval-Cons*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes** *length*  $fs = m$   
**assumes**  $as \in \text{carrier } (R^n)$   
**assumes**  $f \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $(\text{poly-tuple-eval } (f\#fs) \ as) = (\text{eval-at-point } R \ as \ f)\#(\text{poly-tuple-eval } fs \ as)$   
 $\langle \text{proof} \rangle$

**definition** *poly-tuple-pullback* ::  
 $\text{nat} \Rightarrow 'a \text{ list set} \Rightarrow ('a, \text{nat}) \text{ mvar-poly list} \Rightarrow 'a \text{ list set}$  **where**  
 $\text{poly-tuple-pullback } n \ S \ fs = ((\text{poly-tuple-eval } fs) - ' S) \cap (\text{carrier } (R^n))$

**lemma** *poly-pullbackE*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes** *length*  $fs = m$   
**assumes**  $S \subseteq \text{carrier } (R^m)$   
**shows**  $\text{poly-tuple-pullback } n \ S \ fs \subseteq \text{carrier } (R^n)$   
 $\langle \text{proof} \rangle$

**lemma** *poly-pullbackE'*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes** *length*  $fs = m$   
**assumes**  $S \subseteq \text{carrier } (R^m)$   
**assumes**  $as \in \text{poly-tuple-pullback } n \ S \ fs$   
**shows**  $as \in \text{carrier } (R^n)$   
 $\text{poly-tuple-eval } fs \ as \in S$   
 $\langle \text{proof} \rangle$

**lemma** *poly-pullbackI*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes** *length*  $fs = m$   
**assumes**  $S \subseteq \text{carrier } (R^m)$   
**assumes**  $as \in \text{carrier } (R^n)$   
**assumes**  $\text{poly-tuple-eval } fs \ as \in S$   
**shows**  $as \in \text{poly-tuple-pullback } n \ S \ fs$

*<proof>*

**end**

coordinate permutations as pullbacks. The point here is to realize that permutations of indices are just pullbacks (or pushforwards) by particular polynomial maps

**abbreviation** *pvar-list* **where**

*pvar-list*  $R$   $n \equiv \text{map } (\text{pvar } R) (\text{index-list } n)$

**lemma** *pvar-list-elements*:

**assumes**  $i < n$

**shows**  $\text{pvar-list } R$   $n$  !  $i = \text{pvar } R$   $i$

*<proof>*

**lemma** *pvar-list-length*:

$\text{length } (\text{pvar-list } R$   $n) = n$

*<proof>*

**context** *cring-coord-rings*

**begin**

**lemma** *pvar-list-is-poly-tuple*:

**shows** *is-poly-tuple*  $n$  ( $\text{pvar-list } R$   $n$ )

*<proof>*

**lemma** *permutation-of-poly-list-is-poly-list*:

**assumes** *is-poly-tuple*  $k$   $fs$

**assumes**  $\sigma$  *permutes*  $\{.. < \text{length } fs\}$

**shows** *is-poly-tuple*  $k$  ( $\text{permute-list } \sigma$   $fs$ )

*<proof>*

**lemma** *permutation-of-poly-listE*:

**assumes** *is-poly-tuple*  $k$   $fs$

**assumes**  $\sigma$  *permutes*  $\{.. < \text{length } fs\}$

**assumes**  $i < \text{length } fs$

**shows**  $(\text{permute-list } \sigma$   $fs)$  !  $i = fs$  !  $(\sigma$   $i)$

*<proof>*

**lemma** *pushforward-by-permutation-of-poly-list*:

**assumes** *is-poly-tuple*  $k$   $fs$

**assumes**  $\sigma$  *permutes*  $\{.. < \text{length } fs\}$

**assumes**  $as \in \text{carrier } (R^k)$

**shows**  $\text{poly-tuple-eval } (\text{permute-list } \sigma$   $fs)$   $as = \text{permute-list } \sigma$  ( $\text{poly-tuple-eval } fs$   $as$ )

*<proof>*

**lemma** *pushforward-by-pvar-list*:  
**assumes**  $as \in \text{carrier } (R^n)$   
**shows**  $\text{poly-tuple-eval } (\text{pvar-list } R \ n) \ as = as$   
 $\langle \text{proof} \rangle$

**lemma** *pushforward-by-permuted-pvar-list*:  
**assumes**  $\sigma \text{ permutes } \{..< n\}$   
**assumes**  $as \in \text{carrier } (R^n)$   
**shows**  $\text{poly-tuple-eval } (\text{permute-list } \sigma \ (\text{pvar-list } R \ n)) \ as = \text{permute-list } \sigma \ as$   
 $\langle \text{proof} \rangle$

**lemma** *pullback-by-permutation-of-poly-list*:  
**assumes**  $\sigma \text{ permutes } \{..< n\}$   
**assumes**  $S \subseteq \text{carrier } (R^n)$   
**shows**  $\text{poly-tuple-pullback } n \ S \ (\text{permute-list } \sigma \ (\text{pvar-list } R \ n)) =$   
 $\text{permute-list } (\text{fun-inv } \sigma) \ 'S$   
 $\langle \text{proof} \rangle$

**lemma** *pullback-by-permutation-of-poly-list'*:  
**assumes**  $\sigma \text{ permutes } \{..< n\}$   
**assumes**  $S \subseteq \text{carrier } (R^n)$   
**shows**  $\text{poly-tuple-pullback } n \ S \ (\text{permute-list } (\text{fun-inv } \sigma) \ (\text{pvar-list } R \ n)) =$   
 $\text{permute-list } \sigma \ 'S$   
 $\langle \text{proof} \rangle$

## 5.9 Composing Polynomial Tuples With Polynomials

composition of a multivariable polynomial by a list of polynomials

**definition** *poly-compose* ::  
 $\text{nat} \Rightarrow \text{nat} \Rightarrow ('a, \text{nat}) \text{ mvar-poly list} \Rightarrow ('a, \text{nat}) \text{ mvar-poly} \Rightarrow ('a, \text{nat}) \text{ mvar-poly}$   
**where**  
 $\text{poly-compose } n \ m \ fs = \text{indexed-poly-induced-morphism } \{..<n\} \ (\text{coord-ring } R \ m) \ (\lambda$   
 $s. R.\text{indexed-const } s) \ (\lambda i. fs!i)$

**lemma** *poly-compose-var*:  
**assumes**  $\text{is-poly-tuple } m \ fs$   
**assumes**  $\text{length } fs = n$   
**assumes**  $j < n$   
**shows**  $\text{poly-compose } n \ m \ fs \ (\text{pvar } R \ j) = (fs!j)$   
 $\langle \text{proof} \rangle$

**lemma** *Pring-universal-prop-assms*:  
**assumes**  $\text{is-poly-tuple } m \ fs$   
**assumes**  $\text{length } fs = n$   
**shows**  $(\lambda i. fs!i) \in \{..<n\} \rightarrow \text{carrier } (\text{coord-ring } R \ m)$   
 $\text{ring-hom-ring } R \ (\text{coord-ring } R \ m) \ \text{coord-const}$   
 $\langle \text{proof} \rangle$

**lemma** *poly-compose-ring-hom*:

**assumes** *is-poly-tuple*  $m$   $fs$   
**assumes**  $length\ fs = n$   
**shows**  $(ring-hom-ring\ (R[\mathcal{X}_n])\ (coord-ring\ R\ m)\ (poly-compose\ n\ m\ fs))$   
 $\langle proof \rangle$

**lemma** *poly-compose-closed*:  
**assumes** *is-poly-tuple*  $m$   $fs$   
**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (R[\mathcal{X}_n])$   
**shows**  $(poly-compose\ n\ m\ fs\ f) \in carrier\ (coord-ring\ R\ m)$   
 $\langle proof \rangle$

**lemma** *poly-compose-add*:  
**assumes** *is-poly-tuple*  $m$   $fs$   
**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (R[\mathcal{X}_n])$   
**assumes**  $g \in carrier\ (R[\mathcal{X}_n])$   
**shows**  $poly-compose\ n\ m\ fs\ (f \oplus_{R[\mathcal{X}_n]}\ g) = (poly-compose\ n\ m\ fs\ f) \oplus_{coord-ring\ R\ m}$   
 $(poly-compose\ n\ m\ fs\ g)$   
 $\langle proof \rangle$

**lemma** *poly-compose-mult*:  
**assumes** *is-poly-tuple*  $m$   $fs$   
**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (R[\mathcal{X}_n])$   
**assumes**  $g \in carrier\ (R[\mathcal{X}_n])$   
**shows**  $poly-compose\ n\ m\ fs\ (f \otimes_{R[\mathcal{X}_n]}\ g) = (poly-compose\ n\ m\ fs\ f) \otimes_{coord-ring\ R\ m}$   
 $(poly-compose\ n\ m\ fs\ g)$   
 $\langle proof \rangle$

**lemma** *poly-compose-indexed-pmult*:  
**assumes** *is-poly-tuple*  $m$   $fs$   
**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (R[\mathcal{X}_n])$   
**assumes**  $i < n$   
**shows**  $poly-compose\ n\ m\ fs\ (f \otimes\ i) = (poly-compose\ n\ m\ fs\ f) \otimes_{coord-ring\ R\ m}$   
 $(fs!i)$   
 $\langle proof \rangle$

**lemma** *poly-compose-const*:  
**assumes** *is-poly-tuple*  $m$   $fs$   
**assumes**  $length\ fs = n$   
**assumes**  $a \in carrier\ R$   
**shows**  $poly-compose\ n\ m\ fs\ (coord-const\ a) = coord-const\ a$   
 $\langle proof \rangle$

evaluating polynomial compositions

**lemma** *poly-compose-eval*:  
**assumes** *is-poly-tuple*  $m$   $fs$

**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (R[\mathcal{X}_n])$   
**assumes**  $as \in carrier\ (R^m)$   
**shows**  $eval-at-point\ R\ (poly-tuple-eval\ fs\ as)\ f = eval-at-point\ R\ as\ (poly-compose\ n\ m\ fs\ f)$   
 $\langle proof \rangle$

## 5.10 Extensional Polynomial Maps

Polynomial Maps between powers of a ring

**definition**  $poly-map :: nat \Rightarrow ('a, nat)\ mvar-poly\ list \Rightarrow 'a\ list \Rightarrow 'a\ list$  **where**  
 $poly-map\ n\ fs = (\lambda a \in carrier\ (R^n). poly-tuple-eval\ fs\ a)$

**lemma**  $poly-map-is-struct-map$ :  
**assumes**  $is-poly-tuple\ n\ fs$   
**assumes**  $length\ fs = m$   
**shows**  $poly-map\ n\ fs \in struct-maps\ (R^n)\ (R^m)$   
 $\langle proof \rangle$

**lemma**  $poly-map-closed$ :  
**assumes**  $is-poly-tuple\ n\ fs$   
**assumes**  $length\ fs = m$   
**assumes**  $as \in carrier\ (R^n)$   
**shows**  $poly-map\ n\ fs\ as \in carrier\ (R^m)$   
 $\langle proof \rangle$

**definition**  $poly-maps :: nat \Rightarrow nat \Rightarrow ('a\ list \Rightarrow 'a\ list)\ set$  **where**  
 $poly-maps\ n\ m = \{F. (\exists\ fs. length\ fs = m \wedge is-poly-tuple\ n\ fs \wedge F = poly-map\ n\ fs)\}$

**lemma**  $poly-maps-memE$ :  
**assumes**  $F \in poly-maps\ n\ m$   
**obtains**  $fs$  **where**  $length\ fs = m \wedge is-poly-tuple\ n\ fs \wedge F = poly-map\ n\ fs$   
 $\langle proof \rangle$

**lemma**  $poly-maps-memI$ :  
**assumes**  $is-poly-tuple\ n\ fs$   
**assumes**  $length\ fs = m$   
**assumes**  $F = poly-map\ n\ fs$   
**shows**  $F \in poly-maps\ n\ m$   
 $\langle proof \rangle$

**lemma**  $poly-map-compose-closed$ :  
**assumes**  $is-poly-tuple\ n\ fs$   
**assumes**  $length\ fs = m$   
**assumes**  $is-poly-tuple\ k\ gs$   
**assumes**  $length\ gs = n$   
**shows**  $is-poly-tuple\ k\ (map\ (poly-compose\ n\ k\ gs)\ fs)$   
 $\langle proof \rangle$

**lemma** *poly-map-compose-closed'*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes**  $length\ fs = m$   
**assumes** *is-poly-tuple*  $k$   $gs$   
**assumes**  $length\ gs = n$   
**shows**  $poly\text{-}map\ k\ (map\ (poly\text{-}compose\ n\ k\ gs)\ fs) \in poly\text{-}maps\ k\ m$   
 $\langle proof \rangle$

**lemma** *poly-map-pullback-char*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes**  $length\ fs = m$   
**assumes** *is-poly-tuple*  $k$   $gs$   
**assumes**  $length\ gs = n$   
**shows**  $(pullback\ (R^k)\ (poly\text{-}map\ k\ gs)\ (poly\text{-}map\ n\ fs)) =$   
 $poly\text{-}map\ k\ (map\ (poly\text{-}compose\ n\ k\ gs)\ fs)$   
 $\langle proof \rangle$

**lemma** *poly-map-pullback-closed*:  
**assumes**  $F \in poly\text{-}maps\ n\ m$   
**assumes**  $G \in poly\text{-}maps\ k\ n$   
**shows**  $(pullback\ (R^k)\ G\ F) \in poly\text{-}maps\ k\ m$   
 $\langle proof \rangle$

**lemma** *poly-map-cons*:  
**assumes**  $a \in carrier\ (R^n)$   
**shows**  $poly\text{-}map\ n\ (f\#\#fs)\ a = (eval\text{-}at\text{-}point\ R\ a\ f)\#\#poly\text{-}map\ n\ fs\ a$   
 $\langle proof \rangle$

**lemma** *poly-map-append*:  
**assumes**  $a \in carrier\ (R^n)$   
**shows**  $poly\text{-}map\ n\ (fs\@\@gs)\ a = (poly\text{-}map\ n\ fs\ a)\@\@(poly\text{-}map\ n\ gs\ a)$   
 $\langle proof \rangle$

## 6 Nesting of Polynomial Rings

**lemma** *poly-ring-car-mono*:  
**assumes**  $n \leq m$   
**shows**  $carrier\ (R[\mathcal{X}_n]) \subseteq carrier\ (coord\text{-}ring\ R\ m)$   
 $\langle proof \rangle$

**lemma** *poly-ring-car-mono'[simp]*:  
**shows**  $carrier\ (R[\mathcal{X}_n]) \subseteq carrier\ (R[\mathcal{X}_{Suc\ n}])$   
 $carrier\ (R[\mathcal{X}_n]) \subseteq carrier\ (R[\mathcal{X}_{n+m}])$   
 $\langle proof \rangle$

**lemma** *poly-ring-add-mono*:  
**assumes**  $n \leq m$   
**assumes**  $A \in carrier\ (R[\mathcal{X}_n])$

**assumes**  $B \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $A \oplus_{R[\mathcal{X}_n]} B = A \oplus_{\text{coord-ring } R \ m} B$   
 $\langle \text{proof} \rangle$

**lemma** *poly-ring-add-mono'*:  
**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $B \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $A \oplus_{R[\mathcal{X}_n]} B = A \oplus_{R[\mathcal{X}_{\text{Suc } n}]} B$   
 $A \oplus_{R[\mathcal{X}_n]} B = A \oplus_{R[\mathcal{X}_{n+m}]} B$   
 $\langle \text{proof} \rangle$

**lemma** *poly-ring-times-mono*:  
**assumes**  $n \leq m$   
**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $B \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $A \otimes_{R[\mathcal{X}_n]} B = A \otimes_{\text{coord-ring } R \ m} B$   
 $\langle \text{proof} \rangle$

**lemma** *poly-ring-times-mono'*:  
**assumes**  $A \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $B \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $A \otimes_{R[\mathcal{X}_n]} B = A \otimes_{R[\mathcal{X}_{\text{Suc } n}]} B$   
 $A \otimes_{R[\mathcal{X}_n]} B = A \otimes_{R[\mathcal{X}_{n+m}]} B$   
 $\langle \text{proof} \rangle$

**lemma** *poly-ring-one-mono*:  
**assumes**  $n \leq m$   
**shows**  $\mathbf{1}_{R[\mathcal{X}_n]} = \mathbf{1}_{\text{coord-ring } R \ m}$   
 $\langle \text{proof} \rangle$

**lemma** *poly-ring-zero-mono*:  
**assumes**  $n \leq m$   
**shows**  $\mathbf{0}_{R[\mathcal{X}_n]} = \mathbf{0}_{\text{coord-ring } R \ m}$   
 $\langle \text{proof} \rangle$

replacing the variables in a polynomial with new variables

**definition** *shift-vars* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow ('a, \text{nat}) \text{ mvar-poly} \Rightarrow ('a, \text{nat}) \text{ mvar-poly}$   
**where**  
*shift-vars*  $n \ m \ p = \text{indexed-poly-induced-morphism } \{..<n\} (R[\mathcal{X}_{n+m}]) \ \text{coord-const}$   
 $(\lambda i. \text{pvar } R \ (i + m)) \ p$

**lemma** *shift-vars-char*:  
**shows**  $(\text{ring-hom-ring } (R[\mathcal{X}_n]) \ (R[\mathcal{X}_{n+m}]) \ (\text{shift-vars } n \ m))$   
 $(\forall i \in \{..<n\}. (\text{shift-vars } n \ m) (\text{pvar } R \ i) = \text{pvar } R \ (i + m))$   
 $(\forall a \in \text{carrier } R. (\text{shift-vars } n \ m) (R.\text{indexed-const } a) = (\text{coord-const } a))$   
 $\langle \text{proof} \rangle$

**lemma** *shift-vars-constant*:

**assumes**  $a \in \text{carrier } R$   
**shows**  $\text{shift-vars } n \ m \ (\text{coord-const } a) = \text{coord-const } a$   
 $\langle \text{proof} \rangle$

**lemma** *shift-vars-pvar*:  
**assumes**  $i \in \{..<n\}$   
**shows**  $\text{shift-vars } n \ m \ (\text{pvar } R \ i) = \text{pvar } R \ (i + m)$   
 $\langle \text{proof} \rangle$

**lemma** *shift-vars-add*:  
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $Q \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $\text{shift-vars } n \ m \ (p \oplus_{R[\mathcal{X}_n]} Q) = \text{shift-vars } n \ m \ p \oplus_{R[\mathcal{X}_{n+m}]} \text{shift-vars } n$   
 $m \ Q$   
 $\langle \text{proof} \rangle$

**lemma** *shift-vars-mult*:  
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $Q \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $\text{shift-vars } n \ m \ (p \otimes_{R[\mathcal{X}_n]} Q) = \text{shift-vars } n \ m \ p \otimes_{R[\mathcal{X}_{n+m}]} \text{shift-vars } n$   
 $m \ Q$   
 $\langle \text{proof} \rangle$

**lemma** *shift-vars-indexed-pmult*:  
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $i \in \{..<n\}$   
**shows**  $\text{shift-vars } n \ m \ (p \otimes i) = (\text{shift-vars } n \ m \ p) \otimes_{R[\mathcal{X}_{n+m}]} (\text{pvar } R \ (i + m))$   
 $\langle \text{proof} \rangle$

**lemma** *shift-vars-closed*:  
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $\text{shift-vars } n \ m \ p \in \text{carrier } (R[\mathcal{X}_{n+m}])$   
 $\langle \text{proof} \rangle$

**lemma** *shift-vars-eval*:  
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $a \in \text{carrier } (R^m)$   
**assumes**  $b \in \text{carrier } (R^n)$   
**shows**  $\text{eval-at-point } R \ (a \ @ \ b) \ (\text{shift-vars } n \ m \ p) = \text{eval-at-point } R \ b \ p$   
 $\langle \text{proof} \rangle$

Evaluating a polynomial from a lower poly ring in a higher power:

**lemma** *poly-eval-cartesian-prod*:  
**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $b \in \text{carrier } (R^m)$   
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $\text{eval-at-point } R \ a \ p = \text{eval-at-point } R \ (a \ @ \ b) \ p$   
 $\langle \text{proof} \rangle$

Evaluating polynomials at points in higher powers:

**lemma** *eval-at-points-higher-pow*:

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$

**assumes**  $k \geq n$

**assumes**  $a \in \text{carrier } (R^k)$

**shows**  $\text{eval-at-point } R \ a \ p = \text{eval-at-point } R \ (\text{take } n \ a) \ p$

*<proof>*

## 6.1 Diagonal sets in even powers of $R$

In this section, by a diagonal set in  $R^{(2m)}$  we will mean the set of points  $(x, x)$ , where  $x \in R^m$ . This is slightly different from the standard definition. Introducing these sets will be useful for reasoning about multiplicative inverses of functions later on.

**definition** *diagonal* ::  $\text{nat} \Rightarrow 'a \text{ list set}$  **where**

$\text{diagonal } m = \{x \in \text{carrier } (R^{m+m}), \text{take } m \ x = \text{drop } m \ x\}$

**lemma** *diagonalE*:

**assumes**  $x \in \text{diagonal } m$

**shows**  $x = (\text{take } m \ x) @ (\text{take } m \ x)$

$x \in \text{carrier } (R^{m+m})$

$\text{take } m \ x \in \text{carrier } (R^m)$

$\bigwedge i. i < m \implies x!i = x!(i + m)$

*<proof>*

**lemma** *diagonalI*:

**assumes**  $x = (\text{take } m \ x) @ (\text{take } m \ x)$

**assumes**  $\text{take } m \ x \in \text{carrier } (R^m)$

**shows**  $x \in \text{diagonal } m$

*<proof>*

**definition** *diag-def-poly* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow ('a, \text{nat}) \text{ mvar-poly}$  **where**

$\text{diag-def-poly } n \ i = \text{pvar } R \ i \ominus_{\text{coord-ring } R \ (n + n)} \text{pvar } R \ (i + n)$

**lemma** *diag-def-poly-closed*:

**assumes**  $i < n$

**shows**  $\text{diag-def-poly } n \ i \in \text{carrier } (\text{coord-ring } R \ (n + n))$

*<proof>*

**lemma** *diag-def-poly-eval*:

**assumes**  $i < n$

**assumes**  $x \in \text{carrier } (R^{n+n})$

**shows**  $\text{eval-at-point } R \ x \ (\text{diag-def-poly } n \ i) = (x!i) \ominus (x!(i + n))$

*<proof>*

**definition** *diag-def-poly-set* ::  $\text{nat} \Rightarrow ('a, \text{nat}) \text{ mvar-poly set}$  **where**

$\text{diag-def-poly-set } n = \text{diag-def-poly } n \ \{..<n\}$

**lemma** *diag-def-poly-set-in-coord-ring*:  
**shows** *diag-def-poly-set*  $n \subseteq \text{carrier } (\text{coord-ring } R (n + n))$   
 $\langle \text{proof} \rangle$

**lemma** *diag-def-poly-set-finite*:  
*finite* (*diag-def-poly-set*  $n$ )  
 $\langle \text{proof} \rangle$

**lemma** *diag-def-poly-eval-at-diagonal*:  
**assumes**  $x \in \text{diagonal } n$   
**assumes**  $i < n$   
**shows** *eval-at-point*  $R x (\text{diag-def-poly } n i) = \mathbf{0}$   
 $\langle \text{proof} \rangle$

**lemma** *diagonal-as-affine-alg-set*:  
**shows** *diagonal*  $n = \text{affine-alg-set } R (n + n) (\text{diag-def-poly-set } n)$   
 $\langle \text{proof} \rangle$

**lemma** *diagonal-is-algebraic*:  
**shows** *is-algebraic*  $R (n + n) (\text{diagonal } n)$   
 $\langle \text{proof} \rangle$

**end**

## 6.2 Tuples of Functions

**definition** *is-function-tuple* ::  $('a, 'b)$  *ring-scheme*  $\Rightarrow \text{nat} \Rightarrow ('a \text{ list} \Rightarrow 'a) \text{ list} \Rightarrow \text{bool}$  **where**  
*is-function-tuple*  $R n fs = (\text{set } fs \subseteq \text{carrier } (R^n) \rightarrow \text{carrier } R)$

**lemma** *is-function-tupleI*:  
**assumes**  $(\text{set } fs \subseteq \text{carrier } (R^n) \rightarrow \text{carrier } R)$   
**shows** *is-function-tuple*  $R n fs$   
 $\langle \text{proof} \rangle$

**lemma** *is-function-tuple-append*:  
**assumes** *is-function-tuple*  $R n fs$   
**assumes** *is-function-tuple*  $R n gs$   
**shows** *is-function-tuple*  $R n (fs@gs)$   
 $\langle \text{proof} \rangle$

**lemma** *is-function-tuple-Cons*:  
**assumes** *is-function-tuple*  $R n fs$   
**assumes**  $f \in \text{carrier } (R^n) \rightarrow \text{carrier } R$   
**shows** *is-function-tuple*  $R n (f\#fs)$   
 $\langle \text{proof} \rangle$

**lemma** *is-function-tuple-snoc*:

**assumes** *is-function-tuple*  $R\ n\ fs$   
**assumes**  $f \in \text{carrier } (R^n) \rightarrow \text{carrier } R$   
**shows** *is-function-tuple*  $R\ n\ (fs@[f])$   
 <proof>

**lemma** *is-function-tuple-list-update*:  
**assumes** *is-function-tuple*  $R\ n\ fs$   
**assumes**  $f \in \text{carrier } (R^n) \rightarrow \text{carrier } R$   
**assumes**  $i < n$   
**shows** *is-function-tuple*  $R\ n\ (fs[i := f])$   
 <proof>

**definition** *function-tuple-eval* ::  $'b \Rightarrow 'c \Rightarrow ('d \Rightarrow 'a)\ \text{list} \Rightarrow 'd \Rightarrow 'a\ \text{list}$  **where**  
*function-tuple-eval*  $R\ n\ fs\ x = \text{map } (\lambda f. f\ x)\ fs$

**lemma** *function-tuple-eval-closed*:  
**assumes** *is-function-tuple*  $R\ n\ fs$   
**assumes**  $x \in \text{carrier } (R^n)$   
**shows** *function-tuple-eval*  $R\ n\ fs\ x \in \text{carrier } (R^{\text{length } fs})$   
 <proof>

**definition** *coord-fun* ::  
 $('a, 'c)\ \text{ring-scheme} \Rightarrow \text{nat} \Rightarrow ('a\ \text{list} \Rightarrow 'b\ \text{list}) \Rightarrow \text{nat} \Rightarrow 'a\ \text{list} \Rightarrow 'b$  **where**  
*coord-fun*  $R\ n\ g\ i = (\lambda x \in \text{carrier } (R^n). (g\ x)\ !\ i)$

**lemma**(in *cring*) *map-is-coord-fun-tuple*:  
**assumes**  $g \in \text{carrier } (R^n) \rightarrow_E \text{carrier } (R^m)$   
**shows**  $g = (\lambda x \in \text{carrier } (R^n). \text{function-tuple-eval } R\ n\ (\text{map } (\text{coord-fun } R\ n\ g)\ [0..<m])\ x)$   
 <proof>

**definition** *function-tuple-comp* ::  
 $'c \Rightarrow ('a \Rightarrow 'd)\ \text{list} \Rightarrow ('d\ \text{list} \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$  **where**  
*function-tuple-comp*  $R\ fs\ f = f \circ (\text{function-tuple-eval } R\ (0::\text{nat})\ fs)$

**lemma** *function-tuple-comp-closed*:  
**assumes**  $f \in \text{carrier } (R^n) \rightarrow \text{carrier } R$   
**assumes**  $\text{length } fs = n$   
**assumes** *is-function-tuple*  $R\ m\ fs$   
**shows** *function-tuple-comp*  $R\ fs\ f \in \text{carrier } (R^m) \rightarrow \text{carrier } R$   
 <proof>

**fun** *id-function-tuple* **where**  
*id-function-tuple*  $(R::('a, 'b)\ \text{partial-object-scheme})\ 0 = []$   
*id-function-tuple*  $R\ (\text{Suc } n) = \text{id-function-tuple } R\ n\ @\ [(\lambda(x::'a\ \text{list}). x\ !\ n)]$

**lemma** *id-function-tuple-is-function-tuple*:  
 $\bigwedge k. k \geq n \implies \text{is-function-tuple } R\ k\ (\text{id-function-tuple } R\ n)$   
 <proof>

**lemma** *id-function-tuple-is-function-tuple'*:  
*is-function-tuple*  $R$   $n$  (*id-function-tuple*  $R$   $n$ )  
 ⟨*proof*⟩

**lemma** *id-function-tuple-eval-is-take*:  
**assumes**  $a \in \text{carrier } (R^n)$   
**shows**  $k \leq n \implies \text{function-tuple-eval } R$   $n$  (*id-function-tuple*  $R$   $k$ )  $a = \text{take } k$   $a$   
 ⟨*proof*⟩

**lemma** *id-function-tuple-eval-is-id*:  
**assumes**  $a \in \text{carrier } (R^n)$   
**shows** *function-tuple-eval*  $R$   $n$  (*id-function-tuple*  $R$   $n$ )  $a = a$   
 ⟨*proof*⟩

Composing a function tuple with a polynomial

**definition** *poly-function-tuple-comp* ::  
 ( $'a, 'b$ ) *ring-scheme*  $\implies \text{nat} \implies ('a \text{ list} \implies 'a) \text{ list} \implies ('a, \text{nat}) \text{ mvar-poly} \implies 'a \text{ list}$   
 $\implies 'a$  **where**  
*poly-function-tuple-comp*  $R$   $n$   $fs$   $f = \text{eval-at-poly } R$   $f \circ \text{function-tuple-eval } R$   $n$   $fs$

**context** *cring-coord-rings*  
**begin**

**lemma** *poly-function-tuple-comp-closed*:  
**assumes** *is-function-tuple*  $R$   $n$   $fs$   
**assumes**  $f \in \text{carrier } (\text{coord-ring } R$  ( $\text{length } fs$ ))  
**shows** *poly-function-tuple-comp*  $R$   $n$   $fs$   $f \in \text{carrier } (R^n) \rightarrow \text{carrier } R$   
 ⟨*proof*⟩

**lemma** *poly-function-tuple-comp-eq*:  
**assumes** *is-function-tuple*  $R$   $n$   $fs$   
**assumes**  $f \in \text{carrier } (\text{coord-ring } R$  ( $\text{length } fs$ ))  
**assumes**  $a \in \text{carrier } (R^n)$   
**shows** *poly-function-tuple-comp*  $R$   $n$   $fs$   $f$   $a = \text{eval-at-poly } R$   $f$  (*function-tuple-eval*  $R$   $n$   $fs$   $a$ )  
 ⟨*proof*⟩

**lemma** *poly-function-tuple-comp-constant*:  
**assumes** *is-function-tuple*  $R$   $n$   $fs$   
**assumes**  $a \in \text{carrier } R$   
**assumes**  $x \in \text{carrier } (R^n)$   
**shows** *poly-function-tuple-comp*  $R$   $n$   $fs$  (*coord-const*  $a$ )  $x = a$   
 ⟨*proof*⟩

**lemma** *poly-function-tuple-comp-add*:  
**assumes** *is-function-tuple*  $R$   $n$   $fs$   
**assumes**  $k \leq \text{length } fs$   
**assumes**  $p \in \text{carrier } (\text{coord-ring } R$   $k$ )

**assumes**  $Q \in \text{carrier } (\text{coord-ring } R \ k)$   
**assumes**  $x \in \text{carrier } (R^n)$   
**shows**  $\text{poly-function-tuple-comp } R \ n \ \text{fs } (p \oplus_{R[\mathcal{X}_n]} Q) \ x =$   
 $(\text{poly-function-tuple-comp } R \ n \ \text{fs } p \ x) \oplus (\text{poly-function-tuple-comp } R \ n \ \text{fs}$   
 $Q \ x)$   
*<proof>*

**lemma** *poly-function-tuple-comp-mult:*  
**assumes**  $\text{is-function-tuple } R \ n \ \text{fs}$   
**assumes**  $k \leq \text{length } \text{fs}$   
**assumes**  $p \in \text{carrier } (\text{coord-ring } R \ k)$   
**assumes**  $Q \in \text{carrier } (\text{coord-ring } R \ k)$   
**assumes**  $x \in \text{carrier } (R^n)$   
**shows**  $\text{poly-function-tuple-comp } R \ n \ \text{fs } (p \otimes_{R[\mathcal{X}_n]} Q) \ x =$   
 $(\text{poly-function-tuple-comp } R \ n \ \text{fs } p \ x) \otimes (\text{poly-function-tuple-comp } R \ n \ \text{fs}$   
 $Q \ x)$   
*<proof>*

**lemma** *poly-function-tuple-comp-pvar:*  
**assumes**  $\text{is-function-tuple } R \ n \ \text{fs}$   
**assumes**  $k < \text{length } \text{fs}$   
**assumes**  $x \in \text{carrier } (R^n)$   
**shows**  $\text{poly-function-tuple-comp } R \ n \ \text{fs } (\text{pvar } R \ k) \ x = (\text{fs } ! \ k) \ x$   
*<proof>*

**end**

The coordinate ring of polynomials indexed by natural numbers

**definition** *Coord-ring* ::  $('a, 'b) \text{ ring-scheme} \Rightarrow ('a, ('a, \text{nat}) \text{ mvar-poly}) \text{ module}$   
**where**

$\text{Coord-ring } R = \text{Pring } R \ (\text{UNIV} :: \text{nat set})$

Some general closure lemmas for coordinate rings

**context** *cring-coord-rings*

**begin**

**lemma** *coord-ring-monom-term-closed:*  
**assumes**  $a \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $b \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $a \otimes_{(R[\mathcal{X}_n])} b[\overset{\frown}{\frown}]_{(R[\mathcal{X}_n])}(n::\text{nat}) \in \text{carrier } (R[\mathcal{X}_n])$   
*<proof>*

**lemma** *coord-ring-monom-term-plus-closed:*  
**assumes**  $a \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $b \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $c \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $c \oplus_{(R[\mathcal{X}_n])} a \otimes_{(R[\mathcal{X}_n])} b[\overset{\frown}{\frown}]_{(R[\mathcal{X}_n])}(n::\text{nat}) \in \text{carrier } (R[\mathcal{X}_n])$   
*<proof>*

**end**

### 6.3 Generic Univariate Polynomials

By a generic univariate polynomial, we mean a polynomial in one variable whose coefficients are coordinate functions over a ring. That is, a polynomial of the form:

$$f(t) = x_0 + x_1 t + \cdots + x_n t^n$$

Such a polynomial can be construed as an element of  $R[x_0, \dots, x_n](t)$ , or as an element of  $R[x_0, \dots, x_n, x_{n+1}]$ . We will initially define the latter version, and show that it can easily be cast to the former using the function “IP\_to\_UP”. Such a polynomial can be cast to a univariate polynomial over the ring  $R$  by substituting a tuple of ring elements for the coefficients.

**definition** *generic-poly-lt* :: ('a, 'b) ring-scheme  $\Rightarrow$  nat  $\Rightarrow$  ('a, nat) mvar-poly where

*generic-poly-lt* R n = (pvar R (Suc n))  $\otimes_{\text{coord-ring } R \text{ (Suc (Suc n))}}$  (pvar R 0)  $[\overset{\wedge}{\text{coord-ring } R \text{ (Suc (Suc n))}}$  n

**fun** *generic-poly where*

*generic-poly* R (0::nat) = pvar R 1

*generic-poly* R (Suc n) = (*generic-poly* R n)  $\oplus_{(\text{coord-ring } R \text{ (n+3)})}$  *generic-poly-lt* R (Suc n)

**context** *cring-coord-rings*

**begin**

**lemma** *generic-poly-lt-closed*:

*generic-poly-lt* R n  $\in$  carrier (coord-ring R (Suc (Suc n)))

$\langle$ proof $\rangle$

**lemma** *generic-poly-lt-eval*:

**assumes** a  $\in$  carrier ( $R^{n+2}$ )

**shows** eval-at-point R a (*generic-poly-lt* R n) = a!(Suc n)  $\otimes$  (a!0)  $[\overset{\wedge}{n}$

$\langle$ proof $\rangle$

**lemma** *generic-poly-closed*:

*generic-poly* R n  $\in$  carrier (coord-ring R (Suc (Suc n)))

$\langle$ proof $\rangle$

**lemma** *generic-poly-closed'*:

**assumes** k  $\leq$  n

**shows** *generic-poly* R k  $\in$  carrier (coord-ring R (Suc (Suc n)))

$\langle$ proof $\rangle$

**lemma** *generic-poly-eval-at-point*:

**assumes** a  $\in$  carrier ( $R^{n+3}$ )

**shows** eval-at-point R a (*generic-poly* R (Suc n)) = (eval-at-point R a (*generic-poly* R n))  $\oplus$

$$(a!(n + 2)) \otimes (a!0)[\ulcorner](\text{Suc } n)$$

*<proof>*

**end**

We can turn points in  $R^{n+1}$  into univariate polynomials with the associated coefficients via partial evaluation of the generic polynomials of degree  $n$ .

**definition** *ring-cfs-to-poly* ::

*('a, 'b) ring-scheme*  $\Rightarrow$  *nat*  $\Rightarrow$  *'a list*  $\Rightarrow$  *('a, nat) mvar-poly* **where**  
*ring-cfs-to-poly*  $R$   $n$   $as = \text{coord-partial-eval } R \{1..<n+2\} (\mathbf{0}_R \# as)$  (*generic-poly*  $R$   $n$ )

**context** *cring-coord-rings*

**begin**

**lemma** *ring-cfs-to-poly-closed*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**shows** *ring-cfs-to-poly*  $R$   $n$   $as \in \text{carrier } (\text{coord-ring } R \ 1)$

*<proof>*

Variant which maps to the univariate polynomial ring

**definition** *ring-cfs-to-univ-poly* :: *nat*  $\Rightarrow$  *'a list*  $\Rightarrow$  *nat*  $\Rightarrow$  *'a* **where**

*ring-cfs-to-univ-poly*  $n$   $as = \text{IP-to-UP } (0::\text{nat})$  (*ring-cfs-to-poly*  $R$   $n$   $as$ )

**lemma** *ring-cfs-to-univ-poly-closed*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**shows** *ring-cfs-to-univ-poly*  $n$   $as \in \text{carrier } (\text{UP } R)$

*<proof>*

**lemma** *ring-cfs-to-poly-eq*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**assumes**  $k \leq n$

**shows** *ring-cfs-to-poly*  $R$   $k$   $as = \text{ring-cfs-to-poly } R$   $k$  (*take* (*Suc*  $k$ )  $as$ )

*<proof>*

**lemma** *coord-partial-eval-generic-poly-lt*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**shows** *coord-partial-eval*  $R \{1..<n+2\} (\mathbf{0}_R \# as)$  (*generic-poly-lt*  $R$   $n$ ) =  
*poly-scalar-mult*  $R$  ( $as!n$ ) (*pvar*  $R$   $0$ ) $[\ulcorner]$ *coord-ring*  $R$  ( $n+2$ )  $n$ )

*<proof>*

**lemma** *coord-partial-eval-generic-poly-lt'*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**shows** *coord-partial-eval*  $R \{1..<n+2\} (\mathbf{0}_R \# as)$  (*generic-poly-lt*  $R$   $n$ ) =  
*poly-scalar-mult*  $R$  ( $as!n$ ) (*pvar*  $R$   $0$ ) $[\ulcorner]$ *coord-ring*  $R$   $1$   $n$ )

*<proof>*

**lemma** *ring-cfs-to-poly-decomp*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } (Suc\ n)})$   
**shows**  $\text{ring-cfs-to-poly } R\ (Suc\ n)\ as = \text{ring-cfs-to-poly } R\ n\ as \oplus_{\text{coord-ring } R\ 1}$   
 $\text{poly-scalar-mult } R\ (as!(Suc\ n))\ ((\text{pvar } R\ 0)[\checkmark]_{\text{coord-ring } R\ 1}\ (Suc\ n))$   
 $\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-poly-decomp'*:  
**assumes**  $as \in \text{carrier } (R^{\text{Suc } (Suc\ n)})$   
**shows**  $\text{ring-cfs-to-poly } R\ (Suc\ n)\ as =$   
 $\text{ring-cfs-to-poly } R\ n\ (\text{take } (Suc\ n)\ as) \oplus_{\text{coord-ring } R\ 1}$   
 $\text{poly-scalar-mult } R\ (as!(Suc\ n))\ ((\text{pvar } R\ 0)[\checkmark]_{\text{coord-ring } R\ 1}\ (Suc\ n))$   
 $\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-decomp'*:  
**assumes**  $as \in \text{carrier } (R^{\text{Suc } (Suc\ n)})$   
**shows**  $\text{ring-cfs-to-univ-poly } (Suc\ n)\ as =$   
 $\text{ring-cfs-to-univ-poly } n\ (\text{take } (Suc\ n)\ as) \oplus_{UP\ R}$   
 $(as!(Suc\ n)) \odot_{UP\ R}\ (X\text{-poly } R\ [\checkmark]_{UP\ R}\ (Suc\ n))$   
 $\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-decomp*:  
**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$   
**assumes**  $k < n$   
**shows**  $\text{ring-cfs-to-univ-poly } (Suc\ k)\ (\text{take } (Suc\ (Suc\ k))\ as) = \text{ring-cfs-to-univ-poly}$   
 $k\ (\text{take } (Suc\ k)\ as)$   
 $\oplus_{UP\ R}\ (as!(Suc\ k)) \odot_{UP\ R}\ (X\text{-poly } R)[\checkmark]_{UP\ R}\ (Suc\ k)$   
 $\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-degree*:  
**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$   
**shows**  $\text{deg } R\ (\text{ring-cfs-to-univ-poly } n\ as) \leq n$   
 $as!n \neq \mathbf{0} \implies \text{deg } R\ (\text{ring-cfs-to-univ-poly } n\ as) = n$   
 $\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-constant*:  
**assumes**  $as \in \text{carrier } (R^1)$   
**shows**  $\text{ring-cfs-to-univ-poly } 0\ as = \text{to-polynomial } R\ (as!0)$   
 $\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-top-coeff*:  
**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$   
**shows**  $(\text{ring-cfs-to-univ-poly } n\ as)\ n = as!n$   
 $\langle \text{proof} \rangle$

**lemma**(in *UP-crings*) *monom-plus-lower-degree-top-coeff*:  
**assumes**  $\text{degree } p < n$   
**assumes**  $p \in \text{carrier } (UP\ R)$   
**assumes**  $a \in \text{carrier } R$   
**shows**  $(p \oplus_{UP\ R}\ (a \odot_{UP\ R}\ (X\text{-poly } R)[\checkmark]_{UP\ R}\ n))\ n = a$

$\langle \text{proof} \rangle$

**lemma**(in *UP-cring*) *monom-closed*:

**assumes**  $a \in \text{carrier } R$

**shows**  $a \odot_{UP R} ((X\text{-poly } R)[\bigwedge]_{UP R} (n::\text{nat})) \in \text{carrier } (UP R)$

$\langle \text{proof} \rangle$

**lemma**(in *UP-cring*) *monom-bottom-coeff*:

**assumes**  $a \in \text{carrier } R$

**assumes**  $n > 0$

**shows**  $(a \odot_{UP R} ((X\text{-poly } R)[\bigwedge]_{UP R} (n::\text{nat}))) = 0$

$\langle \text{proof} \rangle$

**lemma**(in *UP-cring*) *monom-plus-lower-degree-bottom-coeff*:

**assumes**  $0 < n$

**assumes**  $p \in \text{carrier } (UP R)$

**assumes**  $a \in \text{carrier } R$

**shows**  $(p \oplus_{UP R} (a \odot_{UP R} (X\text{-poly } R)[\bigwedge]_{UP R} (n::\text{nat}))) = p$

$\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-bottom-coeff*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**shows**  $(\text{ring-cfs-to-univ-poly } n \text{ as}) = as ! 0$

$\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-chain*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**assumes**  $l \leq n$

**shows**  $l \leq k \wedge k \leq n \implies (\text{ring-cfs-to-univ-poly } k (\text{take } (\text{Suc } k) \text{ as})) l = (\text{ring-cfs-to-univ-poly } l (\text{take } (\text{Suc } l) \text{ as})) l$

$\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-coeffs*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**assumes**  $l \leq n$

**shows**  $(\text{ring-cfs-to-univ-poly } n \text{ as}) l = (\text{ring-cfs-to-univ-poly } l (\text{take } (\text{Suc } l) \text{ as})) l$

$\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-coeffs'*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**assumes**  $l \leq n$

**shows**  $(\text{ring-cfs-to-univ-poly } n \text{ as}) l = as ! l$

$\langle \text{proof} \rangle$

**lemma** *ring-cfs-to-univ-poly-coeffs''*:

**assumes**  $as \in \text{carrier } (R^{\text{Suc } n})$

**shows**  $(\text{ring-cfs-to-univ-poly } n \text{ as}) l = (\text{if } l \leq n \text{ then } as ! l \text{ else } 0)$

$\langle \text{proof} \rangle$

**end**

**definition** *fun-tuple-to-univ-poly* **where**  
*fun-tuple-to-univ-poly*  $R\ n\ m\ fs\ x = \text{cring-coord-rings.ring-cfs-to-univ-poly } R\ m$   
*(function-tuple-eval*  $R\ n\ fs\ x)$

**context** *cring-coord-rings*  
**begin**

**lemma** *fun-tuple-to-univ-poly-closed*:  
**assumes** *is-function-tuple*  $R\ n\ fs$   
**assumes**  $x \in \text{carrier } (R^n)$   
**assumes**  $\text{length } fs = \text{Suc } m$   
**shows** *fun-tuple-to-univ-poly*  $R\ n\ m\ fs\ x \in \text{carrier } (UP\ R)$   
 $\langle \text{proof} \rangle$

**lemma** *fun-tuple-to-univ-poly-degree-bound*:  
**assumes** *is-function-tuple*  $R\ n\ fs$   
**assumes**  $x \in \text{carrier } (R^n)$   
**assumes**  $\text{length } fs = \text{Suc } m$   
**shows**  $\text{deg } R\ (\text{fun-tuple-to-univ-poly } R\ n\ m\ fs\ x) \leq m$   
 $\langle \text{proof} \rangle$

**lemma** *fun-tuple-to-univ-poly-degree*:  
**assumes** *is-function-tuple*  $R\ n\ fs$   
**assumes**  $x \in \text{carrier } (R^n)$   
**assumes**  $\text{length } fs = \text{Suc } m$   
**assumes**  $(fs!m)\ x \neq \mathbf{0}$   
**shows**  $\text{deg } R\ (\text{fun-tuple-to-univ-poly } R\ n\ m\ fs\ x) = m$   
 $\langle \text{proof} \rangle$

## 6.4 Factoring a Polynomial as a Univariate Polynomial over a Multivariable Polynomial Ring

**definition** *pre-to-univ-poly-hom*  $:: \text{nat} \Rightarrow \text{nat} \Rightarrow ('a, (( 'a, \text{nat}) \text{mvar-poly}, \text{nat}) \text{mvar-poly}) \text{ring-hom}$  **where**  
*pre-to-univ-poly-hom*  $n\ i = MP.\text{indexed-const } (n-1) \circ R.\text{indexed-const}$

**lemma** *pre-to-univ-poly-hom-is-hom*:  
**assumes**  $i < n$   
**shows** *ring-hom-ring*  $R\ (\text{Pring } (\text{coord-ring } R\ (n-1))\ \{i\})\ (\text{pre-to-univ-poly-hom } n\ i)$   
 $\langle \text{proof} \rangle$

**definition** *pre-to-univ-poly-var-ass*  $::$   
 $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow (('a, \text{nat}) \text{mvar-poly}, \text{nat}) \text{mvar-poly}$  **where**  
*pre-to-univ-poly-var-ass*  $n\ i\ j = (\text{if } j < i \text{ then } MP.\text{indexed-const } (n-1)\ (\text{pvar } R\ j) \text{ else}$   
 $(\text{if } j = i \text{ then } \text{pvar } (\text{coord-ring } R\ (n-1))\ i \text{ else}$

1)) else  

$$(if\ j < n\ then\ MP.indexed-const\ (n-1)\ (pvar\ R\ (j - 1)))\ else\ \mathbf{0}Pring\ (coord-ring\ R\ (n-1))\ \{i\}))$$

**lemma** *pre-to-univ-poly-var-ass-closed*:

**assumes**  $i < n$   
**shows**  $closed-fun\ (Pring\ (coord-ring\ R\ (n-1))\ \{i\})\ (pre-to-univ-poly-var-ass\ n\ i)$   
 $\langle proof \rangle$

**lemma** *pre-to-univ-poly-var-ass-closed'*:

**assumes**  $i < n$   
**shows**  $(pre-to-univ-poly-var-ass\ n\ i) \in \{..<n\} \rightarrow carrier\ (Pring\ (coord-ring\ R\ (n-1))\ \{i\})$   
 $\langle proof \rangle$

**definition** *pre-to-univ-poly* ::

$nat \Rightarrow nat \Rightarrow ((!a, nat)\ mvar-poly, (!!a, nat)\ mvar-poly, nat)\ mvar-poly$  ring-hom

**where**

$pre-to-univ-poly\ (n::nat)\ (i::nat) = indexed-poly-induced-morphism\ \{..<n\}\ (Pring\ (coord-ring\ R\ (n-1))\ \{i\})$

$(pre-to-univ-poly-hom\ n\ i)$   
 $(pre-to-univ-poly-var-ass\ n\ i)$

**lemma** *pre-to-univ-poly-is-hom*:

**assumes**  $i < n$   
**assumes**  $\psi = pre-to-univ-poly\ n\ i$   
**shows**  $ring-hom-ring\ (R[\mathcal{X}_n])\ (Pring\ (coord-ring\ R\ (n-1))\ \{i\})\ \psi$   
 $\bigwedge j. j < i \implies \psi\ (pvar\ R\ j) = MP.indexed-const\ (n-1)\ (pvar\ R\ j)$   
 $\psi\ (pvar\ R\ i) = pvar\ (coord-ring\ R\ (n-1))\ i$   
 $\bigwedge j. i < j \wedge j < n \implies \psi\ (pvar\ R\ j) = MP.indexed-const\ (n-1)\ (pvar\ R\ (j - 1))$   
 $\bigwedge a. a \in carrier\ R \implies \psi\ (coord-const\ a) = MP.indexed-const\ (n-1)\ (coord-const\ a)$   
 $\bigwedge p. p \in carrier\ (R[\mathcal{X}_n]) \implies pre-to-univ-poly\ n\ i\ p \in carrier\ (Pring\ (coord-ring\ R\ (n-1))\ \{i\})$   
 $\langle proof \rangle$

**lemma** *insert-at-index-closed*:

**assumes**  $a \in carrier\ (R^n)$   
**assumes**  $x \in carrier\ R$   
**assumes**  $i \leq n$   
**shows**  $insert-at-index\ a\ x\ i \in carrier\ (R^{Suc\ n})$   
 $\langle proof \rangle$

**lemma** *pre-to-univ-poly-eval*:

**assumes**  $i < Suc\ n$   
**assumes**  $p \in carrier\ (R[\mathcal{X}_{Suc\ n}])$   
**assumes**  $a \in carrier\ (R^n)$   
**assumes**  $x \in carrier\ R$

**assumes**  $as = \text{insert-at-index } a \ x \ i$   
**shows**  $\text{eval-at-point } R \ \text{as } p = \text{eval-at-point } R \ a \ (\text{total-eval } (R[\mathcal{X}_n]) \ (\lambda \ i. \ \text{coord-const } x) \ (\text{pre-to-univ-poly } (\text{Suc } n) \ i \ p))$   
 $\langle \text{proof} \rangle$

**definition**  $\text{pre-to-univ-poly-inv-hom} ::$   
 $\text{nat} \Rightarrow \text{nat} \Rightarrow ((\ 'a, \text{nat}) \ \text{mvar-poly}, (\ 'a, \text{nat}) \ \text{mvar-poly}) \ \text{ring-hom}$  **where**  
 $\text{pre-to-univ-poly-inv-hom } n \ i = R.\text{relabel-vars } \{..<(n-1)\} \ \{..<n\} \ (\lambda j. \ \text{if } j < i \ \text{then } j \ \text{else } j + 1)$

**lemma**  $\text{pre-to-univ-poly-inv-hom-is-hom}$ :  
**assumes**  $i < \text{Suc } n$   
**shows**  $\text{ring-hom-ring } (R[\mathcal{X}_n]) \ (R[\mathcal{X}_{\text{Suc } n}]) \ (\text{pre-to-univ-poly-inv-hom } (\text{Suc } n) \ i)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{pre-to-univ-poly-inv-hom-const}$ :  
**assumes**  $i < \text{Suc } n$   
**assumes**  $k \in \text{carrier } R$   
**shows**  $(\text{pre-to-univ-poly-inv-hom } (\text{Suc } n) \ i) \ (R.\text{indexed-const } k) = R.\text{indexed-const } k$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{pre-to-univ-poly-inv-hom-pvar-0}$ :  
**assumes**  $i < \text{Suc } n$   
**assumes**  $j < i$   
**shows**  $\text{pre-to-univ-poly-inv-hom } (\text{Suc } n) \ i \ (\text{pvar } R \ j) = \text{pvar } R \ j$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{pre-to-univ-poly-inv-hom-pvar-1}$ :  
**assumes**  $i < \text{Suc } n$   
**assumes**  $i \leq j$   
**assumes**  $j < n$   
**shows**  $\text{pre-to-univ-poly-inv-hom } (\text{Suc } n) \ i \ (\text{pvar } R \ j) = \text{pvar } R \ (j + 1)$   
 $\langle \text{proof} \rangle$

**definition**  $\text{pre-to-univ-poly-inv-var-ass} ::$   
 $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow (\ 'a, \text{nat}) \ \text{mvar-poly}$  **where**  
 $\text{pre-to-univ-poly-inv-var-ass } n \ i \ j = \text{pvar } R \ i$

**lemma**  $\text{pre-to-univ-poly-inv-var-ass-closed}$ :  
**assumes**  $i < \text{Suc } n$   
**shows**  $\text{pre-to-univ-poly-inv-var-ass } (\text{Suc } n) \ i \in \{i\} \rightarrow \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$   
 $\langle \text{proof} \rangle$

**definition**  $\text{pre-to-univ-poly-inv} ::$   
 $\text{nat} \Rightarrow \text{nat} \Rightarrow ((\ 'a, \text{nat}) \ \text{mvar-poly}, (\ 'a, \text{nat}) \ \text{mvar-poly}) \ \text{ring-hom}$   
**where**

*pre-to-univ-poly-inv*  $n$   $i$  = *indexed-poly-induced-morphism*  $\{i\}$  ( $R[\mathcal{X}_n]$ )  
 (*pre-to-univ-poly-inv-hom*  $n$   $i$ ) (*pre-to-univ-poly-inv-var-ass*  
 $n$   $i$ )

**lemma** *pre-to-univ-poly-inv-is-hom*:

**assumes**  $i < \text{Suc } n$   
**shows** *ring-hom-ring* (*Pring* ( $R[\mathcal{X}_n]$ )  $\{i\}$ ) ( $R[\mathcal{X}_{\text{Suc } n}]$ ) (*pre-to-univ-poly-inv* ( $\text{Suc } n$ )  $i$ )  
 $\langle \text{proof} \rangle$

**lemma** *pre-to-univ-poly-inv-pvar*:

**assumes**  $i < \text{Suc } n$   
**shows** (*pre-to-univ-poly-inv* ( $\text{Suc } n$ )  $i$ ) (*pvar* ( $R[\mathcal{X}_n]$ )  $i$ ) = *pvar*  $R$   $i$   
 $\langle \text{proof} \rangle$

**lemma** *pre-to-univ-poly-inv-const*:

**assumes**  $i < \text{Suc } n$   
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**shows** (*pre-to-univ-poly-inv* ( $\text{Suc } n$ )  $i$ ) (*ring.indexed-const* ( $R[\mathcal{X}_n]$ )  $p$ ) = *pre-to-univ-poly-inv-hom*  
 ( $\text{Suc } n$ )  $i$   $p$   
 $\langle \text{proof} \rangle$

**lemma** *pre-to-univ-poly-inverse*:

**assumes**  $i < \text{Suc } n$   
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$   
**shows** *pre-to-univ-poly-inv* ( $\text{Suc } n$ )  $i$  (*pre-to-univ-poly* ( $\text{Suc } n$ )  $i$   $p$ ) =  $p$   
 $\langle \text{proof} \rangle$

**lemma** *coord-ring-car-induct*:

**assumes**  $Q \in \text{carrier } (R[\mathcal{X}_n])$   
**assumes**  $\bigwedge c. c \in \text{carrier } R \implies A$  (*R.indexed-const*  $c$ )  
**assumes**  $\bigwedge p q. p \in \text{carrier } (R[\mathcal{X}_n]) \implies q \in \text{carrier } (R[\mathcal{X}_n]) \implies A$   $p \implies A$   $q$   
 $\implies A$  ( $p \oplus_{R[\mathcal{X}_n]} q$ )  
**assumes**  $\bigwedge p i. p \in \text{carrier } (R[\mathcal{X}_n]) \implies i < n \implies A$   $p \implies A$  ( $p \otimes_{R[\mathcal{X}_n]} \text{pvar } R$   $i$ )  
**shows**  $A$   $Q$   
 $\langle \text{proof} \rangle$

**lemma** *pre-to-univ-poly-inverse'*:

**assumes**  $i < \text{Suc } n$   
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_n])$   
**shows** *pre-to-univ-poly* ( $\text{Suc } n$ )  $i$  (*pre-to-univ-poly-inv* ( $\text{Suc } n$ )  $i$  (*MP.indexed-const*  
 $n$   $p$ )) = *MP.indexed-const*  $n$   $p$   
 $\langle \text{proof} \rangle$

**definition** *to-univ-poly* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow$

$((\text{'a}, \text{nat}) \text{mvar-poly}, (\text{'a}, \text{nat}) \text{mvar-poly u-poly})$  *ring-hom* **where**  
*to-univ-poly*  $n$   $i$  = *IP-to-UP*  $i$   $\circ$  (*pre-to-univ-poly*  $n$   $i$ )

**definition** *from-univ-poly* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow$   
 $(('a, \text{nat}) \text{ mvar-poly } u\text{-poly}, ('a, \text{nat}) \text{ mvar-poly}) \text{ ring-hom}$  **where**  
*from-univ-poly*  $n \ i = \text{pre-to-univ-poly-inv } n \ i \circ (\text{UP-to-IP } (\text{coord-ring } R \ (n-1)) \ i)$

**lemma** *to-univ-poly-is-hom*:

**assumes**  $i \leq n$

**shows**  $(\text{to-univ-poly } (\text{Suc } n) \ i) \in \text{ring-hom } (R[\mathcal{X}_{\text{Suc } n}]) \ (UP \ (R[\mathcal{X}_n]))$

$\langle \text{proof} \rangle$

**lemma** *from-univ-poly-is-hom*:

**assumes**  $i \leq n$

**shows**  $(\text{from-univ-poly } (\text{Suc } n) \ i) \in \text{ring-hom } (UP \ (R[\mathcal{X}_n])) \ (R[\mathcal{X}_{\text{Suc } n}])$

$\langle \text{proof} \rangle$

**lemma** *to-univ-poly-inverse*:

**assumes**  $i \leq n$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$

**shows**  $\text{from-univ-poly } (\text{Suc } n) \ i \ (\text{to-univ-poly } (\text{Suc } n) \ i \ p) = p$

$\langle \text{proof} \rangle$

**lemma** *to-univ-poly-closed*:

**assumes**  $i \leq n$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$

**shows**  $\text{to-univ-poly } (\text{Suc } n) \ i \ p \in \text{carrier } (UP \ (R[\mathcal{X}_n]))$

$\langle \text{proof} \rangle$

**lemma** *to-univ-poly-add*:

**assumes**  $i \leq n$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$

**assumes**  $Q \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$

**shows**  $\text{to-univ-poly } (\text{Suc } n) \ i \ (p \oplus_{R[\mathcal{X}_{\text{Suc } n}]} Q) =$

$\text{to-univ-poly } (\text{Suc } n) \ i \ p \oplus_{UP \ (R[\mathcal{X}_n])} \text{to-univ-poly } (\text{Suc } n) \ i \ Q$

$\langle \text{proof} \rangle$

**lemma** *to-univ-poly-mult*:

**assumes**  $i \leq n$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$

**assumes**  $Q \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$

**shows**  $\text{to-univ-poly } (\text{Suc } n) \ i \ (p \otimes_{R[\mathcal{X}_{\text{Suc } n}]} Q) =$

$\text{to-univ-poly } (\text{Suc } n) \ i \ p \otimes_{UP \ (R[\mathcal{X}_n])} \text{to-univ-poly } (\text{Suc } n) \ i \ Q$

$\langle \text{proof} \rangle$

**lemma** *from-univ-poly-closed*:

**assumes**  $i \leq n$

**assumes**  $p \in \text{carrier } (UP \ (R[\mathcal{X}_n]))$

**shows**  $\text{from-univ-poly } (\text{Suc } n) \ i \ p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$

$\langle \text{proof} \rangle$

**lemma** *from-univ-poly-add*:

**assumes**  $i \leq n$

**assumes**  $p \in \text{carrier } (UP (R[\mathcal{X}_n]))$

**assumes**  $Q \in \text{carrier } (UP (R[\mathcal{X}_n]))$

**shows**  $\text{from-univ-poly } (Suc\ n)\ i\ (p \oplus_{UP (R[\mathcal{X}_n])} Q) =$

$\text{from-univ-poly } (Suc\ n)\ i\ p \oplus_{R[\mathcal{X}_{Suc\ n}]} \text{from-univ-poly } (Suc\ n)\ i\ Q$

*<proof>*

**lemma** *from-univ-poly-mult*:

**assumes**  $i \leq n$

**assumes**  $p \in \text{carrier } (UP (R[\mathcal{X}_n]))$

**assumes**  $Q \in \text{carrier } (UP (R[\mathcal{X}_n]))$

**shows**  $\text{from-univ-poly } (Suc\ n)\ i\ (p \otimes_{UP (R[\mathcal{X}_n])} Q) =$

$\text{from-univ-poly } (Suc\ n)\ i\ p \otimes_{R[\mathcal{X}_{Suc\ n}]} \text{from-univ-poly } (Suc\ n)\ i\ Q$

*<proof>*

**lemma**(**in** *UP-cring*) *monom-as-mult*:

**assumes**  $a \in \text{carrier } R$

**shows**  $\text{up-ring.monom } (UP\ R)\ a\ n = \text{to-poly } a \otimes_{UP\ R} \text{up-ring.monom } (UP\ R)$

**1**  $n$

*<proof>*

**lemma** *cring-coord-rings-coord-ring*:

*cring-coord-rings*  $(R[\mathcal{X}_n])$

*<proof>*

**lemma** *from-univ-poly-monom-inverse*:

**assumes**  $i < Suc\ n$

**assumes**  $a \in \text{carrier } (R[\mathcal{X}_n])$

**shows**  $\text{to-univ-poly } (Suc\ n)\ i\ (\text{from-univ-poly } (Suc\ n)\ i\ (\text{up-ring.monom } (UP\ (R[\mathcal{X}_n]))\ a\ m)) = \text{up-ring.monom } (UP\ (R[\mathcal{X}_n]))\ a\ m$

*<proof>*

**lemma** *from-univ-poly-inverse*:

**assumes**  $i \leq n$

**assumes**  $p \in \text{carrier } (UP (R[\mathcal{X}_n]))$

**shows**  $\text{to-univ-poly } (Suc\ n)\ i\ (\text{from-univ-poly } (Suc\ n)\ i\ p) = p$

*<proof>*

**lemma** *to-univ-poly-eval*:

**assumes**  $i < Suc\ n$

**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{Suc\ n}])$

**assumes**  $a \in \text{carrier } (R^n)$

**assumes**  $x \in \text{carrier } R$

**assumes**  $as = \text{insert-at-index } a\ x\ i$

**shows**  $\text{eval-at-point } R\ as\ p = \text{eval-at-point } R\ a\ (\text{to-function } (R[\mathcal{X}_n])\ (\text{to-univ-poly } (Suc\ n)\ i\ p)\ (\text{coord-const } x))$

*<proof>*

The function `one_over_poly`, introduced in the theory `Cring_Poly`, maps a polynomial  $p(x)$  to the unique polynomial  $q(x)$  which satisfies the relation  $q(x) = x^n p(1/x)$ . This will be used later to show that the function  $f(x) = 1/x$  is semialgebraic over the field  $\mathbb{Q}_p$ .

**lemma** *to-univ-poly-one-over-poly*:

**assumes** *field*  $R$   
**assumes**  $i < (\text{Suc } n)$   
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$   
**assumes**  $Q = \text{from-univ-poly } (\text{Suc } n) \ i \ (UP\text{-cring.one-over-poly } (R[\mathcal{X}_n]) \ (to\text{-univ-poly } (\text{Suc } n) \ i \ p))$   
**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $x \in \text{carrier } R$   
**assumes**  $x \neq \mathbf{0}$   
**assumes**  $b = \text{insert-at-index } a \ x \ i$   
**assumes**  $c = \text{insert-at-index } a \ (\text{inv } x) \ i$   
**assumes**  $N = UP\text{-ring.degree } (R[\mathcal{X}_n]) \ (to\text{-univ-poly } (\text{Suc } n) \ i \ p)$   
**shows**  $Q \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$   
 $\text{eval-at-point } R \ b \ Q = (x[\uparrow]N) \otimes (\text{eval-at-point } R \ c \ p)$   
*<proof>*

**lemma** *to-univ-poly-one-over-poly'*:

**assumes** *field*  $R$   
**assumes**  $i < (\text{Suc } n)$   
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$   
**assumes**  $Q = \text{from-univ-poly } (\text{Suc } n) \ i \ (UP\text{-cring.one-over-poly } (R[\mathcal{X}_n]) \ (to\text{-univ-poly } (\text{Suc } n) \ i \ p))$   
**assumes**  $a \in \text{carrier } (R^n)$   
**assumes**  $x \in \text{carrier } R$   
**assumes**  $x \neq \mathbf{0}$   
**assumes**  $b = \text{insert-at-index } a \ x \ i$   
**assumes**  $c = \text{insert-at-index } a \ (\text{inv } x) \ i$   
**assumes**  $N = UP\text{-ring.degree } (R[\mathcal{X}_n]) \ (to\text{-univ-poly } (\text{Suc } n) \ i \ p)$   
**assumes**  $q = (\text{pvar } R \ i)[\uparrow]_{R[\mathcal{X}_{\text{Suc } n}]}^{(k::nat)} \otimes_{R[\mathcal{X}_{\text{Suc } n}]} Q$   
**shows**  $q \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$   
 $\text{eval-at-point } R \ b \ q = (x[\uparrow](N + k)) \otimes (\text{eval-at-point } R \ c \ p)$   
*<proof>*

**lemma** *to-univ-poly-one-over-poly''*:

**assumes** *field*  $R$   
**assumes**  $i < (\text{Suc } n)$   
**assumes**  $p \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}])$   
**assumes**  $N \geq UP\text{-ring.degree } (R[\mathcal{X}_n]) \ (to\text{-univ-poly } (\text{Suc } n) \ i \ p)$   
**shows**  $\exists q \in \text{carrier } (R[\mathcal{X}_{\text{Suc } n}]). (\forall x \in \text{carrier } R - \{\mathbf{0}\}. (\forall a \in \text{carrier } (R^n).$   
 $\text{eval-at-point } R \ (\text{insert-at-index } a \ x \ i) \ q = (x[\uparrow]N) \otimes (\text{eval-at-point } R$   
 $(\text{insert-at-index } a \ (\text{inv } x) \ i) \ p))$   
*<proof>*

## 7 Restricted Inverse Images and Complements

This section introduces some versions of basic set operations for extensional functions and sets. We would like a version of the inverse image which intersects the inverse image of a function with the set `carrier` ( $R^n$ ), and a version of the complement of a set which takes the complement relative to `carrier` ( $R^n$ ). These will have to be defined in parametrized families, with one such object for each natural number  $n$ .

**definition** *evimage* (`infixr`  $^{-1}_1$  90) **where**  
 $f^{-1}_n S = ((f \text{ - ' } S) \cap \text{carrier } (R^n))$

**definition** *euminus-set* ::  $\text{nat} \Rightarrow 'a \text{ list set} \Rightarrow 'a \text{ list set}$  (`-c` 70) **where**  
 $S^c_n = \text{carrier } (R^n) - S$

**lemma** *extensional-vimage-closed*:  
 $f^{-1}_n S \subseteq \text{carrier } (R^n)$   
 $\langle \text{proof} \rangle$

### 7.1 Inverse image of a function

**lemma** *evimage-eq* [`simp`]:  $a \in f^{-1}_n B \longleftrightarrow a \in \text{carrier } (R^n) \wedge f a \in B$   
 $\langle \text{proof} \rangle$

**lemma** *evimage-singleton-eq*:  $a \in f^{-1}_n \{b\} \longleftrightarrow a \in \text{carrier } (R^n) \wedge f a = b$   
 $\langle \text{proof} \rangle$

**lemma** *evimageI* [`intro`]:  $a \in \text{carrier } (R^n) \Longrightarrow f a = b \Longrightarrow b \in B \Longrightarrow a \in f^{-1}_n B$   
 $\langle \text{proof} \rangle$

**lemma** *evimageI2*:  $a \in \text{carrier } (R^n) \Longrightarrow f a \in A \Longrightarrow a \in f^{-1}_n A$   
 $\langle \text{proof} \rangle$

**lemma** *evimageE* [`elim!`]:  $a \in f^{-1}_n B \Longrightarrow (\bigwedge x. f a = x \Longrightarrow x \in B \Longrightarrow p) \Longrightarrow p$   
 $\langle \text{proof} \rangle$

**lemma** *evimageD*:  $a \in f^{-1}_n A \Longrightarrow f a \in A$   
 $\langle \text{proof} \rangle$

**lemma** *evimage-empty* [`simp`]:  $f^{-1}_n \{\} = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *evimage-Compl*:  
**assumes**  $f \in \text{carrier } (R^n) \rightarrow \text{carrier } (R^m)$   
**shows**  $(f^{-1}_n (A^c_m)) = ((f \text{ - ' } A)^c_n)$   
 $\langle \text{proof} \rangle$

**lemma** *evimage-Un* [`simp`]:  $f^{-1}_n (A \cup B) = (f^{-1}_n A) \cup (f^{-1}_n B)$

*<proof>*

**lemma** *evimage-Int* [simp]:  $f^{-1}_n (A \cap B) = (f^{-1}_n A) \cap (f^{-1}_n B)$   
*<proof>*

**lemma** *evimage-Collect-eq* [simp]:  $f^{-1}_n \text{Collect } p = \{y \in \text{carrier } (R^n). p (f y)\}$   
*<proof>*

**lemma** *evimage-Collect*:  $(\bigwedge x. x \in \text{carrier } (R^n) \implies p (f x) = Q x) \implies f^{-1}_n (\text{Collect } p) = \text{Collect } Q \cap \text{carrier } (R^n)$   
*<proof>*

**lemma** *evimage-insert*:  $f^{-1}_n (\text{insert } a B) = (f^{-1}_n \{a\}) \cup (f^{-1}_n B)$   
— NOT suitable for rewriting because of the recurrence of  $\{a\}$ .  
*<proof>*

**lemma** *evimage-Diff*:  $f^{-1}_n (A - B) = (f^{-1}_n A) - (f^{-1}_n B)$   
*<proof>*

**lemma** *evimage-UNIV* [simp]:  $f^{-1}_n \text{UNIV} = \text{carrier } (R^n)$   
*<proof>*

**lemma** *evimage-mono*:  $A \subseteq B \implies f^{-1}_n A \subseteq f^{-1}_n B$   
— monotonicity  
*<proof>*

**lemma** *evimage-image-eq*:  $(f^{-1}_n (f ' A)) = \{y \in \text{carrier } (R^n). \exists x \in A. f x = f y\}$   
*<proof>*

**lemma** *image-evimage-subset*:  $f ' (f^{-1}_n A) \subseteq A$   
*<proof>*

**lemma** *image-evimage-eq* [simp]:  $f ' (f^{-1}_n A) = A \cap (f ' \text{carrier } (R^n))$   
*<proof>*

**lemma** *image-subset-iff-subset-evimage*:  $A \subseteq \text{carrier } (R^n) \implies f ' A \subseteq B \iff A \subseteq f^{-1}_n B$   
*<proof>*

**lemma** *evimage-const* [simp]:  $((\lambda x. c)^{-1}_n A) = (\text{if } c \in A \text{ then } \text{carrier } (R^n) \text{ else } \{\})$   
*<proof>*

**lemma** *evimage-if* [simp]:  $((\lambda x. \text{if } x \in B \text{ then } c \text{ else } d)^{-1}_n A) =$   
 $(\text{if } c \in A \text{ then } (\text{if } d \in A \text{ then } \text{carrier } (R^n) \text{ else } B \cap \text{carrier } (R^n))$   
 $\text{else if } d \in A \text{ then } B^c \text{ else } \{\})$   
*<proof>*

**lemma** *evimage-inter-cong*:  $(\bigwedge w. w \in S \implies f w = g w) \implies f^{-1}_n y \cap S = g$

${}^{-1}_n y \cap S$   
 $\langle proof \rangle$

**lemma** *evimage-ident* [*simp*]:  $(\lambda x. x) {}^{-1}_n Y = Y \cap carrier (R^n)$   
 $\langle proof \rangle$

**end**

**end**  
**theory** *Padic-Fields*  
**imports** *Fraction-Field Padic-Ints.Hensels-Lemma*

**begin**

## 8 Constructing the $p$ -adic Valued Field

As a field, we can define the field  $\mathbb{Q}_p$  immediately as the fraction field of  $\mathbb{Z}_p$ . The valuation can then be extended from  $\mathbb{Z}_p$  to  $\mathbb{Q}_p$  by defining  $\text{val}(a/b) = \text{val } a - \text{val } b$  where  $a, b \in \mathbb{Z}_p$ .

### 8.1 A Locale for $p$ -adic Fields

This section builds a locale for reasoning about general  $p$ -adic fields for a fixed  $p$ . The locale fixes constants for the ring of  $p$ -adic integers ( $\mathbb{Z}_p$ ) and the inclusion map  $\iota : \mathbb{Z}_p \rightarrow \mathbb{Q}_p$ .

**type-synonym** *padic-number* =  $((\text{nat} \Rightarrow \text{int}) \times (\text{nat} \Rightarrow \text{int})) \text{ set}$

**locale** *padic-fields*=

**fixes**  $Q_p:: \text{- ring (structure)}$

**fixes**  $Z_p:: \text{- ring (structure)}$

**fixes**  $p$

**fixes**  $\iota$

**defines**  $Z_p \equiv \text{padic-int } p$

**defines**  $Q_p \equiv \text{Frac } Z_p$

**defines**  $\iota \equiv \text{domain-frac.inc } Z_p$

**assumes** *prime*:  $\text{prime } p$

**sublocale** *padic-fields* <  $Zp?$ : *domain-frac*  $Z_p$

$\langle proof \rangle$

**sublocale** *padic-fields* <  $Qp?$ : *ring*  $Q_p$

$\langle proof \rangle$

**sublocale** *padic-fields* <  $Qp?$ : *cring*  $Q_p$

*<proof>*

**sublocale** *padic-fields* < *Qp?*: field  $\mathbb{Q}_p$

*<proof>*

**sublocale** *padic-fields* < *Qp?*: domain  $\mathbb{Q}_p$

*<proof>*

**sublocale** *padic-fields* < *padic-integers*  $\mathbb{Z}_p$

*<proof>*

**sublocale** *padic-fields* < *UPQ?*: *UP-cring*  $\mathbb{Q}_p$  *UP*  $\mathbb{Q}_p$

*<proof>*

## 8.2 The Valuation Ring in $\mathbb{Q}_p$

The valuation ring  $\mathcal{O}_p$  is the subring of elements in  $\mathbb{Q}_p$  with positive valuation. It is an isomorphic copy of  $\mathbb{Z}_p$ .

**context** *padic-fields*

**begin**

**abbreviation**  $\mathcal{O}_p$  where

$\mathcal{O}_p \equiv \iota$  ' carrier  $\mathbb{Z}_p$

**lemma** *inc-closed*:

**assumes**  $a \in$  carrier  $\mathbb{Z}_p$

**shows**  $\iota a \in$  carrier  $\mathbb{Q}_p$

*<proof>*

**lemma** *inc-is-hom*:

$\iota \in$  ring-hom  $\mathbb{Z}_p$   $\mathbb{Q}_p$

*<proof>*

An alternate formula of the map  $\iota$

**lemma** *inc-def*:

**assumes**  $a \in$  carrier  $\mathbb{Z}_p$

**shows**  $\iota a = \text{frac } a \mathbf{1}_{\mathbb{Z}_p}$

*<proof>*

**lemma** *inc-of-nonzero*:

**assumes**  $a \in$  nonzero  $\mathbb{Z}_p$

**shows**  $\iota a \in$  nonzero  $\mathbb{Q}_p$

*<proof>*

**lemma** *inc-of-one*:

$\iota \mathbf{1}_{\mathbb{Z}_p} = \mathbf{1}$

*<proof>*

**lemma** *inc-of-zero*:

$\iota \mathbf{0}_{Z_p} = \mathbf{0}$

*<proof>*

**lemma** *inc-of-sum*:

**assumes**  $a \in \text{carrier } Z_p$

**assumes**  $b \in \text{carrier } Z_p$

**shows**  $\iota (a \oplus_{Z_p} b) = (\iota a) \oplus (\iota b)$

*<proof>*

**lemma** *inc-of-prod*:

**assumes**  $a \in \text{carrier } Z_p$

**assumes**  $b \in \text{carrier } Z_p$

**shows**  $\iota (a \otimes_{Z_p} b) = (\iota a) \otimes (\iota b)$

*<proof>*

**lemma** *inc-pow*:

**assumes**  $a \in \text{nonzero } Z_p$

**shows**  $\iota (a[\wedge]_{Z_p} (n::\text{nat})) = (\iota a)[\wedge] n$

*<proof>*

**lemma** *inc-of-diff*:

**assumes**  $a \in \text{carrier } Z_p$

**assumes**  $b \in \text{carrier } Z_p$

**shows**  $\iota (a \ominus_{Z_p} b) = (\iota a) \ominus (\iota b)$

*<proof>*

**lemma** *Units-nonzero-Qp*:

**assumes**  $u \in \text{Units } Q_p$

**shows**  $u \in \text{nonzero } Q_p$

*<proof>*

**lemma** *Units-eq-nonzero*:

$\text{Units } Q_p = \text{nonzero } Q_p$

*<proof>*

**lemma** *Units-inverse-Qp*:

**assumes**  $u \in \text{Units } Q_p$

**shows**  $\text{inv}_{Q_p} u \in \text{Units } Q_p$

*<proof>*

**lemma** *nonzero-inverse-Qp*:

**assumes**  $u \in \text{nonzero } Q_p$

**shows**  $\text{inv}_{Q_p} u \in \text{nonzero } Q_p$

*<proof>*

**lemma** *frac-add*:

**assumes**  $a \in \text{carrier } Z_p$

**assumes**  $b \in \text{nonzero } Z_p$   
**assumes**  $c \in \text{carrier } Z_p$   
**assumes**  $d \in \text{nonzero } Z_p$   
**shows**  $(\text{frac } a \ b) \oplus (\text{frac } c \ d) = (\text{frac } ((a \otimes_{Z_p} d) \oplus_{Z_p} (b \otimes_{Z_p} c)) (b \otimes_{Z_p} d))$   
 <proof>

**lemma** *frac-add-common-denom:*

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $b \in \text{carrier } Z_p$   
**assumes**  $c \in \text{nonzero } Z_p$   
**shows**  $(\text{frac } a \ c) \oplus (\text{frac } b \ c) = \text{frac } (a \oplus_{Z_p} b) \ c$   
 <proof>

**lemma** *frac-mult:*

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**assumes**  $c \in \text{carrier } Z_p$   
**assumes**  $d \in \text{nonzero } Z_p$   
**shows**  $(\text{frac } a \ b) \otimes (\text{frac } c \ d) = (\text{frac } (a \otimes_{Z_p} c) (b \otimes_{Z_p} d))$   
 <proof>

**lemma** *frac-one:*

**assumes**  $a \in \text{nonzero } Z_p$   
**shows**  $\text{frac } a \ a = \mathbf{1}$   
 <proof>

**lemma** *frac-closed:*

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**shows**  $\text{frac } a \ b \in \text{carrier } Q_p$   
 <proof>

**lemma** *inv-in-frac:*

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $a \neq \mathbf{0}$   
**shows**  $\text{inv}_{Q_p} a \in \text{carrier } Q_p$   
      $\text{inv}_{Q_p} a \neq \mathbf{0}$   
      $\text{inv}_{Q_p} a \in \text{nonzero } Q_p$   
 <proof>

**lemma** *nonzero-numer-imp-nonzero-fraction:*

**assumes**  $a \in \text{nonzero } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**shows**  $\text{frac } a \ b \neq \mathbf{0}$   
 <proof>

**lemma** *nonzero-fraction-imp-numer-not-zero:*

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$

**assumes**  $\text{frac } a \ b \neq \mathbf{0}$   
**shows**  $a \neq \mathbf{0}_{Z_p}$   
 $\langle \text{proof} \rangle$

**lemma** *nonzero-fraction-imp-nonzero-numer*:

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**assumes**  $\text{frac } a \ b \neq \mathbf{0}$   
**shows**  $a \in \text{nonzero } Z_p$   
 $\langle \text{proof} \rangle$

**lemma**(in *padic-fields*) *frac-inv-id*:

**assumes**  $a \in \text{nonzero } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**assumes**  $c \in \text{nonzero } Z_p$   
**assumes**  $d \in \text{nonzero } Z_p$   
**assumes**  $\text{frac } a \ b = \text{frac } c \ d$   
**shows**  $\text{frac } b \ a = \text{frac } d \ c$   
 $\langle \text{proof} \rangle$

**lemma**(in *padic-fields*) *frac-uminus*:

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**shows**  $\ominus (\text{frac } a \ b) = \text{frac } (\ominus_{Z_p} a) \ b$   
 $\langle \text{proof} \rangle$

**lemma**(in *padic-fields*) *i-mult*:

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $c \in \text{carrier } Z_p$   
**assumes**  $d \in \text{nonzero } Z_p$   
**shows**  $(\iota a) \otimes (\text{frac } c \ d) = \text{frac } (a \otimes_{Z_p} c) \ d$   
 $\langle \text{proof} \rangle$

**lemma** *numer-denom-facts*:

**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $(\text{numer } a) \in \text{carrier } Z_p$   
 $(\text{denom } a) \in \text{nonzero } Z_p$   
 $a \neq \mathbf{0} \implies \text{numer } a \neq \mathbf{0}_{Z_p}$   
 $a \otimes (\iota (\text{denom } a)) = \iota (\text{numer } a)$   
 $a = \text{frac } (\text{numer } a) \ (\text{denom } a)$   
 $\langle \text{proof} \rangle$

**lemma** *get-common-denominator*:

**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{carrier } Q_p$   
**obtains**  $a \ b \ c$  **where**  
 $a \in \text{carrier } Z_p$   
 $b \in \text{carrier } Z_p$   
 $c \in \text{nonzero } Z_p$

$x = \text{frac } a \ c$   
 $y = \text{frac } b \ c$   
 $\langle \text{proof} \rangle$

**abbreviation**  $\text{frac} :: - \Rightarrow - \Rightarrow -$  (**infixl**  $\div$  50) **where**  
 $(\text{frac } a \ b) \equiv (a \otimes (\text{inv}_{Q_p} \ b))$

$\text{frac}$  generalizes  $\text{frac}$

**lemma**  $\text{frac-frac}$ :

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**shows**  $(\text{frac } a \ b) = (\iota \ a \ \div \ \iota \ b)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{frac-eq}$ :

**assumes**  $a \in \text{nonzero } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**assumes**  $\text{frac } a \ b = \mathbf{1}$   
**shows**  $a = b$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{frac-cancel-right}$ :

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{nonzero } Q_p$   
**shows**  $b \otimes (a \ \div \ b) = a$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{frac-cancel-left}$ :

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{nonzero } Q_p$   
**shows**  $(a \ \div \ b) \otimes b = a$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{frac-mult}$ :

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{nonzero } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $d \in \text{nonzero } Q_p$   
**shows**  $(a \ \div \ b) \otimes (c \ \div \ d) = ((a \ \otimes \ c) \ \div \ (b \ \otimes \ d))$   
 $\langle \text{proof} \rangle$

**lemma**  $Q_p\text{-nat-pow-nonzero}$ :

**assumes**  $x \in \text{nonzero } Q_p$   
**shows**  $x[\wedge](n::\text{nat}) \in \text{nonzero } Q_p$   
 $\langle \text{proof} \rangle$

**lemma**  $Q_p\text{-nonzero-nat-pow}$ :

**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $n > 0$

**assumes**  $x[\ulcorner](n::nat) \in nonzero\ Q_p$   
**shows**  $x \in nonzero\ Q_p$   
 $\langle proof \rangle$

**lemma** *Qp-int-pow-nonzero*:  
**assumes**  $x \in nonzero\ Q_p$   
**shows**  $x[\ulcorner](n::int) \in nonzero\ Q_p$   
 $\langle proof \rangle$

**lemma** *Qp-nonzero-int-pow*:  
**assumes**  $x \in carrier\ Q_p$   
**assumes**  $n > 0$   
**assumes**  $x[\ulcorner](n::int) \in nonzero\ Q_p$   
**shows**  $x \in nonzero\ Q_p$   
 $\langle proof \rangle$

**lemma** *pow-p-frac-0*:  
**assumes**  $(m::int) \geq n$   
**assumes**  $n \geq 0$   
**shows**  $(frac\ (p[\ulcorner]_{Z_p}\ m)\ (p[\ulcorner]_{Z_p}\ n)) = \iota\ (p[\ulcorner]_{Z_p}\ (m-n))$   
 $\langle proof \rangle$

**lemma** *pow-p-frac*:  
**assumes**  $(m::int) \leq n$   
**assumes**  $m \geq 0$   
**shows**  $(frac\ (p[\ulcorner]_{Z_p}\ m)\ (p[\ulcorner]_{Z_p}\ n)) = frac\ \mathbf{1}_{Z_p}\ (p[\ulcorner]_{Z_p}\ (n-m))$   
 $\langle proof \rangle$

The copy of the prime  $p$  living in  $\mathbb{Q}_p$ :

**abbreviation**  $\mathfrak{p}$  where

$\mathfrak{p} \equiv [p] \cdot_{Q_p} \mathbf{1}$

**lemma**(**in** *domain-frac*) *frac-inc-of-nat*:  
 $frac-inc\ R\ ([ (n::nat) ] \cdot \mathbf{1}) = [n] \cdot_{Frac\ R} \mathbf{1}_{Frac\ R}$   
 $\langle proof \rangle$

**lemma** *inc-of-nat*:  
 $(\iota\ ([ (n::nat) ] \cdot_{Z_p} \mathbf{1}_{Z_p})) = [n] \cdot_{Q_p} \mathbf{1}$   
 $\langle proof \rangle$

**lemma**(**in** *domain-frac*) *frac-inc-of-int*:  
 $frac-inc\ R\ ([ (n::int) ] \cdot \mathbf{1}) = [n] \cdot_{Frac\ R} \mathbf{1}_{Frac\ R}$   
 $\langle proof \rangle$

**lemma** *inc-of-int*:  
 $(\iota\ ([ (n::int) ] \cdot_{Z_p} \mathbf{1}_{Z_p})) = [n] \cdot_{Q_p} \mathbf{1}$   
 $\langle proof \rangle$

**lemma** *p-inc*:

$\mathfrak{p} = \iota \mathfrak{p}$   
 $\langle \text{proof} \rangle$

**lemma** *p-nonzero*:  
 $\mathfrak{p} \in \text{nonzero } Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *p-natpow-inc*:  
**fixes**  $n::\text{nat}$   
**shows**  $\mathfrak{p}[\uparrow]n = \iota (\mathfrak{p} [\uparrow]_{Z_p} n)$   
 $\langle \text{proof} \rangle$

**lemma** *p-intpow-inc*:  
**fixes**  $n::\text{int}$   
**assumes**  $n \geq 0$   
**shows**  $\mathfrak{p}[\uparrow]n = \iota (\mathfrak{p} [\uparrow]_{Z_p} n)$   
 $\langle \text{proof} \rangle$

**lemma** *p-intpow*:  
**fixes**  $n::\text{int}$   
**assumes**  $n < 0$   
**shows**  $\mathfrak{p}[\uparrow]n = (\text{frac } \mathbf{1}_{Z_p} (\mathfrak{p} [\uparrow]_{Z_p} (-n)))$   
 $\langle \text{proof} \rangle$

**lemma** *p-natpow-closed[simp]*:  
**fixes**  $n::\text{nat}$   
**shows**  $(\mathfrak{p}[\uparrow]n) \in (\text{carrier } Q_p)$   
 $(\mathfrak{p}[\uparrow]n) \in (\text{nonzero } Q_p)$   
 $\langle \text{proof} \rangle$

**lemma** *nonzero-int-pow-distrib*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{nonzero } Q_p$   
**shows**  $(a \otimes b) [\uparrow](k::\text{int}) = a[\uparrow]k \otimes b[\uparrow]k$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-subring*:  
 $\text{subring } \mathcal{O}_p \ Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-closed*:  
 $\mathcal{O}_p \subseteq \text{carrier } Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *p-pow-diff*:  
**fixes**  $n::\text{int}$   
**fixes**  $m::\text{int}$   
**assumes**  $n \geq 0$   
**assumes**  $m \geq 0$

**shows**  $\mathfrak{p} \lceil \lceil (n - m) = \text{frac } (\mathfrak{p} \lceil \lceil_{Z_p} n) (\mathfrak{p} \lceil \lceil_{Z_p} m)$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-int-pow-add:*

**fixes**  $n::\text{int}$

**fixes**  $m::\text{int}$

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $a \lceil \lceil (n + m) = (a \lceil \lceil n) \otimes (a \lceil \lceil m)$

$\langle \text{proof} \rangle$

**lemma** *Qp-nat-pow-pow:*

**fixes**  $n::\text{nat}$

**fixes**  $m::\text{nat}$

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $(a \lceil \lceil (n * m)) = ((a \lceil \lceil n) \lceil \lceil m)$

$\langle \text{proof} \rangle$

**lemma** *Qp-p-nat-pow-pow:*

**fixes**  $n::\text{nat}$

**fixes**  $m::\text{nat}$

**shows**  $(\mathfrak{p} \lceil \lceil (n * m)) = ((\mathfrak{p} \lceil \lceil n) \lceil \lceil m)$

$\langle \text{proof} \rangle$

**lemma** *Qp-units-int-pow:*

**fixes**  $n::\text{int}$

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $a \lceil \lceil n = a \lceil \lceil_{\text{units-of } Q_p} n$

$\langle \text{proof} \rangle$

**lemma** *Qp-int-pow-pow:*

**fixes**  $n::\text{int}$

**fixes**  $m::\text{int}$

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $(a \lceil \lceil (n * m)) = ((a \lceil \lceil n) \lceil \lceil m)$

$\langle \text{proof} \rangle$

**lemma** *Qp-p-int-pow-pow:*

**fixes**  $n::\text{int}$

**fixes**  $m::\text{int}$

**shows**  $(\mathfrak{p} \lceil \lceil (n * m)) = ((\mathfrak{p} \lceil \lceil n) \lceil \lceil m)$

$\langle \text{proof} \rangle$

**lemma** *Qp-int-nat-pow-pow:*

**fixes**  $n::\text{int}$

**fixes**  $m::\text{nat}$

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $(a \lceil \lceil (n * m)) = ((a \lceil \lceil n) \lceil \lceil m)$

$\langle \text{proof} \rangle$

**lemma** *Qp-p-int-nat-pow-pow*:

**fixes**  $n::int$

**fixes**  $m::nat$

**shows**  $(p \ [ \ ] \ (n * m)) = ((p \ [ \ ] \ n) \ [ \ ] \ m)$

*<proof>*

**lemma** *Qp-nat-int-pow-pow*:

**fixes**  $n::nat$

**fixes**  $m::int$

**assumes**  $a \in nonzero \ Q_p$

**shows**  $(a \ [ \ ] \ (n * m)) = ((a \ [ \ ] \ n) \ [ \ ] \ m)$

*<proof>*

**lemma** *Qp-p-nat-int-pow-pow*:

**fixes**  $n::nat$

**fixes**  $m::int$

**shows**  $(p \ [ \ ] \ (n * m)) = ((p \ [ \ ] \ n) \ [ \ ] \ m)$

*<proof>*

**lemma** *p-intpow-closed*:

**fixes**  $n::int$

**shows**  $(p \ [ \ ] \ n) \in (carrier \ Q_p)$

$(p \ [ \ ] \ n) \in (nonzero \ Q_p)$

*<proof>*

**lemma** *p-intpow-add*:

**fixes**  $n::int$

**fixes**  $m::int$

**shows**  $p \ [ \ ] \ (n + m) = (p \ [ \ ] \ n) \otimes (p \ [ \ ] \ m)$

*<proof>*

**lemma** *p-intpow-inv*:

**fixes**  $n::int$

**shows**  $(p \ [ \ ] \ n) \otimes (p \ [ \ ] \ -n) = \mathbf{1}$

*<proof>*

**lemma** *p-intpow-inv'*:

**fixes**  $n::int$

**shows**  $(p \ [ \ ] \ -n) \otimes (p \ [ \ ] \ n) = \mathbf{1}$

*<proof>*

**lemma** *p-intpow-inv''*:

**fixes**  $n::int$

**shows**  $(p \ [ \ ] \ -n) = inv_{Q_p} (p \ [ \ ] \ n)$

*<proof>*

**lemma** *p-int-pow-factor-int-pow*:

**assumes**  $a \in nonzero \ Q_p$

**shows**  $(p \ [ \ ] \ (n::int) \otimes a) \ [ \ ] \ (k::int) = p \ [ \ ] \ (n * k) \otimes a \ [ \ ] \ k$

*<proof>*

**lemma** *p-nat-pow-factor-int-pow:*

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $(\mathfrak{p}[\uparrow](n::\text{nat}) \otimes a)[\uparrow](k::\text{int}) = \mathfrak{p}[\uparrow](n*k) \otimes a[\uparrow]k$

*<proof>*

**lemma** *p-pow-factor:*

$\mathfrak{p}[\uparrow]((\text{int } N)*l + k) = (\mathfrak{p}[\uparrow]l)[\uparrow](N::\text{nat}) \otimes \mathfrak{p}[\uparrow]k$

*<proof>*

**lemma** *p-nat-pow-factor-nat-pow:*

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $(\mathfrak{p}[\uparrow](n::\text{nat}) \otimes a)[\uparrow](k::\text{nat}) = \mathfrak{p}[\uparrow](n*k) \otimes a[\uparrow]k$

*<proof>*

**lemma** *p-int-pow-factor-nat-pow:*

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $(\mathfrak{p}[\uparrow](n::\text{int}) \otimes a)[\uparrow](k::\text{nat}) = \mathfrak{p}[\uparrow](n*k) \otimes a[\uparrow]k$

*<proof>*

**lemma**(in *ring*) *r-minus-distr:*

**assumes**  $a \in \text{carrier } R$

**assumes**  $b \in \text{carrier } R$

**assumes**  $c \in \text{carrier } R$

**shows**  $a \otimes b \ominus a \otimes c = a \otimes (b \ominus c)$

*<proof>*

## 8.3 The Valuation on $\mathbb{Q}_p$

### 8.3.1 Extending the Valuation from $\mathbb{Z}_p$ to $\mathbb{Q}_p$

The valuation of a  $p$ -adic number can be defined as the difference of the valuations of an arbitrary choice of numerator and denominator.

**definition** *ord where*

$\text{ord } x = (\text{ord-}Z_p(\text{numer } x)) - (\text{ord-}Z_p(\text{denom } x))$

**definition** *val where*

$\text{val } x = (\text{if } x = \mathbf{0} \text{ then } (\infty::\text{eint}) \text{ else } \text{eint } (\text{ord } x))$

**lemma** *val-ord[simp]:*

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $\text{val } a = \text{ord } a$

*<proof>*

### 8.3.2 Properties of the Valuation

**lemma** *ord-of-frac:*

**assumes**  $a \in \text{nonzero } Z_p$

**assumes**  $b \in \text{nonzero } Z_p$   
**shows**  $\text{ord} (\text{frac } a \ b) = (\text{ord-}Z_p \ a) - (\text{ord-}Z_p \ b)$   
 $\langle \text{proof} \rangle$

**lemma** *val-zero*:  
 $\text{val } \mathbf{0} = \infty \langle \text{proof} \rangle$

**lemma** *ord-one[simp]*:  
 $\text{ord } \mathbf{1} = 0$   
 $\langle \text{proof} \rangle$

**lemma** *val-one[simp]*:  
 $\text{val } (\mathbf{1}) = 0$   
 $\langle \text{proof} \rangle$

**lemma** *val-of-frac*:  
**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**shows**  $\text{val} (\text{frac } a \ b) = (\text{val-}Z_p \ a) - (\text{val-}Z_p \ b)$   
 $\langle \text{proof} \rangle$

**lemma**  *$Z_p$ -division- $Q_p$ -0[simp]*:  
**assumes**  $u \in \text{Units } Z_p$   
**assumes**  $v \in \text{Units } Z_p$   
**shows**  $\text{frac} (u \otimes_{Z_p} (\text{inv}_{Z_p} \ v)) \ \mathbf{1}_{Z_p} = \text{frac } u \ v$   
 $\langle \text{proof} \rangle$

**lemma**  *$Z_p$ -division- $Q_p$ -1*:  
**assumes**  $u \in \text{Units } Z_p$   
**assumes**  $v \in \text{Units } Z_p$   
**obtains**  $w$  **where**  $w \in \text{Units } Z_p$   
 $\iota \ w = \text{frac } u \ v$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-ord-criterion*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $a \neq \mathbf{0}$   
**assumes**  $\text{ord } a \geq 0$   
**shows**  $a \in \mathcal{O}_p$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-val-criterion*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $\text{val } a \geq 0$   
**shows**  $a \in \mathcal{O}_p$   
 $\langle \text{proof} \rangle$

**lemma** *ord-of-inv*:  
**assumes**  $a \in \text{carrier } Q_p$

**assumes**  $a \neq \mathbf{0}$   
**shows**  $\text{ord}(\text{inv}_{Q_p} a) = -(\text{ord } a)$   
*<proof>*

**lemma** *val-of-inv*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $a \neq \mathbf{0}$   
**shows**  $\text{val}(\text{inv}_{Q_p} a) = -(\text{val } a)$   
*<proof>*

$Z_p$  is a valuation ring in  $Q_p$

**lemma** *Z<sub>p</sub>-mem*:  
**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $a \in \mathcal{O}_p \vee (\text{inv}_{Q_p} a \in \mathcal{O}_p)$   
*<proof>*

**lemma** *Q<sub>p</sub>-val-ringI*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $\text{val } a \geq 0$   
**shows**  $a \in \mathcal{O}_p$   
*<proof>*

Criterion for determining when an element in  $Q_p$  is zero

**lemma** *val-nonzero*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $s > \text{val } a$   
**shows**  $a \in \text{nonzero } Q_p$   
*<proof>*

**lemma** *val-ineq*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $\text{val } \mathbf{0} \leq \text{val } a$   
**shows**  $a = \mathbf{0}$   
*<proof>*

**lemma** *ord-minus*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**shows**  $\text{ord } a = \text{ord}(\ominus a)$   
*<proof>*

**lemma** *val-minus*:  
**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $\text{val } a = \text{val}(\ominus a)$   
*<proof>*

The valuation is multiplicative:

**lemma** *ord-mult*:  
**assumes**  $x \in \text{nonzero } Q_p$

**assumes**  $y \in \text{nonzero } Q_p$   
**shows**  $(\text{ord } (x \otimes y)) = (\text{ord } x) + (\text{ord } y)$   
 $\langle \text{proof} \rangle$

**lemma** *val-mult0*:  
**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $y \in \text{nonzero } Q_p$   
**shows**  $(\text{val } (x \otimes y)) = (\text{val } x) + (\text{val } y)$   
 $\langle \text{proof} \rangle$

val is multiplicative everywhere

**lemma** *val-mult*:  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{carrier } Q_p$   
**shows**  $(\text{val } (x \otimes y)) = (\text{val } x) + (\text{val } y)$   
 $\langle \text{proof} \rangle$

val and ord are compatible with inclusion

**lemma** *ord-of-inc*:  
**assumes**  $x \in \text{nonzero } Z_p$   
**shows**  $\text{ord-}Z_p x = \text{ord}(\iota x)$   
 $\langle \text{proof} \rangle$

**lemma** *val-of-inc*:  
**assumes**  $x \in \text{carrier } Z_p$   
**shows**  $\text{val-}Z_p x = \text{val } (\iota x)$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-inc-id*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $\text{ord } a \geq 0$   
**obtains**  $b$  **where**  $b \in \text{nonzero } Z_p$  **and**  $a = \iota b$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-memI*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $\text{val } a \geq 0$   
**shows**  $a \in \mathcal{O}_p$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-memE*:  
**assumes**  $a \in \mathcal{O}_p$   
**shows**  $\text{val } a \geq 0$   $a \in \text{carrier } Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-add-closed*:  
**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $b \in \mathcal{O}_p$   
**shows**  $a \oplus b \in \mathcal{O}_p$

*<proof>*

**lemma** *val-ring-times-closed*:

**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $b \in \mathcal{O}_p$   
**shows**  $a \otimes b \in \mathcal{O}_p$   
*<proof>*

**lemma** *val-ring-ainv-closed*:

**assumes**  $a \in \mathcal{O}_p$   
**shows**  $\ominus a \in \mathcal{O}_p$   
*<proof>*

**lemma** *val-ring-minus-closed*:

**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $b \in \mathcal{O}_p$   
**shows**  $a \ominus b \in \mathcal{O}_p$   
*<proof>*

**lemma** *one-in-val-ring*:

$\mathbf{1} \in \mathcal{O}_p$   
*<proof>*

**lemma** *zero-in-val-ring*:

$\mathbf{0} \in \mathcal{O}_p$   
*<proof>*

**lemma** *ord-p*:

$\text{ord } \mathfrak{p} = 1$   
*<proof>*

**lemma** *ord-p-pow-nat*:

$\text{ord } (\mathfrak{p} [ \wedge ] (n::\text{nat})) = n$   
*<proof>*

**lemma** *ord-p-pow-int*:

$\text{ord } (\mathfrak{p} [ \wedge ] (n::\text{int})) = n$   
*<proof>*

**lemma** *ord-nonneg*:

**assumes**  $x \in \mathcal{O}_p$   
**assumes**  $x \neq \mathbf{0}$   
**shows**  $\text{ord } x \geq 0$   
*<proof>*

**lemma** *val-p*:

$\text{val } \mathfrak{p} = 1$   
*<proof>*

**lemma** *val-p-int-pow*:

*val* ( $\mathfrak{p}[\wedge](k::\text{int})$ ) =  $k$

$\langle \text{proof} \rangle$

**lemma** *val-p-int-pow-neg*:

*val* ( $\mathfrak{p}[\wedge](-k::\text{int})$ ) =  $- \text{eint } k$

$\langle \text{proof} \rangle$

**lemma** *nonzero-nat-pow-ord*:

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $\text{ord } (a [\wedge] (n::\text{nat})) = n * \text{ord } a$

$\langle \text{proof} \rangle$

**lemma** *add-cancel-eint-geq*:

**assumes**  $(\text{eint } a) + x \geq (\text{eint } a) + y$

**shows**  $x \geq y$

$\langle \text{proof} \rangle$

**lemma**(*in padic-fields*) *prod-equal-val-imp-equal-val*:

**assumes**  $a \in \text{nonzero } Q_p$

**assumes**  $b \in \text{carrier } Q_p$

**assumes**  $c \in \text{carrier } Q_p$

**assumes**  $\text{val } (a \otimes b) = \text{val } (a \otimes c)$

**shows**  $\text{val } b = \text{val } c$

$\langle \text{proof} \rangle$

**lemma** *two-times-eint*:

**shows**  $2*(x::\text{eint}) = x + x$

$\langle \text{proof} \rangle$

**lemma** *times-cfs-val-mono*:

**assumes**  $u \in \text{Units } Q_p$

**assumes**  $a \in \text{carrier } Q_p$

**assumes**  $b \in \text{carrier } Q_p$

**assumes**  $\text{val } (u \otimes a) \leq \text{val } (u \otimes b)$

**shows**  $\text{val } a \leq \text{val } b$

$\langle \text{proof} \rangle$

**lemma** *times-cfs-val-mono'*:

**assumes**  $u \in \text{Units } Q_p$

**assumes**  $a \in \text{carrier } Q_p$

**assumes**  $b \in \text{carrier } Q_p$

**assumes**  $\text{val } (u \otimes a) \leq \text{val } (u \otimes b) + \alpha$

**shows**  $\text{val } a \leq \text{val } b + \alpha$

$\langle \text{proof} \rangle$

**lemma** *times-cfs-val-mono''*:

**assumes**  $u \in \text{Units } Q_p$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $\text{val } a \leq \text{val } b + \alpha$   
**shows**  $\text{val } (u \otimes a) \leq \text{val } (u \otimes b) + \alpha$   
*<proof>*

**lemma** *val-ineq-cancel-leq*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $\text{val } (a \otimes b) \leq \text{val } (a \otimes c)$   
**shows**  $\text{val } b \leq \text{val } c$   
*<proof>*

**lemma** *val-ineq-cancel-leq'*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $\text{val } b \leq \text{val } c$   
**shows**  $\text{val } (a \otimes b) \leq \text{val } (a \otimes c)$   
*<proof>*

**lemma** *val-ineq-cancel-le*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $\text{val } (a \otimes b) < \text{val } (a \otimes c)$   
**shows**  $\text{val } b < \text{val } c$   
*<proof>*

**lemma** *val-ineq-cancel-le'*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $\text{val } b < \text{val } c$   
**shows**  $\text{val } (a \otimes b) < \text{val } (a \otimes c)$   
*<proof>*

**lemma** *finite-val-imp-nonzero*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $\text{val } a \neq \infty$   
**shows**  $a \in \text{nonzero } Q_p$   
*<proof>*

**lemma** *val-ineq-cancel-leq''*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$

**assumes**  $val\ b \leq val\ c + eint\ N$   
**shows**  $val\ (a \otimes b) \leq val\ (a \otimes c) + eint\ N$   
 ⟨*proof*⟩

### 8.3.3 The Ultrametric Inequality on $\mathbb{Q}_p$

**lemma** *ord-ultrametric*:  
**assumes**  $x \in nonzero\ \mathbb{Q}_p$   
**assumes**  $y \in nonzero\ \mathbb{Q}_p$   
**assumes**  $x \oplus y \in nonzero\ \mathbb{Q}_p$   
**shows**  $ord\ (x \oplus y) \geq min\ (ord\ x)\ (ord\ y)$   
 ⟨*proof*⟩

**lemma** *ord-ultrametric'*:  
**assumes**  $x \in nonzero\ \mathbb{Q}_p$   
**assumes**  $y \in nonzero\ \mathbb{Q}_p$   
**assumes**  $x \ominus_{\mathbb{Q}_p} y \in nonzero\ \mathbb{Q}_p$   
**shows**  $ord\ (x \ominus_{\mathbb{Q}_p} y) \geq min\ (ord\ x)\ (ord\ y)$   
 ⟨*proof*⟩

**lemma** *val-ultrametric0*:  
**assumes**  $x \in nonzero\ \mathbb{Q}_p$   
**assumes**  $y \in nonzero\ \mathbb{Q}_p$   
**assumes**  $x \oplus y \in nonzero\ \mathbb{Q}_p$   
**shows**  $min\ (val\ x)\ (val\ y) \leq val\ (x \oplus y)$   
 ⟨*proof*⟩

**lemma** *val-ultrametric*:  
**assumes**  $x \in carrier\ \mathbb{Q}_p$   
**assumes**  $y \in carrier\ \mathbb{Q}_p$   
**shows**  $min\ (val\ x)\ (val\ y) \leq val\ (x \oplus y)$   
 ⟨*proof*⟩

**lemma** *val-ultrametric'*:  
**assumes**  $x \in carrier\ \mathbb{Q}_p$   
**assumes**  $y \in carrier\ \mathbb{Q}_p$   
**shows**  $min\ (val\ x)\ (val\ y) \leq val\ (x \ominus y)$   
 ⟨*proof*⟩

**lemma** *diff-ord-nonzero*:  
**assumes**  $x \in nonzero\ \mathbb{Q}_p$   
**assumes**  $y \in nonzero\ \mathbb{Q}_p$   
**assumes**  $ord\ x \neq ord\ y$   
**shows**  $x \oplus y \in nonzero\ \mathbb{Q}_p$   
 ⟨*proof*⟩

**lemma** *ord-ultrametric-noteq*:  
**assumes**  $x \in nonzero\ \mathbb{Q}_p$   
**assumes**  $y \in nonzero\ \mathbb{Q}_p$

**assumes**  $\text{ord } x > \text{ord } y$   
**shows**  $\text{ord } (x \oplus y) = (\text{ord } y)$   
*<proof>*

**lemma** *ord-ultrametric-noteq'*:  
**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $y \in \text{nonzero } Q_p$   
**assumes**  $\text{ord } x > \text{ord } y$   
**shows**  $\text{ord } (x \ominus y) = (\text{ord } y)$   
*<proof>*

**lemma** *ord-ultrametric-noteq''*:  
**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $y \in \text{nonzero } Q_p$   
**assumes**  $\text{ord } y > \text{ord } x$   
**shows**  $\text{ord } (x \ominus y) = (\text{ord } x)$   
*<proof>*

**lemma** *val-ultrametric-noteq*:  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{carrier } Q_p$   
**assumes**  $\text{val } x > \text{val } y$   
**shows**  $\text{val } (x \oplus y) = \text{val } y$   
*<proof>*

**lemma** *val-ultrametric-noteq'*:  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{carrier } Q_p$   
**assumes**  $\text{val } x > \text{val } y$   
**shows**  $\text{val } (x \ominus y) = \text{val } y$   
*<proof>*

**lemma** *ultrametric-equal-eq*:  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{carrier } Q_p$   
**assumes**  $\text{val } (y \ominus x) > \text{val } x$   
**shows**  $\text{val } x = \text{val } y$   
*<proof>*

**lemma** *ultrametric-equal-eq'*:  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{carrier } Q_p$   
**assumes**  $\text{val } (x \ominus y) > \text{val } x$   
**shows**  $\text{val } x = \text{val } y$   
*<proof>*

**lemma** *val-ultrametric-noteq''*:  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{carrier } Q_p$

**assumes**  $val\ x > val\ y$   
**shows**  $val\ (y \ominus x) = val\ y$   
 $\langle proof \rangle$

Ultrametric over finite sums:

**lemma** *Min-mono*:  
**assumes**  $finite\ A$   
**assumes**  $A \neq \{\}$   
**assumes**  $\bigwedge a. a \in A \implies f\ a \leq a$   
**shows**  $Min\ (f'A) \leq Min\ A$   
 $\langle proof \rangle$

**lemma** *Min-mono'*:  
**assumes**  $finite\ A$   
**assumes**  $\bigwedge (a::'a). a \in A \implies (f::'a \Rightarrow eint)\ a \leq g\ a$   
**shows**  $Min\ (f'A) \leq Min\ (g\ 'A)$   
 $\langle proof \rangle$

**lemma** *eint-ord-trans*:  
**assumes**  $(a::eint) \leq b$   
**assumes**  $b \leq c$   
**shows**  $a \leq c$   
 $\langle proof \rangle$

**lemma** *eint-Min-geq*:  
**assumes**  $finite\ (A::eint\ set)$   
**assumes**  $\bigwedge x. x \in A \implies x \geq c$   
**assumes**  $A \neq \{\}$   
**shows**  $Min\ A \geq c$   
 $\langle proof \rangle$

**lemma** *eint-Min-gr*:  
**assumes**  $finite\ (A::eint\ set)$   
**assumes**  $\bigwedge x. x \in A \implies x > c$   
**assumes**  $A \neq \{\}$   
**shows**  $Min\ A > c$   
 $\langle proof \rangle$

**lemma** *finsum-val-ultrametric*:  
**assumes**  $g \in A \rightarrow carrier\ Q_p$   
**assumes**  $finite\ A$   
**assumes**  $A \neq \{\}$   
**shows**  $val\ (finsum\ Q_p\ g\ A) \geq Min\ (val\ '(g'A))$   
 $\langle proof \rangle$

**lemma** (in *padic-fields*) *finsum-val-ultrametric'*:  
**assumes**  $g \in A \rightarrow carrier\ Q_p$   
**assumes**  $finite\ A$   
**assumes**  $\bigwedge i. i \in A \implies val\ (g\ i) \geq c$

**shows**  $\text{val} (\text{finsum } Q_p g A) \geq c$   
 ⟨proof⟩

**lemma** (in *padic-fields*) *finsum-val-ultrametric''*:

**assumes**  $g \in A \rightarrow \text{carrier } Q_p$   
**assumes** *finite*  $A$   
**assumes**  $\bigwedge i. i \in A \implies \text{val} (g i) > c$   
**assumes**  $c < \infty$   
**shows**  $\text{val} (\text{finsum } Q_p g A) > c$   
 ⟨proof⟩

**lemma** *Qp-diff-diff*:

**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $d \in \text{carrier } Q_p$   
**shows**  $(x \ominus c) \ominus (d \ominus c) = x \ominus d$   
 ⟨proof⟩

This variant of the ultrametric identity formalizes the common saying that "all triangles in  $Q_p$  are isosceles":

**lemma** *Qp-isosceles*:

**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $d \in \text{carrier } Q_p$   
**assumes**  $\text{val} (x \ominus c) \geq v$   
**assumes**  $\text{val} (d \ominus c) \geq v$   
**shows**  $\text{val} (x \ominus d) \geq v$   
 ⟨proof⟩

More variants on the ultrametric inequality

**lemma** *MinE*:

**assumes** *finite* ( $A::\text{eint set}$ )  
**assumes**  $a = \text{Min } A$   
**assumes**  $b \in A$   
**shows**  $a \leq b$   
 ⟨proof⟩

**lemma** *MinE'*:

**assumes** *finite* ( $A::\text{eint set}$ )  
**assumes**  $a = \text{Min } A$   
**assumes**  $b \in A - \{a\}$   
**shows**  $a < b$   
 ⟨proof⟩

**lemma** *MinE''*:

**assumes** *finite*  $A$   
**assumes**  $f \in A \rightarrow (\text{UNIV} :: \text{eint set})$   
**assumes**  $a = \text{Min} (f ' A)$   
**assumes**  $b \in A$

**shows**  $a \leq f b$   
 ⟨proof⟩

**lemma** *finsum-val-ultrametric-diff*:

**assumes**  $g \in A \rightarrow \text{carrier } Q_p$   
**assumes** *finite*  $A$   
**assumes**  $A \neq \{\}$   
**assumes**  $\bigwedge a b. a \in A \implies b \in A \implies a \neq b \implies \text{val } (g a) \neq \text{val } (g b)$   
**shows**  $\text{val } (\text{finsum } Q_p g A) = \text{Min } (\text{val } 'g'A)$   
 ⟨proof⟩

**lemma** *finsum-val-ultrametric-diff'*:

**assumes**  $g \in A \rightarrow \text{carrier } Q_p$   
**assumes** *finite*  $A$   
**assumes**  $A \neq \{\}$   
**assumes**  $\bigwedge a b. a \in A \implies b \in A \implies a \neq b \implies \text{val } (g a) \neq \text{val } (g b)$   
**shows**  $\text{val } (\text{finsum } Q_p g A) = (\text{MIN } a \in A. (\text{val } (g a)))$   
 ⟨proof⟩

## 8.4 Constructing the Angular Component Maps on $\mathbb{Q}_p$

### 8.4.1 Unreduced Angular Component Map

While one can compute the residue of a  $p$ -adic integer mod  $p^n$ , this operation does not generalize to the  $p$ -adic field unless we restrict our attention to the valuation ring. However, we can still define the angular component maps on the field  $\mathbb{Q}_p$ , which allows us to take a sort of residue for any element  $x \in \mathbb{Q}_p$ . Given a nonzero element  $x \in \mathbb{Q}_p^\times$ , we can normalize it to obtain  $p^{-\text{ord}(x)}x$  which has of valuation zero, and then computes its residue (viewed as an element of  $\mathbb{Z}_p$ ). The resulting map agrees with the standard residue map on elements of  $\mathbb{Q}_p$  of valuation zero, but not on terms of positive or negative valuation. For example, the element  $p^2$  has an order 1 residue of 0, but its order 1 angular component is 1. In the formalism below, we will use the term "angular\_component" to refer to the unreduced normalization map  $x \mapsto p^{-\text{ord}(x)}x$ , and use the notation "ac n" to refer to the angular component which has been reduced mod  $p^n$ . This is line with the terminology used in [1].

**definition** *angular-component where*

*angular-component*  $a = (\text{ac-Zp } (\text{numer } a)) \otimes_{Z_p} (\text{inv}_{Z_p} \text{ ac-Zp } (\text{denom } a))$

**lemma** *ac-fract*:

**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $a \in \text{nonzero } Z_p$   
**assumes**  $b \in \text{nonzero } Z_p$   
**assumes**  $c = \text{frac } a b$   
**shows** *angular-component*  $c = (\text{ac-Zp } a) \otimes_{Z_p} \text{inv}_{Z_p} (\text{ac-Zp } b)$   
 ⟨proof⟩

**lemma** *angular-component-closed*:

**assumes**  $a \in \text{nonzero } Q_p$   
**shows** *angular-component*  $a \in \text{carrier } Z_p$   
*<proof>*

**lemma** *angular-component-unit*:

**assumes**  $a \in \text{nonzero } Q_p$   
**shows** *angular-component*  $a \in \text{Units } Z_p$   
*<proof>*

**lemma** *angular-component-factors-x*:

**assumes**  $x \in \text{nonzero } Q_p$   
**shows**  $x = (\mathfrak{p}[\uparrow](\text{ord } x)) \otimes \iota (\text{angular-component } x)$   
*<proof>*

**lemma** *angular-component-mult*:

**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $y \in \text{nonzero } Q_p$   
**shows** *angular-component*  $(x \otimes y) = (\text{angular-component } x) \otimes_{Z_p} (\text{angular-component } y)$   
*<proof>*

**lemma** *angular-component-inv*:

**assumes**  $x \in \text{nonzero } Q_p$   
**shows** *angular-component*  $(\text{inv}_{Q_p} x) = \text{inv}_{Z_p} (\text{angular-component } x)$   
*<proof>*

**lemma** *angular-component-one*:

*angular-component*  $\mathbf{1} = \mathbf{1}_{Z_p}$   
*<proof>*

**lemma** *angular-component-ord-zero*:

**assumes**  $\text{ord } x = 0$   
**assumes**  $x \in \text{nonzero } Q_p$   
**shows**  $\iota (\text{angular-component } x) = x$   
*<proof>*

**lemma** *angular-component-of-inclusion*:

**assumes**  $x \in \text{nonzero } Z_p$   
**assumes**  $y = \iota x$   
**shows** *angular-component*  $y = \text{ac-}Z_p x$   
*<proof>*

**lemma** *res-uminus*:

**assumes**  $k > 0$   
**assumes**  $f \in \text{carrier } Z_p$   
**assumes**  $c \in \text{carrier } (Z_p\text{-res-ring } k)$   
**assumes**  $c = \ominus_{Z_p\text{-res-ring } k} (f k)$

**shows**  $c = ((\ominus_{Z_p} f) k)$   
 $\langle proof \rangle$

**lemma** *ord-fract*:  
**assumes**  $a \in nonzero\ Q_p$   
**assumes**  $b \in nonzero\ Q_p$   
**shows**  $ord\ (a \div b) = ord\ a - ord\ b$   
 $\langle proof \rangle$

**lemma** *val-fract*:  
**assumes**  $a \in carrier\ Q_p$   
**assumes**  $b \in nonzero\ Q_p$   
**shows**  $val\ (a \div b) = val\ a - val\ b$   
 $\langle proof \rangle$

**lemma** *zero-fract*:  
**assumes**  $a \in nonzero\ Q_p$   
**shows**  $0 \div a = 0$   
 $\langle proof \rangle$

**lemma** *fract-closed*:  
**assumes**  $a \in carrier\ Q_p$   
**assumes**  $b \in nonzero\ Q_p$   
**shows**  $(a \div b) \in carrier\ Q_p$   
 $\langle proof \rangle$

**lemma** *val-of-power*:  
**assumes**  $a \in nonzero\ Q_p$   
**shows**  $val\ (a[\wedge](n::nat)) = n*(val\ a)$   
 $\langle proof \rangle$

**lemma** *val-zero-imp-val-pow-zero*:  
**assumes**  $a \in carrier\ Q_p$   
**assumes**  $val\ a = 0$   
**shows**  $val\ (a[\wedge](n::nat)) = 0$   
 $\langle proof \rangle$

val and ord of powers of p

**lemma** *val-p-nat-pow*:  
 $val\ (p[\wedge](k::nat)) = eint\ k$   
 $\langle proof \rangle$

**lemma** *ord-p-int-pow*:  
 $ord\ (p[\wedge](k::int)) = k$   
 $\langle proof \rangle$

**lemma** *ord-p-nat-pow*:  
 $ord\ (p[\wedge](k::nat)) = k$   
 $\langle proof \rangle$

**lemma** *val-nonzero-frac*:

**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{nonzero } Q_p$   
**assumes**  $\text{val } (a \div b) = c$   
**shows**  $\text{val } a = \text{val } b + c$

*<proof>*

**lemma** *val-nonzero-frac'*:

**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{nonzero } Q_p$   
**assumes**  $\text{val } (a \div b) = 0$   
**shows**  $\text{val } a = \text{val } b$

*<proof>*

**lemma** *equal-val-imp-equal-ord*:

**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $\text{val } a = \text{val } b$   
**shows**  $\text{ord } a = \text{ord } b \text{ } b \in \text{nonzero } Q_p$

*<proof>*

**lemma** *int-pow-ord*:

**assumes**  $a \in \text{nonzero } Q_p$   
**shows**  $\text{ord } (a[\wedge](i::\text{int})) = i * (\text{ord } a)$

*<proof>*

**lemma** *int-pow-val*:

**assumes**  $a \in \text{nonzero } Q_p$   
**shows**  $\text{val } (a[\wedge](i::\text{int})) = i * (\text{val } a)$

*<proof>*

**lemma** *neg-int-pow-val*:

**assumes**  $a \in \text{nonzero } Q_p$   
**shows**  $\text{val } (a[\wedge]-(i::\text{int})) = - (\text{val } (a[\wedge]i))$

*<proof>*

**lemma** *int-pow-sum-val*:

**assumes**  $a \in \text{nonzero } Q_p$   
**shows**  $\text{val } (a[\wedge]((i::\text{int}) + j)) = (\text{val } (a[\wedge]i)) + \text{val } (a[\wedge]j)$

*<proof>*

**lemma** *int-pow-diff-val*:

**assumes**  $a \in \text{nonzero } Q_p$   
**shows**  $\text{val } (a[\wedge]((i::\text{int}) - j)) = (\text{val } (a[\wedge]i)) - \text{val } (a[\wedge]j)$

*<proof>*

**lemma** *nat-add-pow-mult-assoc*:

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $[(n::nat)].a = [n].\mathbf{1} \otimes a$   
 $\langle proof \rangle$

**lemma**(in *padic-integers*) *equal-res-imp-equal-ord-Zp*:  
**assumes**  $N > 0$   
**assumes**  $a \in carrier\ Zp$   
**assumes**  $b \in carrier\ Zp$   
**assumes**  $a\ N = b\ N$   
**assumes**  $a\ N \neq 0$   
**shows**  $ord\text{-}Zp\ a = ord\text{-}Zp\ b$   
 $\langle proof \rangle$

**lemma**(in *padic-integers*) *equal-res-mod*:  
**assumes**  $N > k$   
**assumes**  $a \in carrier\ Zp$   
**assumes**  $b \in carrier\ Zp$   
**assumes**  $a\ N = b\ N$   
**shows**  $a\ k = b\ k$   
 $\langle proof \rangle$

**lemma** *Qp-char-0*:  
**assumes**  $(n::nat) \neq 0$   
**shows**  $[n].\mathbf{1} \neq \mathbf{0}$   
 $\langle proof \rangle$

**lemma** *Qp-char-0-int*:  
**assumes**  $(n::int) \neq 0$   
**shows**  $[n].\mathbf{1} \neq \mathbf{0}$   
 $\langle proof \rangle$

**lemma** *add-int-pow-inject*:  
**assumes**  $[(k::int)].\mathbf{1} = [(j::int)].\mathbf{1}$   
**shows**  $k = j$   
 $\langle proof \rangle$

**lemma** *val-ord-nat-inc*:  
**assumes**  $(n::nat) > 0$   
**shows**  $ord\ ([n].\mathbf{1}) = val([n].\mathbf{1})$   
 $\langle proof \rangle$

**lemma** *val-ord-int-inc*:  
**assumes**  $(n::int) \neq 0$   
**shows**  $ord\ ([n].\mathbf{1}) = val([n].\mathbf{1})$   
 $\langle proof \rangle$

## 8.4.2 Reduced Angular Component Maps

**definition**  $ac :: nat \Rightarrow padic\text{-}number \Rightarrow int$  **where**  
 $ac\ n\ x = (if\ x = \mathbf{0}\ then\ 0\ else\ (angular\text{-}component\ x)\ n)$

**lemma** *ac-in-res-ring*:

**assumes**  $x \in \text{nonzero } Q_p$   
**shows**  $ac\ n\ x \in \text{carrier } (Zp\text{-res-ring } n)$   
 $\langle \text{proof} \rangle$

**lemma** *ac-in-res-ring'[simp]*:

**assumes**  $x \in \text{carrier } Q_p$   
**shows**  $ac\ n\ x \in \text{carrier } (Zp\text{-res-ring } n)$   
 $\langle \text{proof} \rangle$

**lemma** *ac-mult'*:

**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $y \in \text{nonzero } Q_p$   
**shows**  $ac\ n\ (x \otimes y) = (ac\ n\ x) \otimes_{Zp\text{-res-ring } n} (ac\ n\ y)$   
 $\langle \text{proof} \rangle$

**lemma** *ac-mult*:

**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{carrier } Q_p$   
**shows**  $ac\ n\ (x \otimes y) = (ac\ n\ x) \otimes_{Zp\text{-res-ring } n} (ac\ n\ y)$   
 $\langle \text{proof} \rangle$

**lemma** *ac-one[simp]*:

**assumes**  $n \geq 1$   
**shows**  $ac\ n\ \mathbf{1} = 1$   
 $\langle \text{proof} \rangle$

**lemma** *ac-one'*:

**assumes**  $n > 0$   
**shows**  $ac\ n\ \mathbf{1} = \mathbf{1}_{Zp\text{-res-ring } n}$   
 $\langle \text{proof} \rangle$

**lemma** *ac-units*:

**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $n > 0$   
**shows**  $ac\ n\ x \in \text{Units } (Zp\text{-res-ring } n)$   
 $\langle \text{proof} \rangle$

**lemma** *ac-inv*:

**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $n > 0$   
**shows**  $ac\ n\ (\text{inv } x) = \text{inv}_{Zp\text{-res-ring } n} (ac\ n\ x)$   
 $\langle \text{proof} \rangle$

**lemma** *ac-inv'*:

**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $n > 0$   
**shows**  $ac\ n\ (\text{inv } x) \otimes_{Zp\text{-res-ring } n} (ac\ n\ x) = \mathbf{1}_{Zp\text{-res-ring } n}$

*<proof>*

**lemma** *ac-inv''*:

**assumes**  $x \in \text{nonzero } Q_p$

**assumes**  $n > 0$

**shows**  $(ac\ n\ x) \otimes_{Zp\text{-res-ring } n} ac\ n\ (inv\ x) = \mathbf{1}_{Zp\text{-res-ring } n}$

*<proof>*

**lemma** *ac-inv'''*:

**assumes**  $x \in \text{nonzero } Q_p$

**assumes**  $n > 0$

**shows**  $(ac\ n\ x) \otimes_{Zp\text{-res-ring } n} ac\ n\ (inv\ x) = 1$

$ac\ n\ (inv\ x) \otimes_{Zp\text{-res-ring } n} (ac\ n\ x) = 1$

*<proof>*

**lemma** *ac-val*:

**assumes**  $a \in \text{nonzero } Q_p$

**assumes**  $b \in \text{nonzero } Q_p$

**assumes**  $val\ a = val\ b$

**assumes**  $val\ (a \ominus b) \geq val\ a + n$

**shows**  $ac\ n\ a = ac\ n\ b$

*<proof>*

**lemma** *angular-component-nat-pow*:

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $angular\text{-component}\ (a\ [\wedge]\ (k::nat)) = (angular\text{-component}\ a)\ [\wedge]_{Z_p}\ k$

*<proof>*

**lemma** *angular-component-int-pow*:

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $angular\text{-component}\ (a\ [\wedge]\ (k::int)) = (angular\text{-component}\ a)\ [\wedge]_{Z_p}\ k$

*<proof>*

**lemma** *ac-nat-pow*:

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $ac\ n\ (a\ [\wedge]\ (k::nat)) = (ac\ n\ a)^k \text{ mod } (p^n)$

*<proof>*

**lemma** *ac-nat-pow'*:

**assumes**  $a \in \text{nonzero } Q_p$

**assumes**  $n \neq 0$

**shows**  $ac\ n\ (a\ [\wedge]\ (k::nat)) = (ac\ n\ a)[\wedge]_{Zp\text{-res-ring } n}\ k$

*<proof>*

**lemma** *ac-int-pow*:

**assumes**  $a \in \text{nonzero } Q_p$

**assumes**  $n > 0$

**shows**  $ac\ n\ (a\ [\wedge]\ (k::int)) = (ac\ n\ a)[\wedge]_{Zp\text{-res-ring } n}\ k$

*<proof>*

**lemma** *angular-component-p:*

*angular-component*  $\mathfrak{p} = \mathbf{1}_{Z_p}$

*<proof>*

**lemma** *angular-component-p-nat-pow:*

*angular-component*  $(\mathfrak{p} [\uparrow] (n::nat)) = \mathbf{1}_{Z_p}$

*<proof>*

**lemma** *angular-component-p-int-pow:*

*angular-component*  $(\mathfrak{p} [\uparrow] (n::int)) = \mathbf{1}_{Z_p}$

*<proof>*

**lemma** *ac-p-nat-pow:*

**assumes**  $k > 0$

**shows** *ac*  $k$   $(\mathfrak{p} [\uparrow] (n::nat)) = 1$

*<proof>*

**lemma** *ac-p:*

**assumes**  $k > 0$

**shows** *ac*  $k$   $\mathfrak{p} = 1$

*<proof>*

**lemma** *ac-p-int-pow:*

**assumes**  $k > 0$

**shows** *ac*  $k$   $(\mathfrak{p} [\uparrow] (n::int)) = 1$

*<proof>*

**lemma** *angular-component-p-nat-pow-factor:*

**assumes**  $a \in \text{nonzero } Q_p$

**shows** *angular-component*  $((\mathfrak{p} [\uparrow] (n::nat)) \otimes a) = \text{angular-component } a$

*<proof>*

**lemma** *ac-p-nat-pow-factor:*

**assumes**  $m > 0$

**assumes**  $a \in \text{nonzero } Q_p$

**shows** *ac*  $m$   $((\mathfrak{p} [\uparrow] (n::nat)) \otimes a) = \text{ac } m \ a$

*<proof>*

**lemma** *angular-component-p-nat-pow-factor-right:*

**assumes**  $a \in \text{nonzero } Q_p$

**shows** *angular-component*  $(a \otimes (\mathfrak{p} [\uparrow] (n::nat))) = \text{angular-component } a$

*<proof>*

**lemma** *ac-p-nat-pow-factor-right:*

**assumes**  $m > 0$

**assumes**  $a \in \text{carrier } Q_p$

**shows** *ac*  $m$   $(a \otimes (\mathfrak{p} [\uparrow] (n::nat))) = \text{ac } m \ a$

*<proof>*

**lemma** *angular-component-p-int-pow-factor:*

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $\text{angular-component } ((\mathfrak{p} \uparrow (n::\text{int})) \otimes a) = \text{angular-component } a$

*<proof>*

**lemma** *ac-p-int-pow-factor:*

**assumes**  $a \in \text{nonzero } Q_p$

**shows**  $\text{ac } m ((\mathfrak{p} \uparrow (n::\text{int})) \otimes a) = \text{ac } m a$

*<proof>*

**lemma** *angular-component-p-int-pow-factor-right:*

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $\text{angular-component } (a \otimes (\mathfrak{p} \uparrow (n::\text{int}))) = \text{angular-component } a$

*<proof>*

**lemma** *ac-p-int-pow-factor-right:*

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $\text{ac } m (a \otimes (\mathfrak{p} \uparrow (n::\text{int}))) = \text{ac } m a$

*<proof>*

## 8.5 An Inverse for the inclusion map $\iota$

**definition** *to-Zp where*

$\text{to-Zp } a = (\text{if } (a \in \mathcal{O}_p) \text{ then } (\text{SOME } x. x \in \text{carrier } Z_p \wedge \iota x = a) \text{ else } \mathbf{0}_{Z_p})$

**lemma** *to-Zp-closed:*

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $\text{to-Zp } a \in \text{carrier } Z_p$

*<proof>*

**lemma** *to-Zp-inc:*

**assumes**  $a \in \mathcal{O}_p$

**shows**  $\iota (\text{to-Zp } a) = a$

*<proof>*

**lemma** *inc-to-Zp:*

**assumes**  $b \in \text{carrier } Z_p$

**shows**  $\text{to-Zp } (\iota b) = b$

*<proof>*

**lemma** *to-Zp-add:*

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \mathcal{O}_p$

**shows**  $\text{to-Zp } (a \oplus b) = \text{to-Zp } a \oplus_{Z_p} (\text{to-Zp } b)$

*<proof>*

**lemma** *to-Zp-mult:*

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \mathcal{O}_p$   
**shows**  $to\text{-}Zp (a \otimes b) = to\text{-}Zp a \otimes_{Z_p} (to\text{-}Zp b)$   
 $\langle proof \rangle$

**lemma** *to-Zp-minus*:  
**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $b \in \mathcal{O}_p$   
**shows**  $to\text{-}Zp (a \ominus b) = to\text{-}Zp a \ominus_{Z_p} (to\text{-}Zp b)$   
 $\langle proof \rangle$

**lemma** *to-Zp-one*:  
**shows**  $to\text{-}Zp \mathbf{1} = \mathbf{1}_{Z_p}$   
 $\langle proof \rangle$

**lemma** *to-Zp-zero*:  
**shows**  $to\text{-}Zp \mathbf{0} = \mathbf{0}_{Z_p}$   
 $\langle proof \rangle$

**lemma** *to-Zp-ominus*:  
**assumes**  $a \in \mathcal{O}_p$   
**shows**  $to\text{-}Zp (\ominus a) = \ominus_{Z_p} (to\text{-}Zp a)$   
 $\langle proof \rangle$

**lemma** *to-Zp-val*:  
**assumes**  $a \in \mathcal{O}_p$   
**shows**  $val\text{-}Zp (to\text{-}Zp a) = val a$   
 $\langle proof \rangle$

**lemma** *val-of-nat-inc*:  
 $val ((k::nat) \cdot \mathbf{1}) \geq 0$   
 $\langle proof \rangle$

**lemma** *val-of-int-inc*:  
 $val ((k::int) \cdot \mathbf{1}) \geq 0$   
 $\langle proof \rangle$

**lemma** *to-Zp-nat-inc*:  
 $to\text{-}Zp ((a::nat) \cdot \mathbf{1}) = [a] \cdot_{Z_p} \mathbf{1}_{Z_p}$   
 $\langle proof \rangle$

**lemma** *to-Zp-int-neg*:  
 $to\text{-}Zp ((-int (a::nat)) \cdot \mathbf{1}) = \ominus_{Z_p} ([int a] \cdot_{Z_p} \mathbf{1}_{Z_p})$   
 $\langle proof \rangle$

**lemma**(*in ring*) *int-add-pow*:  
 $[int n] \cdot \mathbf{1} = [n] \cdot \mathbf{1}$   
 $\langle proof \rangle$

**lemma** *int-add-pow*:

$$[int\ n] \cdot \mathbf{1} = [n] \cdot \mathbf{1}$$

*<proof>*

**lemma** *Zp-int-add-pow*:

$$[int\ n] \cdot_{Z_p} \mathbf{1}_{Z_p} = [n] \cdot_{Z_p} \mathbf{1}_{Z_p}$$

*<proof>*

**lemma** *to-Zp-int-inc*:

$$to\text{-}Zp\ ([a::int]) \cdot \mathbf{1} = ([a] \cdot_{Z_p} \mathbf{1}_{Z_p})$$

*<proof>*

**lemma** *to-Zp-nat-add-pow*:

**assumes**  $a \in \mathcal{O}_p$

**shows**  $to\text{-}Zp\ (([n::nat]) \cdot a) = [n] \cdot_{Z_p} to\text{-}Zp\ a$

*<proof>*

**lemma** *val-ring-res*:

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \mathcal{O}_p$

**shows**  $to\text{-}Zp\ (a \ominus b)\ N = to\text{-}Zp\ a\ N \ominus_{Zp\text{-}res\text{-}ring\ N} to\text{-}Zp\ b\ N$

*<proof>*

**lemma** *res-diff-in-val-ring-imp-in-val-ring*:

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in carrier\ Q_p$

**assumes**  $a \ominus b \in \mathcal{O}_p$

**shows**  $b \in \mathcal{O}_p$

*<proof>*

**lemma**(*in padic-fields*) *equal-res-imp-res-diff-zero*:

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \mathcal{O}_p$

**assumes**  $to\text{-}Zp\ a\ N = to\text{-}Zp\ b\ N$

**shows**  $to\text{-}Zp\ (a \ominus b)\ N = 0$

*<proof>*

**lemma**(*in padic-fields*) *equal-res-imp-val-diff-bound*:

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \mathcal{O}_p$

**assumes**  $to\text{-}Zp\ a\ N = to\text{-}Zp\ b\ N$

**shows**  $val\ (a \ominus b) \geq N$

*<proof>*

**lemma**(*in padic-fields*) *equal-res-equal-val*:

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \mathcal{O}_p$

**assumes**  $val\ a < N$

**assumes**  $to\text{-}Zp\ a\ N = to\text{-}Zp\ b\ N$

```

    shows val a = val b
  <proof>

lemma(in padic-fields) val-ring-equal-res-imp-equal-val:
  assumes a ∈ Op
  assumes b ∈ Op
  assumes val a < eint N
  assumes val b < eint N
  assumes to-Zp a N = to-Zp b N
  shows val a = val b
<proof>

end
end
theory Padic-Field-Polynomials
  imports Padic-Fields

```

```
begin
```

## 9 $p$ -adic Univariate Polynomials and Hensel's Lemma

```
type-synonym padic-field-poly = nat ⇒ padic-number
```

```
type-synonym padic-field-fun = padic-number ⇒ padic-number
```

### 9.1 Gauss Norms of Polynomials

The Gauss norm of a polynomial is defined to be the minimum valuation of a coefficient of that polynomial. This induces a valuation on the ring of polynomials, and in particular it satisfies the ultrametric inequality. In addition, the Gauss norm of a polynomial  $f(x)$  gives a lower bound for the value  $\text{val}(f(a))$  in terms of  $\text{val}(a)$ , for a point  $a \in \mathbb{Q}_p$ . We introduce Gauss norms here as a useful tool for stating and proving Hensel's Lemma for the field  $\mathbb{Q}_p$ . We are abusing terminology slightly in calling this the Gauss norm, rather than the Gauss valuation, but this is just to conform with our decision to work exclusively with the  $p$ -adic valuation and not discuss the equivalent real-valued  $p$ -adic norm. For a detailed treatment of Gauss norms one can see, for example [2].

```
context padic-fields
begin
```

```
no-notation Zp.to-fun (infixl<·> 70)
```

```
abbreviation(input) Qp-x where
  Qp-x ≡ UP Qp
```

```
definition gauss-norm where
```

$gauss\text{-}norm\ g = Min (val\ 'g\ '\{..degree\ g\})$

**lemma** *gauss-normE*:

**assumes**  $g \in carrier\ Q_p\text{-}x$   
**shows**  $gauss\text{-}norm\ g \leq val\ (g\ k)$   
*<proof>*

**lemma** *gauss-norm-geqI*:

**assumes**  $g \in carrier\ (UP\ Q_p)$   
**assumes**  $\bigwedge n. val\ (g\ n) \geq \alpha$   
**shows**  $gauss\text{-}norm\ g \geq \alpha$   
*<proof>*

**lemma** *gauss-norm-eqI*:

**assumes**  $g \in carrier\ (UP\ Q_p)$   
**assumes**  $\bigwedge n. val\ (g\ n) \geq \alpha$   
**assumes**  $val\ (g\ i) = \alpha$   
**shows**  $gauss\text{-}norm\ g = \alpha$   
*<proof>*

**lemma** *nonzero-poly-nonzero-coeff*:

**assumes**  $g \in carrier\ Q_p\text{-}x$   
**assumes**  $g \neq \mathbf{0}_{Q_p\text{-}x}$   
**shows**  $\exists k. k \leq degree\ g \wedge g\ k \neq \mathbf{0}_{Q_p}$   
*<proof>*

**lemma** *gauss-norm-prop*:

**assumes**  $g \in carrier\ Q_p\text{-}x$   
**assumes**  $g \neq \mathbf{0}_{Q_p\text{-}x}$   
**shows**  $gauss\text{-}norm\ g \neq \infty$   
*<proof>*

**lemma** *gauss-norm-coeff-norm*:

$\exists n \leq degree\ g. (gauss\text{-}norm\ g) = val\ (g\ n)$   
*<proof>*

**lemma** *gauss-norm-smult-cfs*:

**assumes**  $g \in carrier\ Q_p\text{-}x$   
**assumes**  $a \in carrier\ Q_p$   
**assumes**  $gauss\text{-}norm\ g = val\ (g\ k)$   
**shows**  $gauss\text{-}norm\ (a \odot_{Q_p\text{-}x} g) = val\ a + val\ (g\ k)$   
*<proof>*

**lemma** *gauss-norm-smult*:

**assumes**  $g \in carrier\ Q_p\text{-}x$   
**assumes**  $a \in carrier\ Q_p$   
**shows**  $gauss\text{-}norm\ (a \odot_{Q_p\text{-}x} g) = val\ a + gauss\text{-}norm\ g$   
*<proof>*

**lemma** *gauss-norm-ultrametric*:

**assumes**  $g \in \text{carrier } Q_p\text{-}x$

**assumes**  $h \in \text{carrier } Q_p\text{-}x$

**shows**  $\text{gauss-norm } (g \oplus_{Q_p\text{-}x} h) \geq \min (\text{gauss-norm } g) (\text{gauss-norm } h)$

*<proof>*

**lemma** *gauss-norm-a-inv*:

**assumes**  $f \in \text{carrier } (UP\ Q_p)$

**shows**  $\text{gauss-norm } (\ominus_{UP\ Q_p} f) = \text{gauss-norm } f$

*<proof>*

**lemma** *gauss-norm-ultrametric'*:

**assumes**  $f \in \text{carrier } (UP\ Q_p)$

**assumes**  $g \in \text{carrier } (UP\ Q_p)$

**shows**  $\text{gauss-norm } (f \ominus_{UP\ Q_p} g) \geq \min (\text{gauss-norm } f) (\text{gauss-norm } g)$

*<proof>*

**lemma** *gauss-norm-finsum*:

**assumes**  $f \in A \rightarrow \text{carrier } Q_p\text{-}x$

**assumes** *finite*  $A$

**assumes**  $A \neq \{\}$

**shows**  $\text{gauss-norm } (\bigoplus_{Q_p\text{-}x} i \in A. f\ i) \geq \text{Min } (\text{gauss-norm } ' (f\ A))$

*<proof>*

**lemma** *gauss-norm-monom*:

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $\text{gauss-norm } (\text{monom } Q_p\text{-}x\ a\ n) = \text{val } a$

*<proof>*

**lemma** *val-val-ring-prod*:

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \text{carrier } Q_p$

**shows**  $\text{val } (a \otimes_{Q_p} b) \geq \text{val } b$

*<proof>*

**lemma** *val-val-ring-prod'*:

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \text{carrier } Q_p$

**shows**  $\text{val } (b \otimes_{Q_p} a) \geq \text{val } b$

*<proof>*

**lemma** *val-ring-nat-pow-closed*:

**assumes**  $a \in \mathcal{O}_p$

**shows**  $(a[\uparrow](n::\text{nat})) \in \mathcal{O}_p$

*<proof>*

**lemma** *val-ringI*:

**assumes**  $a \in \text{carrier } Q_p$

**assumes**  $val\ a \geq 0$   
**shows**  $a \in \mathcal{O}_p$   
 $\langle proof \rangle$

**notation**  $UPQ.to\text{-}fun$  (**infixl**  $\langle \cdot \rangle$  70)

**lemma** *val-gauss-norm-eval*:  
**assumes**  $g \in carrier\ Q_p\text{-}x$   
**assumes**  $a \in \mathcal{O}_p$   
**shows**  $val\ (g \cdot a) \geq gauss\text{-}norm\ g$   
 $\langle proof \rangle$

**lemma** *positive-gauss-norm-eval*:  
**assumes**  $g \in carrier\ Q_p\text{-}x$   
**assumes**  $gauss\text{-}norm\ g \geq 0$   
**assumes**  $a \in \mathcal{O}_p$   
**shows**  $(g \cdot a) \in \mathcal{O}_p$   
 $\langle proof \rangle$

**lemma** *positive-gauss-norm-valuation-ring-coeffs*:  
**assumes**  $g \in carrier\ Q_p\text{-}x$   
**assumes**  $gauss\text{-}norm\ g \geq 0$   
**shows**  $g\ n \in \mathcal{O}_p$   
 $\langle proof \rangle$

**lemma** *val-ring-cfs-imp-nonneg-gauss-norm*:  
**assumes**  $g \in carrier\ (UP\ Q_p)$   
**assumes**  $\bigwedge n. g\ n \in \mathcal{O}_p$   
**shows**  $gauss\text{-}norm\ g \geq 0$   
 $\langle proof \rangle$

**lemma** *val-of-add-pow*:  
**assumes**  $a \in carrier\ Q_p$   
**shows**  $val\ ([n::nat].a) \geq val\ a$   
 $\langle proof \rangle$

**lemma** *gauss-norm-pderiv*:  
**assumes**  $g \in carrier\ (UP\ Q_p)$   
**shows**  $gauss\text{-}norm\ g \leq gauss\text{-}norm\ (pderiv\ g)$   
 $\langle proof \rangle$

## 9.2 Mapping Polynomials with Value Ring Coefficients to Polynomials over $\mathbb{Z}_p$

**definition** *to- $\mathbb{Z}_p$ -poly* where  
 $to\text{-}\mathbb{Z}_p\text{-poly}\ g = (\lambda n. to\text{-}\mathbb{Z}_p\ (g\ n))$

**lemma** *to- $\mathbb{Z}_p$ -poly-closed*:  
**assumes**  $g \in carrier\ Q_p\text{-}x$

**assumes** *gauss-norm*  $g \geq 0$   
**shows** *to-Zp-poly*  $g \in \text{carrier } (UP\ Z_p)$   
 $\langle \text{proof} \rangle$

**definition** *poly-inc where*  
*poly-inc*  $g = (\lambda n::\text{nat. } \iota (g\ n))$

**lemma** *poly-inc-closed:*  
**assumes**  $g \in \text{carrier } (UP\ Z_p)$   
**shows** *poly-inc*  $g \in \text{carrier } Q_p\text{-}x$   
 $\langle \text{proof} \rangle$

**lemma** *poly-inc-inverse-right:*  
**assumes**  $g \in \text{carrier } (UP\ Z_p)$   
**shows** *to-Zp-poly* (*poly-inc*  $g$ ) =  $g$   
 $\langle \text{proof} \rangle$

**lemma** *poly-inc-inverse-left:*  
**assumes**  $g \in \text{carrier } Q_p\text{-}x$   
**assumes** *gauss-norm*  $g \geq 0$   
**shows** *poly-inc* (*to-Zp-poly*  $g$ ) =  $g$   
 $\langle \text{proof} \rangle$

**lemma** *poly-inc-plus:*  
**assumes**  $f \in \text{carrier } (UP\ Z_p)$   
**assumes**  $g \in \text{carrier } (UP\ Z_p)$   
**shows** *poly-inc* ( $f \oplus_{UP\ Z_p} g$ ) = *poly-inc*  $f \oplus_{UP\ Q_p}$  *poly-inc*  $g$   
 $\langle \text{proof} \rangle$

**lemma** *poly-inc-monom:*  
**assumes**  $a \in \text{carrier } Z_p$   
**shows** *poly-inc* (*monom* ( $UP\ Z_p$ )  $a\ m$ ) = *monom* ( $UP\ Q_p$ ) ( $\iota\ a$ )  $m$   
 $\langle \text{proof} \rangle$

**lemma** *poly-inc-times:*  
**assumes**  $f \in \text{carrier } (UP\ Z_p)$   
**assumes**  $g \in \text{carrier } (UP\ Z_p)$   
**shows** *poly-inc* ( $f \otimes_{UP\ Z_p} g$ ) = *poly-inc*  $f \otimes_{UP\ Q_p}$  *poly-inc*  $g$   
 $\langle \text{proof} \rangle$

**lemma** *poly-inc-one:*  
*poly-inc* ( $\mathbf{1}_{UP\ Z_p}$ ) =  $\mathbf{1}_{UP\ Q_p}$   
 $\langle \text{proof} \rangle$

**lemma** *poly-inc-zero:*  
*poly-inc* ( $\mathbf{0}_{UP\ Z_p}$ ) =  $\mathbf{0}_{UP\ Q_p}$   
 $\langle \text{proof} \rangle$

**lemma** *poly-inc-hom:*

$poly-inc \in ring-hom (UP Z_p) (UP Q_p)$   
 $\langle proof \rangle$

**lemma** *poly-inc-as-poly-lift-hom*:  
assumes  $f \in carrier (UP Z_p)$   
shows  $poly-inc f = poly-lift-hom Z_p Q_p \iota f$   
 $\langle proof \rangle$

**lemma** *poly-inc-eval*:  
assumes  $g \in carrier (UP Z_p)$   
assumes  $a \in carrier Z_p$   
shows  $to-function Q_p (poly-inc g) (\iota a) = \iota (to-function Z_p g a)$   
 $\langle proof \rangle$

**lemma** *val-ring-poly-eval*:  
assumes  $f \in carrier (UP Q_p)$   
assumes  $\bigwedge i. f i \in \mathcal{O}_p$   
shows  $\bigwedge x. x \in \mathcal{O}_p \implies f \cdot x \in \mathcal{O}_p$   
 $\langle proof \rangle$

**lemma** *Zp-res-of-pow*:  
assumes  $a \in carrier Z_p$   
assumes  $b \in carrier Z_p$   
assumes  $a n = b n$   
shows  $(a[\wedge]_{Z_p}(k::nat)) n = (b[\wedge]_{Z_p}(k::nat)) n$   
 $\langle proof \rangle$

**lemma** *to-Zp-nat-pow*:  
assumes  $a \in \mathcal{O}_p$   
shows  $to-Zp (a[\wedge](n::nat)) = (to-Zp a)[\wedge]_{Z_p}(n::nat)$   
 $\langle proof \rangle$

**lemma** *to-Zp-res-of-pow*:  
assumes  $a \in \mathcal{O}_p$   
assumes  $b \in \mathcal{O}_p$   
assumes  $to-Zp a n = to-Zp b n$   
shows  $to-Zp (a[\wedge](k::nat)) n = to-Zp (b[\wedge](k::nat)) n$   
 $\langle proof \rangle$

**lemma** *poly-eval-cong*:  
assumes  $g \in carrier (UP Q_p)$   
assumes  $\bigwedge i. g i \in \mathcal{O}_p$   
assumes  $a \in \mathcal{O}_p$   
assumes  $b \in \mathcal{O}_p$   
assumes  $to-Zp a k = to-Zp b k$   
shows  $to-Zp (g \cdot a) k = to-Zp (g \cdot b) k$   
 $\langle proof \rangle$

**lemma** *to-Zp-poly-eval*:

**assumes**  $g \in \text{carrier } Q_p\text{-}x$   
**assumes**  $\text{gauss-norm } g \geq 0$   
**assumes**  $a \in \mathcal{O}_p$   
**shows**  $\text{to-}Z_p (\text{to-function } Q_p g a) = \text{to-function } Z_p (\text{to-}Z_p\text{-poly } g) (\text{to-}Z_p a)$   
 $\langle \text{proof} \rangle$

**lemma** *poly-eval-equal-val*:  
**assumes**  $g \in \text{carrier } (UP Q_p)$   
**assumes**  $\bigwedge x. g x \in \mathcal{O}_p$   
**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $b \in \mathcal{O}_p$   
**assumes**  $\text{val } (g \cdot a) < \text{eint } n$   
**assumes**  $\text{to-}Z_p a n = \text{to-}Z_p b n$   
**shows**  $\text{val } (g \cdot b) = \text{val } (g \cdot a)$   
 $\langle \text{proof} \rangle$

**lemma** *to-}Z\_p\text{-poly-monom}*:  
**assumes**  $a \in \mathcal{O}_p$   
**shows**  $\text{to-}Z_p\text{-poly } (\text{monom } (UP Q_p) a n) = \text{monom } (UP Z_p) (\text{to-}Z_p a) n$   
 $\langle \text{proof} \rangle$

**lemma** *to-}Z\_p\text{-poly-add}*:  
**assumes**  $f \in \text{carrier } (UP Q_p)$   
**assumes**  $\text{gauss-norm } f \geq 0$   
**assumes**  $g \in \text{carrier } (UP Q_p)$   
**assumes**  $\text{gauss-norm } g \geq 0$   
**shows**  $\text{to-}Z_p\text{-poly } (f \oplus_{UP Q_p} g) = \text{to-}Z_p\text{-poly } f \oplus_{UP Z_p} \text{to-}Z_p\text{-poly } g$   
 $\langle \text{proof} \rangle$

**lemma** *to-}Z\_p\text{-poly-zero}*:  
 $\text{to-}Z_p\text{-poly } (\mathbf{0}_{UP Q_p}) = \mathbf{0}_{UP Z_p}$   
 $\langle \text{proof} \rangle$

**lemma** *to-}Z\_p\text{-poly-one}*:  
 $\text{to-}Z_p\text{-poly } (\mathbf{1}_{UP Q_p}) = \mathbf{1}_{UP Z_p}$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-add-pow*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $\text{val } a \geq 0$   
**shows**  $\text{val } ([n::\text{nat}]\cdot a) \geq 0$   
 $\langle \text{proof} \rangle$

**lemma** *to-}Z\_p\text{-poly-pderiv}*:  
**assumes**  $g \in \text{carrier } (UP Q_p)$   
**assumes**  $\text{gauss-norm } g \geq 0$   
**shows**  $\text{to-}Z_p\text{-poly } (\text{pderiv } g) = Z_p.\text{pderiv } (\text{to-}Z_p\text{-poly } g)$   
 $\langle \text{proof} \rangle$

**lemma** *val-p-int-pow*:  
 $\text{val } (\mathfrak{p}[\ulcorner]k) = \text{eint } (k)$   
 ⟨proof⟩

**definition** *int-gauss-norm where*  
 $\text{int-gauss-norm } g = (\text{SOME } n::\text{int. } \text{eint } n = \text{gauss-norm } g)$

**lemma** *int-gauss-norm-eq*:  
 assumes  $g \in \text{carrier } (UP \ Q_p)$   
 assumes  $g \neq \mathbf{0}_{UP \ Q_p}$   
 shows  $\text{eint } (\text{int-gauss-norm } g) = \text{gauss-norm } g$   
 ⟨proof⟩

**lemma** *int-gauss-norm-smult*:  
 assumes  $g \in \text{carrier } (UP \ Q_p)$   
 assumes  $g \neq \mathbf{0}_{UP \ Q_p}$   
 assumes  $a \in \text{nonzero } Q_p$   
 shows  $\text{int-gauss-norm } (a \odot_{UP \ Q_p} g) = \text{ord } a + \text{int-gauss-norm } g$   
 ⟨proof⟩

**definition** *normalize-poly where*  
 $\text{normalize-poly } g = (\text{if } g = \mathbf{0}_{UP \ Q_p} \text{ then } g \text{ else } (\mathfrak{p}[\ulcorner](- \text{int-gauss-norm } g)) \odot_{Q_p-x} g)$

**lemma** *normalize-poly-zero*:  
 $\text{normalize-poly } \mathbf{0}_{UP \ Q_p} = \mathbf{0}_{UP \ Q_p}$   
 ⟨proof⟩

**lemma** *normalize-poly-nonzero-eq*:  
 assumes  $g \neq \mathbf{0}_{UP \ Q_p}$   
 assumes  $g \in \text{carrier } (UP \ Q_p)$   
 shows  $\text{normalize-poly } g = (\mathfrak{p}[\ulcorner](- \text{int-gauss-norm } g)) \odot_{UP \ Q_p} g$   
 ⟨proof⟩

**lemma** *int-gauss-norm-normalize-poly*:  
 assumes  $g \neq \mathbf{0}_{UP \ Q_p}$   
 assumes  $g \in \text{carrier } (UP \ Q_p)$   
 shows  $\text{int-gauss-norm } (\text{normalize-poly } g) = 0$   
 ⟨proof⟩

**lemma** *normalize-poly-closed*:  
 assumes  $g \in \text{carrier } (UP \ Q_p)$   
 shows  $\text{normalize-poly } g \in \text{carrier } (UP \ Q_p)$   
 ⟨proof⟩

**lemma** *normalize-poly-nonzero*:  
 assumes  $g \neq \mathbf{0}_{UP \ Q_p}$   
 assumes  $g \in \text{carrier } (UP \ Q_p)$

**shows**  $\text{normalize-poly } g \neq \mathbf{0}_{UP \ Q_p}$   
 ⟨proof⟩

**lemma** *gauss-norm-normalize-poly*:

**assumes**  $g \neq \mathbf{0}_{UP \ Q_p}$   
**assumes**  $g \in \text{carrier } (UP \ Q_p)$   
**shows**  $\text{gauss-norm } (\text{normalize-poly } g) = 0$   
 ⟨proof⟩

**lemma** *taylor-term-eval-eq*:

**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $t \in \text{carrier } Q_p$   
**assumes**  $\bigwedge j. i \neq j \implies \text{val } (UPQ.\text{taylor-term } x \ f \ i \cdot t) < \text{val } (UPQ.\text{taylor-term } x \ f \ j \cdot t)$   
**shows**  $\text{val } (f \cdot t) = \text{val } (UPQ.\text{taylor-term } x \ f \ i \cdot t)$   
 ⟨proof⟩

### 9.3 Hensel's Lemma for $p$ -adic fields

**theorem** *hensels-lemma*:

**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $\text{gauss-norm } f \geq 0$   
**assumes**  $\text{val } (f \cdot a) > 2 * \text{val } ((p\text{deriv } f) \cdot a)$   
**shows**  $\exists ! \alpha \in \mathcal{O}_p. f \cdot \alpha = \mathbf{0} \wedge \text{val } (a \ominus \alpha) > \text{val } ((p\text{deriv } f) \cdot a)$   
 ⟨proof⟩

**lemma** *nth-root-poly-root-fixed*:

**assumes**  $(n::\text{nat}) > 1$   
**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $\text{val } (\mathbf{1} \ominus_{Q_p} a) > 2 * \text{val } ([n] \cdot \mathbf{1})$   
**shows**  $(\exists ! b \in \mathcal{O}_p. (b \frown n) = a \wedge \text{val } (b \ominus \mathbf{1}) > \text{val } ([n] \cdot \mathbf{1}))$   
 ⟨proof⟩

**lemma** *mod-zeroE*:

**assumes**  $(a::\text{int}) \text{ mod } k = 0$   
**shows**  $\exists l. a = l * k$   
 ⟨proof⟩

**lemma** *to-Zp-poly-closed'*:

**assumes**  $g \in \text{carrier } (UP \ Q_p)$   
**assumes**  $\bigwedge i. g \ i \in \mathcal{O}_p$   
**shows**  $\text{to-Zp-poly } g \in \text{carrier } (UP \ Z_p)$   
 ⟨proof⟩

**lemma** *to-Zp-poly-eval-to-Zp*:

**assumes**  $g \in \text{carrier } (UP \ Q_p)$   
**assumes**  $\bigwedge i. g \ i \in \mathcal{O}_p$

**assumes**  $a \in \mathcal{O}_p$   
**shows** *to-function*  $Z_p$  (*to-Zp-poly*  $g$ ) (*to-Zp*  $a$ ) = *to-Zp* ( $g \cdot a$ )  
 ⟨*proof*⟩

**lemma** *inc-nat-pow*:  
**assumes**  $a \in \text{carrier } Z_p$   
**shows**  $\iota ([n::\text{nat}]) \cdot_{Z_p} a = [n] \cdot (\iota a)$   
 ⟨*proof*⟩

**lemma** *poly-inc-pderiv*:  
**assumes**  $g \in \text{carrier } (UP\ Z_p)$   
**shows** *poly-inc* ( $Z_p.pderiv\ g$ ) = *UPQ.pderiv* (*poly-inc*  $g$ )  
 ⟨*proof*⟩

**lemma** *Zp-hensels-lemma*:  
**assumes**  $f \in \text{carrier } Z_p\text{-}x$   
**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $Z_p.to\text{-fun } (Z_p.pderiv\ f)\ a \neq \mathbf{0}_{Z_p}$   
**assumes**  $Z_p.to\text{-fun } f\ a \neq \mathbf{0}_{Z_p}$   
**assumes**  $val\text{-}Z_p\ (Z_p.to\text{-fun } f\ a) > \text{eint } 2 * val\text{-}Z_p\ (Z_p.to\text{-fun } (Z_p.pderiv\ f)\ a)$   
**obtains**  $\alpha$  **where**  
 $Z_p.to\text{-fun } f\ \alpha = \mathbf{0}_{Z_p}$  **and**  $\alpha \in \text{carrier } Z_p$   
 $val\text{-}Z_p\ (a \ominus_{Z_p} \alpha) > val\text{-}Z_p\ (Z_p.to\text{-fun } (Z_p.pderiv\ f)\ a)$   
 $val\text{-}Z_p\ (a \ominus_{Z_p} \alpha) = val\text{-}Z_p\ (\text{divide } (Z_p.to\text{-fun } f\ a)\ (Z_p.to\text{-fun } (Z_p.pderiv\ f)\ a))$   
 a))  
 $val\text{-}Z_p\ (Z_p.to\text{-fun } (Z_p.pderiv\ f)\ \alpha) = val\text{-}Z_p\ (Z_p.to\text{-fun } (Z_p.pderiv\ f)\ a)$   
 ⟨*proof*⟩

**end**

**end**

**theory** *Padic-Field-Topology*

**imports** *Padic-Fields*

**begin**

## 10 Topology of $p$ -adic Fields

In this section we develop some basic properties of the topology on the  $p$ -adics. Open and closed sets are defined, convex subsets of the value group are characterized.

**type-synonym** *padic-univ-poly* = *nat*  $\Rightarrow$  *padic-number*

### 10.1 $p$ -adic Balls

**context** *padic-fields*

**begin**

**definition** *c-ball* :: *int*  $\Rightarrow$  *padic-number*  $\Rightarrow$  *padic-number set* ( $B_{-}[-]$ ) **where**

$c\text{-ball } n \ c = \{x \in \text{carrier } Q_p. \text{ val } (x \ominus c) \geq n\}$

**lemma** *c-ballI*:

**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $\text{val } (x \ominus c) \geq n$   
**shows**  $x \in c\text{-ball } n \ c$   
*<proof>*

**lemma** *c-ballE*:

**assumes**  $x \in c\text{-ball } n \ c$   
**shows**  $x \in \text{carrier } Q_p$   
 $\text{val } (x \ominus c) \geq n$   
*<proof>*

**lemma** *c-ball-in-Qp*:

$B_n[c] \subseteq \text{carrier } Q_p$   
*<proof>*

**definition**

$q\text{-ball} :: \text{nat} \Rightarrow \text{int} \Rightarrow \text{int} \Rightarrow \text{padic-number} \Rightarrow \text{padic-number set}$  **where**  
 $q\text{-ball } n \ k \ m \ c = \{x \in \text{carrier } Q_p. (\text{ac } n \ (x \ominus c) = k \wedge (\text{ord } (x \ominus c)) = m)\}$

**lemma** *q-ballI*:

**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $\text{ac } n \ (x \ominus c) = k$   
**assumes**  $(\text{ord } (x \ominus c)) = m$   
**shows**  $x \in q\text{-ball } n \ k \ m \ c$   
*<proof>*

**lemma** *q-ballE*:

**assumes**  $x \in q\text{-ball } n \ k \ m \ c$   
**shows**  $x \in \text{carrier } Q_p$

*<proof>*

**lemma** *q-ballE'*:

**assumes**  $x \in q\text{-ball } n \ k \ m \ c$   
**shows**  $\text{ac } n \ (x \ominus c) = k$   
 $(\text{ord } (x \ominus c)) = m$   
*<proof>*

**lemma** *q-ball-in-Qp*:

$q\text{-ball } n \ k \ m \ c \subseteq \text{carrier } Q_p$   
*<proof>*

**lemma** *ac-ord-prop*:

**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{nonzero } Q_p$   
**assumes**  $\text{ord } a = \text{ord } b$

**assumes**  $\text{ord } a = n$   
**assumes**  $\text{ac } m \ a = \text{ac } m \ b$   
**assumes**  $m > 0$   
**shows**  $\text{val } (a \ominus b) \geq m + n$   
 <proof>

**lemma** *c-ball-q-ball*:  
**assumes**  $b \in \text{nonzero } Q_p$   
**assumes**  $n > 0$   
**assumes**  $k = \text{ac } n \ b$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $d \in \text{q-ball } n \ k \ m \ c$   
**shows**  $\text{q-ball } n \ k \ m \ c = \text{c-ball } (m + n) \ d$   
 <proof>

**definition** *is-ball* :: *padic-number set*  $\Rightarrow$  *bool* **where**  
*is-ball*  $B = (\exists (m::\text{int}). \exists c \in \text{carrier } Q_p. (B = B_m[c]))$

**lemma** *is-ball-imp-in-Qp*:  
**assumes** *is-ball*  $B$   
**shows**  $B \subseteq \text{carrier } Q_p$   
 <proof>

**lemma** *c-ball-centers*:  
**assumes** *is-ball*  $B$   
**assumes**  $B = B_n[c]$   
**assumes**  $d \in B$   
**assumes**  $c \in \text{carrier } Q_p$   
**shows**  $B = B_n[d]$   
 <proof>

**lemma** *c-ball-center-in*:  
**assumes** *is-ball*  $B$   
**assumes**  $B = B_n[c]$   
**assumes**  $c \in \text{carrier } Q_p$   
**shows**  $c \in B$   
 <proof>

Every point  $a$  has a point  $b$  of distance exactly  $n$  away from it.

**lemma** *dist-nonempty*:  
**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $\exists b \in \text{carrier } Q_p. \text{val } (b \ominus a) = \text{eint } n$   
 <proof>

**lemma** *dist-nonempty'*:  
**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $\exists b \in \text{carrier } Q_p. \text{val } (b \ominus a) = \alpha$   
 <proof>

**lemma** *ball-rad-0*:  
**assumes** *is-ball*  $B$   
**assumes**  $B_m[c] \subseteq B_n[c]$   
**assumes**  $c \in \text{carrier } Q_p$   
**shows**  $n \leq m$   
 $\langle \text{proof} \rangle$

**lemma** *ball-rad*:  
**assumes** *is-ball*  $B$   
**assumes**  $B = B_n[c]$   
**assumes**  $B = B_m[c]$   
**assumes**  $c \in \text{carrier } Q_p$   
**shows**  $n = m$   
 $\langle \text{proof} \rangle$

**definition** *radius* :: *padic-number set*  $\Rightarrow$  *int (rad)* **where**  
*radius*  $B = (\text{SOME } n. (\exists c \in \text{carrier } Q_p . B = B_n[c]))$

**lemma** *radius-of-ball*:  
**assumes** *is-ball*  $B$   
**assumes**  $c \in B$   
**shows**  $B = B_{\text{rad } B}[c]$   
 $\langle \text{proof} \rangle$

**lemma** *ball-rad'*:  
**assumes** *is-ball*  $B$   
**assumes**  $B = B_n[c]$   
**assumes**  $B = B_m[d]$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $d \in \text{carrier } Q_p$   
**shows**  $n = m$   
 $\langle \text{proof} \rangle$

**lemma** *nested-balls*:  
**assumes** *is-ball*  $B$   
**assumes**  $B = B_n[c]$   
**assumes**  $B' = B_m[c]$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $d \in \text{carrier } Q_p$   
**shows**  $n \geq m \iff B \subseteq B'$   
 $\langle \text{proof} \rangle$

**lemma** *nested-balls'*:  
**assumes** *is-ball*  $B$   
**assumes** *is-ball*  $B'$   
**assumes**  $B \cap B' \neq \{\}$   
**shows**  $B \subseteq B' \vee B' \subseteq B$   
 $\langle \text{proof} \rangle$

**definition** *is-bounded*:: *padic-number set*  $\Rightarrow$  *bool* **where**  
*is-bounded*  $S = (\exists n. \exists c \in \text{carrier } Q_p. S \subseteq B_n[c] )$

**lemma** *empty-is-bounded*:  
*is-bounded*  $\{ \}$   
 $\langle \text{proof} \rangle$

## 10.2 $p$ -adic Open Sets

**definition** *is-open*:: *padic-number set*  $\Rightarrow$  *bool* **where**  
*is-open*  $U \equiv (U \subseteq \text{carrier } Q_p) \wedge (\forall c \in U. \exists n. B_n[c] \subseteq U )$

**lemma** *is-openI*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**assumes**  $\bigwedge c. c \in U \Longrightarrow \exists n. B_n[c] \subseteq U$   
**shows** *is-open*  $U$   
 $\langle \text{proof} \rangle$

**lemma** *ball-is-open*:  
**assumes** *is-ball*  $B$   
**shows** *is-open*  $B$   
 $\langle \text{proof} \rangle$

**lemma** *is-open-imp-in-Qp*:  
**assumes** *is-open*  $U$   
**shows**  $U \subseteq \text{carrier } Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *is-open-imp-in-Qp'*:  
**assumes** *is-open*  $U$   
**assumes**  $x \in U$   
**shows**  $x \in \text{carrier } Q_p$   
 $\langle \text{proof} \rangle$

Owing to the total disconnectedness of the  $p$ -adic field, every open set can be decomposed into a disjoint union of balls which are maximal with respect to containment in that set. This unique decomposition is occasionally useful.

**definition** *is-max-ball-of* :: *padic-number set*  $\Rightarrow$  *padic-number set*  $\Rightarrow$  *bool* **where**  
*is-max-ball-of*  $U B \equiv (is-ball B) \wedge (B \subseteq U) \wedge (\forall B'. ((is-ball B') \wedge (B' \subseteq U) \wedge B \subseteq B') \longrightarrow B' \subseteq B)$

**lemma** *is-max-ball-ofI*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**assumes**  $(B_m[c]) \subseteq U$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $\forall m'. m' < m \longrightarrow \neg B_{m'}[c] \subseteq U$   
**shows** *is-max-ball-of*  $U (B_m[c])$   
 $\langle \text{proof} \rangle$

**lemma** *int-prop*:  
**fixes**  $P :: int \Rightarrow bool$   
**assumes**  $P\ n$   
**assumes**  $\forall m. m \leq N \longrightarrow \neg P\ m$   
**shows**  $\exists n. P\ n \wedge (\forall n'. P\ n' \longrightarrow n' \geq n)$   
 $\langle proof \rangle$

**lemma** *open-max-ball*:  
**assumes** *is-open*  $U$   
**assumes**  $U \neq \text{carrier } Q_p$   
**assumes**  $c \in U$   
**shows**  $\exists B. \text{is-max-ball-of } U\ B \wedge c \in B$   
 $\langle proof \rangle$

**definition** *interior where*  
 $\text{interior } U = \{a. \exists B. \text{is-open } B \wedge B \subseteq U \wedge a \in B\}$

**lemma** *interior-subset*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**shows**  $\text{interior } U \subseteq U$   
 $\langle proof \rangle$

**lemma** *interior-open*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**shows** *is-open* ( $\text{interior } U$ )  
 $\langle proof \rangle$

**lemma** *interiorI*:  
**assumes**  $W \subseteq U$   
**assumes** *is-open*  $W$   
**shows**  $W \subseteq \text{interior } U$   
 $\langle proof \rangle$

**lemma** *max-ball-interior*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**assumes**  $\text{is-max-ball-of } (\text{interior } U)\ B$   
**shows**  $\text{is-max-ball-of } U\ B$   
 $\langle proof \rangle$

**lemma** *ball-in-max-ball*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**assumes**  $U \neq \text{carrier } Q_p$   
**assumes**  $c \in U$   
**assumes**  $\exists B. B \subseteq U \wedge \text{is-ball } B \wedge c \in B$   
**shows**  $\exists B'. \text{is-max-ball-of } U\ B' \wedge c \in B'$   
 $\langle proof \rangle$

**lemma** *ball-in-max-ball'*:  
**assumes**  $U \subseteq \text{carrier } Q_p$

**assumes**  $U \neq \text{carrier } Q_p$   
**assumes**  $B \subseteq U \wedge \text{is-ball } B$   
**shows**  $\exists B'. \text{is-max-ball-of } U B' \wedge B \subseteq B'$   
 <proof>

**lemma** *max-balls-disjoint*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**assumes**  $\text{is-max-ball-of } U B$   
**assumes**  $\text{is-max-ball-of } U B'$   
**assumes**  $B \neq B'$   
**shows**  $B \cap B' = \{\}$   
 <proof>

**definition** *max-balls* :: *padic-number set*  $\Rightarrow$  *padic-number set set* **where**  
*max-balls*  $U = \{B. \text{is-max-ball-of } U B\}$

**lemma** *max-balls-interior*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**assumes**  $U \neq \text{carrier } Q_p$   
**shows**  $\text{interior } U = \{x \in \text{carrier } Q_p. (\exists B \in (\text{max-balls } U). x \in B)\}$   
 <proof>

**lemma** *max-balls-interior'*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**assumes**  $U \neq \text{carrier } Q_p$   
**assumes**  $B \in \text{max-balls } U$   
**shows**  $B \subseteq \text{interior } U$   
 <proof>

**lemma** *max-balls-interior''*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**assumes**  $U \neq \text{carrier } Q_p$   
**assumes**  $a \in \text{interior } U$   
**shows**  $\exists B \in \text{max-balls } U. a \in B$   
 <proof>

**lemma** *open-interior*:  
**assumes**  $\text{is-open } U$   
**shows**  $\text{interior } U = U$   
 <proof>

**lemma** *interior-idempotent*:  
**assumes**  $U \subseteq \text{carrier } Q_p$   
**shows**  $\text{interior } (\text{interior } U) = \text{interior } U$   
 <proof>

### 10.3 Convex Subsets of the Value Group

The content of this section will be useful for defining and reasoning about  $p$ -adic cells in the proof of Macintyre's theorem. It is proved that every convex set in the extended integers is either an open ray, a closed ray, a closed interval, or a left-closed interval.

**definition** *is-convex* :: *eint set*  $\Rightarrow$  *bool* **where**

*is-convex*  $A = (\forall x \in A. \forall y \in A. \forall c. x \leq c \wedge c \leq y \longrightarrow c \in A)$

**lemma** *is-convexI*:

**assumes**  $\bigwedge x y c. x \in A \Longrightarrow y \in A \Longrightarrow x \leq c \wedge c \leq y \Longrightarrow c \in A$

**shows** *is-convex*  $A$

*<proof>*

**lemma** *is-convexE*:

**assumes** *is-convex*  $A$

**assumes**  $x \in A$

**assumes**  $y \in A$

**assumes**  $x \leq a$

**assumes**  $a \leq y$

**shows**  $a \in A$

*<proof>*

**lemma** *empty-convex*:

*is-convex*  $\{\}$

*<proof>*

**lemma** *UNIV-convex*:

*is-convex* *UNIV*

*<proof>*

**definition** *closed-interval* (*I*[- -]) **where**

*closed-interval*  $\alpha \beta = \{a . \alpha \leq a \wedge a \leq \beta\}$

**lemma** *closed-interval-is-convex*:

**assumes**  $A = \text{closed-interval } \alpha \beta$

**shows** *is-convex*  $A$

*<proof>*

**lemma** *empty-closed-interval*:

$\{\} = \text{closed-interval } \infty (\text{eint } 1)$

*<proof>*

**definition** *left-closed-interval* **where**

*left-closed-interval*  $\alpha \beta = \{a . \alpha \leq a \wedge a < \beta\}$

**lemma** *left-closed-interval-is-convex*:

**assumes**  $A = \text{left-closed-interval } \alpha \beta$

**shows** *is-convex*  $A$

*<proof>*

**definition** *closed-ray* **where**

*closed-ray*  $\alpha \beta = \{a . a \leq \beta\}$

**lemma** *closed-ray-is-convex*:

**assumes**  $A = \text{closed-ray } \alpha \beta$

**shows** *is-convex*  $A$

*<proof>*

**lemma** *UNIV-closed-ray*:

$(\text{UNIV}::\text{eint set}) = \text{closed-ray } \alpha \infty$

*<proof>*

**definition** *open-ray* :: *eint*  $\Rightarrow$  *eint*  $\Rightarrow$  *eint set* **where**

*open-ray*  $\alpha \beta = \{a . a < \beta\}$

**lemma** *open-ray-is-convex*:

**assumes**  $A = \text{open-ray } \alpha \beta$

**shows** *is-convex*  $A$

*<proof>*

**lemma** *open-rayE*:

**assumes**  $a < \beta$

**shows**  $a \in \text{open-ray } \alpha \beta$

*<proof>*

**lemma** *value-group-is-open-ray*:

$\text{UNIV} - \{\infty\} = \text{open-ray } \alpha \infty$

*<proof>*

This is a predicate which identifies a certain kind of set-valued function on the extended integers. Convex conditions will be important in the definition of  $p$ -adic cells later, and it will be proved that every convex set is induced by a convex condition.

**definition** *is-convex-condition* :: (*eint*  $\Rightarrow$  *eint*  $\Rightarrow$  *eint set*)  $\Rightarrow$  *bool*

**where** *is-convex-condition*  $I \equiv$

$I = \text{closed-interval} \vee I = \text{left-closed-interval} \vee I = \text{closed-ray} \vee I = \text{open-ray}$

**lemma** *convex-condition-imp-convex*:

**assumes** *is-convex-condition*  $I$

**shows** *is-convex*  $(I \alpha \beta)$

*<proof>*

**lemma** *bounded-order*:

**assumes**  $(a::\text{eint}) < \infty$

**assumes**  $b \leq a$

**obtains**  $k::\text{nat}$  **where**  $a = b + k$

*<proof>*

Every convex set is given by a convex condition

**lemma** *convex-imp-convex-condition*:

**assumes** *is-convex*  $A$

**shows**  $\exists I \alpha \beta. \text{is-convex-condition } I \wedge A = (I \alpha \beta)$

*<proof>*

**lemma** *ex-val-less*:

**shows**  $\exists (\alpha::\text{eint}). \alpha < \beta$

*<proof>*

**lemma** *ex-dist-less*:

**assumes**  $c \in \text{carrier } Q_p$

**shows**  $\exists a \in \text{carrier } Q_p. \text{val } (a \ominus c) < \beta$

*<proof>*

**end**

**end**

**theory** *Generated-Boolean-Algebra*

**imports** *Main*

**begin**

## 11 Generated Boolean Algebras of Sets

### 11.1 Definitions and Basic Lemmas

**lemma** *equalityI'*:

**assumes**  $\bigwedge x. x \in A \implies x \in B$

**assumes**  $\bigwedge x. x \in B \implies x \in A$

**shows**  $A = B$

*<proof>*

**lemma** *equalityI''*:

**assumes**  $\bigwedge x. A \ x \implies B \ x$

**assumes**  $\bigwedge x. B \ x \implies A \ x$

**shows**  $\{x. A \ x\} = \{x. B \ x\}$

*<proof>*

**lemma** *SomeE*:

**assumes**  $a = (\text{SOME } x. P \ x)$

**assumes**  $P \ c$

**shows**  $P \ a$

*<proof>*

**lemma** *SomeE'*:

**assumes**  $a = (\text{SOME } x. P \ x)$

**assumes**  $\exists x. P \ x$

**shows**  $P \ a$

*<proof>*

## 12 Basic notions about boolean algebras over a set $S$ , generated by a set of generators $B$

Note that the generators  $B$  need not be subsets of the set  $S$

**inductive-set** *gen-boolean-algebra*

**for**  $S$  **and**  $B$  **where**

*universe:*  $S \in \text{gen-boolean-algebra } S B$

| *generator:*  $A \in B \implies A \cap S \in \text{gen-boolean-algebra } S B$

| *union:*  $\llbracket A \in \text{gen-boolean-algebra } S B; C \in \text{gen-boolean-algebra } S B \rrbracket \implies A \cup C \in \text{gen-boolean-algebra } S B$

| *complement:*  $A \in \text{gen-boolean-algebra } S B \implies S - A \in \text{gen-boolean-algebra } S B$

**lemma** *gen-boolean-algebra-subset:*

**shows**  $A \in \text{gen-boolean-algebra } S B \implies A \subseteq S$

*<proof>*

**lemma** *gen-boolean-algebra-intersect:*

**assumes**  $A \in \text{gen-boolean-algebra } S B$

**assumes**  $C \in \text{gen-boolean-algebra } S B$

**shows**  $A \cap C \in \text{gen-boolean-algebra } S B$

*<proof>*

**lemma** *gen-boolean-algebra-diff:*

**assumes**  $A \in \text{gen-boolean-algebra } S B$

**assumes**  $C \in \text{gen-boolean-algebra } S B$

**shows**  $A - C \in \text{gen-boolean-algebra } S B$

*<proof>*

**lemma** *gen-boolean-algebra-diff-eq:*

**assumes**  $A \in \text{gen-boolean-algebra } S B$

**assumes**  $C \in \text{gen-boolean-algebra } S B$

**shows**  $A - C = A \cap (S - C)$

*<proof>*

**lemma** *gen-boolean-algebra-finite-union:*

**assumes**  $\bigwedge a. a \in A \implies a \in \text{gen-boolean-algebra } S B$

**assumes** *finite*  $A$

**shows**  $\bigcup A \in \text{gen-boolean-algebra } S B$

*<proof>*

**lemma** *gen-boolean-algebra-finite-intersection:*

**assumes**  $\bigwedge a. a \in A \implies a \in \text{gen-boolean-algebra } S B$

**assumes** *finite*  $A$

**assumes**  $A \neq \{\}$

**shows**  $\bigcap A \in \text{gen-boolean-algebra } S B$

*<proof>*

**lemma** *gen-boolean-algebra-generators:*

**assumes**  $\bigwedge b. b \in B \implies b \subseteq S$   
**assumes**  $b \in B$   
**shows**  $b \in \text{gen-boolean-algebra } S B$   
 ⟨proof⟩

**lemma** *gen-boolean-algebra-generator-subset*:  
**assumes**  $A \in \text{gen-boolean-algebra } S As$   
**assumes**  $As \subseteq Bs$   
**shows**  $A \in \text{gen-boolean-algebra } S Bs$   
 ⟨proof⟩

**lemma** *gen-boolean-algebra-generators-union*:  
**assumes**  $A \in \text{gen-boolean-algebra } S As$   
**assumes**  $C \in \text{gen-boolean-algebra } S Cs$   
**shows**  $A \cup C \in \text{gen-boolean-algebra } S (As \cup Cs)$   
 ⟨proof⟩

**lemma** *gen-boolean-algebra-finite-gen-wits*:  
**assumes**  $A \in \text{gen-boolean-algebra } S B$   
**shows**  $\exists Bs. \text{finite } Bs \wedge Bs \subseteq B \wedge A \in \text{gen-boolean-algebra } S Bs$   
 ⟨proof⟩

**lemma** *gen-boolean-algebra-univ-mono*:  
**assumes**  $A \in \text{gen-boolean-algebra } S B$   
**shows**  $\text{gen-boolean-algebra } A B \subseteq \text{gen-boolean-algebra } S B$   
 ⟨proof⟩

The boolean algebra generated by a collection of elements in another algebra is contained in the original algebra:

**lemma** *gen-boolean-algebra-subalgebra*:  
**assumes**  $Xs \subseteq \text{gen-boolean-algebra } S B$   
**shows**  $\text{gen-boolean-algebra } S Xs \subseteq \text{gen-boolean-algebra } S B$   
 ⟨proof⟩

**lemma** *gen-boolean-algebra-idempotent*:  
**assumes**  $S = \bigcup Xs$   
**shows**  $\text{gen-boolean-algebra } S (\text{gen-boolean-algebra } S Xs) = (\text{gen-boolean-algebra } S Xs)$   
 ⟨proof⟩

We can always replace the set of generators  $Xs$  with their intersections with the universe set  $S$ , and obtain the same algebra.

**lemma** *gen-boolean-algebra-restrict-generators*:  
 $\text{gen-boolean-algebra } S Xs = \text{gen-boolean-algebra } S ((\cap) S \text{ ' } Xs)$   
 ⟨proof⟩

Adding a generator to a generated boolean algebra is redundant if the generator already lies in the algebra.

**lemma** *add-generators*:

**assumes**  $A \in \text{gen-boolean-algebra } S \ Xs$   
**shows**  $\text{gen-boolean-algebra } S \ Xs = \text{gen-boolean-algebra } S \ (\text{insert } A \ Xs)$   
 <proof>

## 12.1 Turning a Family of Sets into a Family of Disjoint Sets

This section outlines the standard construction where sets  $A_0, \dots, A_n$  are replaced by sets  $A_0, A_1 - A_0, A_2 - (A_0 \cup A_1), \dots, A_n - (\bigcup_{i=0}^{n-1} A_i)$  to obtain a disjoint family of the same cardinality.

**fun** *rec-disjointify* **where**  
*rec-disjointify* 0  $f = \{\}$   
*rec-disjointify* (Suc  $m$ )  $f = \text{insert } (f \ m - \bigcup (\text{rec-disjointify } m \ f)) (\text{rec-disjointify } m \ f)$

**lemma** *card-of-rec-disjointify*:  
 $\text{card } (\text{rec-disjointify } m \ f) \leq m$   
 <proof>

**lemma** *rec-disjointify-finite*:  
 $\text{finite } (\text{rec-disjointify } m \ f)$   
 <proof>

**lemma** *rec-disjointify-in-gen-boolean-algebra*:  
**assumes**  $f \ ' \ \{..<m\} \subseteq \text{gen-boolean-algebra } S \ B$   
**shows**  $\text{rec-disjointify } m \ f \subseteq \text{gen-boolean-algebra } S \ B$   
 <proof>

**lemma** *rec-disjointify-union*:  
 $\bigcup (\text{rec-disjointify } m \ f) = (\bigcup i \in \{..<m\}. f \ i)$   
 <proof>

**definition** *enum-rec-disjointify* **where**  
 $\text{enum-rec-disjointify } f \ m = f \ m - \bigcup (\text{rec-disjointify } m \ f)$

**lemma** *rec-disjointify-as-enum-rec-disjointify-image*:  
 $\text{rec-disjointify } m \ f = \text{enum-rec-disjointify } f \ ' \ \{..<m\}$   
 <proof>

**lemma** *enum-rec-disjointify-subset*:  
 $\text{enum-rec-disjointify } f \ m \subseteq f \ m$   
 <proof>

**lemma** *enum-rec-disjointify-disjoint*:  
**assumes**  $k < m$   
**shows**  $\text{enum-rec-disjointify } f \ m \cap \text{enum-rec-disjointify } f \ k = \{\}$   
 <proof>

**lemma** *enum-rec-disjointify-disjoint'*:

**assumes**  $k \neq m$

**shows**  $\text{enum-rec-disjointify } f \ m \cap \text{enum-rec-disjointify } f \ k = \{\}$

*<proof>*

**lemma** *rec-disjointify-is-disjoint*:

**assumes**  $A \in \text{rec-disjointify } m \ f$

**assumes**  $B \in \text{rec-disjointify } m \ f$

**assumes**  $A \neq B$

**shows**  $A \cap B = \{\}$

*<proof>*

**definition** *enumerates where*

$\text{enumerates } A \ f \equiv \text{finite } A \wedge A = f \ ' \ \{.. < (\text{card } A)\} \wedge \text{inj-on } f \ \{.. < (\text{card } A)\}$

**lemma** *finite-imp-exists-enumeration*:

**assumes** *finite*  $A$

**shows**  $\exists f. \text{enumerates } A \ f$

*<proof>*

**lemma** *enumeratesE*:

**assumes** *enumerates*  $A \ f$

**shows**  $\text{finite } A \ A = f \ ' \ \{.. < \text{card } A\} \ \text{inj-on } f \ \{.. < \text{card } A\}$

*<proof>*

**lemma** *rec-disjointify-finite-set*:

**assumes** *enumerates*  $A \ f$

**shows**  $\bigcup (\text{rec-disjointify } (\text{card } A) \ f) = \bigcup A$

*<proof>*

**definition** *enumerate where*

$\text{enumerate } A = (\text{SOME } f. \text{enumerates } A \ f)$

**lemma** *enumerate-enumerates*:

**assumes** *finite*  $A$

**shows** *enumerates*  $A \ (\text{enumerate } A)$

*<proof>*

**lemma** *enumerateE*:

**assumes** *finite*  $A$

**assumes**  $a \in A$

**shows**  $\exists i < \text{card } A. a = (\text{enumerate } A) \ i$

*<proof>*

**definition** *disjointify where*

$\text{disjointify } As = \text{rec-disjointify } (\text{card } As) \ (\text{enumerate } As)$

**lemma** *disjointify-is-disjoint*:

**assumes** *finite*  $As$

**assumes**  $A \in \text{disjointify } As$   
**assumes**  $B \in \text{disjointify } As$   
**assumes**  $A \neq B$   
**shows**  $A \cap B = \{\}$   
*<proof>*

**lemma** *disjointify-union:*  
**assumes** *finite*  $As$   
**shows**  $\bigcup (\text{disjointify } As) = \bigcup As$   
*<proof>*

**lemma** *disjointify-gen-boolean-algebra:*  
**assumes** *finite*  $As$   
**assumes**  $As \subseteq \text{gen-boolean-algebra } S B$   
**shows**  $\text{disjointify } As \subseteq \text{gen-boolean-algebra } S B$   
*<proof>*

**lemma** *disjointify-finite:*  
**assumes** *finite*  $As$   
**shows** *finite*  $(\text{disjointify } As)$   
*<proof>*

**lemma** *disjointify-card:*  
**assumes** *finite*  $As$   
**shows**  $\text{card } (\text{disjointify } As) \leq \text{card } As$   
*<proof>*

**lemma** *disjointify-subset:*  
**assumes** *finite*  $As$   
**assumes**  $A \in \text{disjointify } As$   
**shows**  $\exists B \in As. A \subseteq B$   
*<proof>*

## 12.2 The Atoms Generated by Collections of Sets

We can also turn a family of sets into a disjoint family by taking the atoms of the boolean algebra generated by these sets. This will still yield a finite family if the initial family is finite, but in general will be much larger in size.

### 12.2.1 Defining the Atoms of a Family of Sets

Here we intend that  $As$  is a subset of the collection of sets  $Xs$ . This function associate to each subset  $As \subseteq Xs$  a set which is contained in each element of  $As$ , and is disjoint from each element of  $Xs - As$ . Note that in general this may yield the empty set, but we will ultimately be interested in the cases where the result is nonempty.

**definition** *subset-to-atom* **where**

$$\text{subset-to-atom } Xs \text{ } As = \bigcap As - \bigcup (Xs - As)$$

**lemma** *subset-to-atom-memI*:

**assumes**  $\bigwedge A. A \in As \implies x \in A$   
**assumes**  $\bigwedge A. A \in Xs \implies A \notin As \implies x \notin A$   
**shows**  $x \in \text{subset-to-atom } Xs \text{ } As$   
 $\langle \text{proof} \rangle$

**lemma** *subset-to-atom-memE*:

**assumes**  $x \in \text{subset-to-atom } Xs \text{ } As$   
**shows**  $\bigwedge A. A \in As \implies x \in A$   
 $\bigwedge A. A \in Xs \implies A \notin As \implies x \notin A$   
 $\langle \text{proof} \rangle$

**lemma** *subset-to-atom-closed*:

**assumes**  $As \neq \{\}$   
**assumes**  $As \subseteq Xs$   
**shows**  $\text{subset-to-atom } Xs \text{ } As \subseteq \bigcup Xs$   
 $\langle \text{proof} \rangle$

**lemma** *subset-to-atom-as-intersection*:

**assumes**  $As \neq \{\}$   
**assumes**  $As \subseteq Xs$   
**assumes**  $S = \bigcup Xs$   
**shows**  $\text{subset-to-atom } Xs \text{ } As = \bigcap As \cap (\bigcap X \in Xs - As. S - X)$   
 $\langle \text{proof} \rangle$

**definition** *atoms-of where*

$$\text{atoms-of } Xs = (\text{subset-to-atom } Xs \text{ } ((\text{Pow } Xs) - \{\{\}\})) - \{\{\}\}$$

**lemma** *atoms-nonempty*:

**assumes**  $A \in \text{atoms-of } Xs$   
**shows**  $A \neq \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *atoms-of-disjoint*:

**assumes**  $A \in \text{atoms-of } Xs$   
**assumes**  $B \in \text{atoms-of } Xs$   
**assumes**  $A \neq B$   
**shows**  $A \cap B = \{\}$   
 $\langle \text{proof} \rangle$

The atoms of a family of sets  $Xs$  are minimal in the sense that they are either contained in or disjoint from each element of  $Xs$ .

**lemma** *atoms-are-minimal*:

**assumes**  $A \in \text{atoms-of } Xs$   
**assumes**  $X \in Xs$   
**shows**  $X \cap A = \{\} \vee A \subseteq X$   
 $\langle \text{proof} \rangle$

### 12.2.2 Atoms Induced by Types of Points

The set of sets in  $Xs$  which contain some point  $x$ . In the case where  $Xs$  is some collection of first order formulas, this is just the type of  $x$  over these formulas.

**definition** *point-to-type* **where**  
*point-to-type*  $Xs\ x = \{X \in Xs. x \in X\}$

The type of a point  $x$  induces the unique atom of  $Xs$  which contains  $x$ .

**lemma** *point-in-atom-of-type*:  
**assumes**  $x \in \bigcup Xs$   
**shows**  $x \in \text{subset-to-atom } Xs$  (*point-to-type*  $Xs\ x$ )  
(*proof*)

**lemma** *point-to-type-nonempty*:  
**assumes**  $x \in \bigcup Xs$   
**shows** *point-to-type*  $Xs\ x \neq \{\}$   
(*proof*)

**lemma** *point-to-type-closed*:  
*point-to-type*  $Xs\ x \subseteq \text{Pow } (\bigcup Xs)$   
(*proof*)

**lemma** *atoms-of-covers*:  
**assumes**  $X = \bigcup Xs$   
**shows**  $\bigcup (\text{atoms-of } Xs) = X$   
(*proof*)

**lemma** *atoms-of-covers'*:  
**shows**  $\bigcup (\text{atoms-of } Xs) = \bigcup Xs$   
(*proof*)

Every atom of a collection  $Xs$  of sets is realized as the atom generated by the type of an element in that atom.

**lemma** *nonempty-atom-from-point-to-type*:  
**assumes**  $A \in \text{atoms-of } Xs$   
**assumes**  $a \in A$   
**shows**  $A = \text{subset-to-atom } Xs$  (*point-to-type*  $Xs\ a$ )  
(*proof*)

In light of the previous theorem, a point  $a$  and a collection of sets  $Xs$  is enough to recover the the unique atom of  $Xs$  which contains  $a$ .

**definition** *point-to-atom* **where**  
*point-to-atom*  $Xs\ a = \text{subset-to-atom } Xs$  (*point-to-type*  $Xs\ a$ )

**lemma** *point-to-atom-closed*:  
**assumes**  $x \in \bigcup Xs$   
**shows** *point-to-atom*  $Xs\ x \in \text{atoms-of } Xs$

*<proof>*

All atoms of  $Xs$  are the atom induced by some point in the union of  $Xs$ .

**lemma** *atoms-induced-by-points:*

*atoms-of*  $Xs = \text{point-to-atom } Xs \text{ ' } (\bigcup Xs)$

*<proof>*

### 12.2.3 Atoms of Generated Boolean Algebras

**lemma** *atoms-of-gen-boolean-algebra:*

**assumes**  $Xs \subseteq \text{gen-boolean-algebra } S B$

**assumes** *finite*  $Xs$

**shows** *atoms-of*  $Xs \subseteq \text{gen-boolean-algebra } S B$

*<proof>*

If the generators of a boolean algebra are contained in the universe, the atoms induced by the generators alone are minimal elements of the entire algebra.

**lemma** *finite-algebra-atoms-are-minimal:*

**assumes** *finite*  $Xs$

**assumes**  $\bigcup Xs \subseteq S$

**assumes**  $A \in \text{atoms-of } Xs$

**assumes**  $X \in \text{gen-boolean-algebra } S Xs$

**shows**  $X \cap A = \{\} \vee A \subseteq X$

*<proof>*

**lemma** *finite-set-imp-finite-atoms:*

**assumes** *finite*  $Xs$

**shows** *finite* (*atoms-of*  $Xs$ )

*<proof>*

Every element in the boolean algebra generated by  $Xs$  over  $S$  is a (disjoint) union of atoms of generators:

**lemma** *gen-boolean-algebra-elem-uni-of-atoms:*

**assumes** *finite*  $Xs$

**assumes**  $S = \bigcup Xs$

**assumes**  $X \in \text{gen-boolean-algebra } S Xs$

**shows**  $X = \bigcup \{a \in \text{atoms-of } Xs. a \subseteq X\}$

*<proof>*

In fact, every generated boolean algebra is the power set of the atoms of its generators:

**lemma** *gen-boolean-algebra-generated-by-atoms:*

**assumes** *finite*  $Xs$

**assumes**  $S = \bigcup Xs$

**shows** *gen-boolean-algebra*  $S Xs = \bigcup \text{' } (\text{Pow } (\text{atoms-of } Xs))$

*<proof>*

Finitely generated boolean algebras are finite

**lemma** *fin-gens-imp-fin-algebra*:  
**assumes** *finite Xs*  
**assumes**  $S = \bigcup Xs$   
**shows** *finite (gen-boolean-algebra S Xs)*  
 $\langle proof \rangle$

**lemma** *point-to-atom-equal*:  
**assumes** *finite Xs*  
**assumes**  $S = \bigcup Xs$   
**assumes**  $x \in S$   
**shows** *point-to-atom Xs x = point-to-atom (gen-boolean-algebra S Xs) x*  
 $\langle proof \rangle$

When the set  $Xs$  of generators covers the universe set  $S$ , the atoms of  $Xs$  in the above sense are the same as the atoms of the boolean algebra they generate over  $S$ .

**lemma** *atoms-of-sets-eq-atoms-of-algebra*:  
**assumes** *finite Xs*  
**assumes**  $S = \bigcup Xs$   
**shows** *atoms-of Xs = atoms-of (gen-boolean-algebra S Xs)*  
 $\langle proof \rangle$

**lemma** *atoms-closed*:  
**assumes** *finite Xs*  
**assumes**  $A \in \text{atoms-of (gen-boolean-algebra S Xs)}$   
**assumes**  $S = \bigcup Xs$   
**shows**  $A \in (\text{gen-boolean-algebra S Xs})$   
 $\langle proof \rangle$

**lemma** *atoms-finite*:  
**assumes** *finite Xs*  
**shows** *finite ((atoms-of (gen-boolean-algebra S Xs)))*  
 $\langle proof \rangle$

We can distinguish atoms of a set of generators  $Cs$  by finding some element of  $Cs$  which includes one and excludes the other.

**lemma** *distinct-atoms*:  
**assumes**  $Cs \neq \{\}$   
**assumes**  $a \in \text{atoms-of } Cs$   
**assumes**  $b \in \text{atoms-of } Cs$   
**assumes**  $a \neq b$   
**shows**  $(\exists B \in Cs. b \subseteq B \wedge a \cap B = \{\}) \vee (\exists A \in Cs. a \subseteq A \wedge b \cap A = \{\})$   
 $\langle proof \rangle$

### 12.3 Partitions of a Set

**definition** *disjoint* :: 'a set set  $\Rightarrow$  bool **where**  
*disjoint Ss* =  $(\forall A \in Ss. \forall B \in Ss. A \neq B \longrightarrow A \cap B = \{\})$

**lemma disjointE:**  
**assumes** *disjoint Ss*  
**assumes**  $A \in Ss$   
**assumes**  $B \in Ss$   
**assumes**  $A \neq B$   
**shows**  $A \cap B = \{\}$   
 $\langle proof \rangle$

**lemma disjointI:**  
**assumes**  $\bigwedge A B. A \in Ss \implies B \in Ss \implies A \neq B \implies A \cap B = \{\}$   
**shows** *disjoint Ss*  
 $\langle proof \rangle$

**definition is-partition** :: 'a set set  $\Rightarrow$  'a set  $\Rightarrow$  bool (**infixl** *partitions* 75) **where**  
*S partitions A* = (*disjoint S*  $\wedge$   $\bigcup S = A$ )

**lemma is-partitionE:**  
**assumes** *S partitions A*  
**shows** *disjoint S*  
 $\bigcup S = A$   
 $\langle proof \rangle$

**lemma is-partitionI:**  
**assumes** *disjoint S*  
**assumes**  $\bigcup S = A$   
**shows** *S partitions A*  
 $\langle proof \rangle$

If we start with a finite partition of a set  $A$ , and each element in that partition has a finite partition with some property  $P$ , then  $A$  itself has a finite partition where each element has property  $P$ .

**lemma iter-partition:**  
**assumes** *As partitions A*  
**assumes** *finite As*  
**assumes**  $\bigwedge a. a \in As \implies \exists Bs. \text{finite } Bs \wedge Bs \text{ partitions } a \wedge (\forall b \in Bs. P b)$   
**shows**  $\exists Bs. \text{finite } Bs \wedge Bs \text{ partitions } A \wedge (\forall b \in Bs. P b)$   
 $\langle proof \rangle$

## 12.4 Intersections of Families of Sets

**definition pairwise-intersect** **where**  
*pairwise-intersect As Bs* =  $\{c. \exists a \in As. \exists b \in Bs. c = a \cap b\}$

**lemma partition-intersection:**  
**assumes** *As partitions A*  
**assumes** *Bs partitions B*  
**shows** (*pairwise-intersect As Bs*) *partitions*  $(A \cap B)$   
 $\langle proof \rangle$

**lemma** *pairwise-intersect-finite*:

**assumes** *finite As*

**assumes** *finite Bs*

**shows** *finite (pairwise-intersect As Bs)*

*<proof>*

**definition** *family-intersect where*

*family-intersect parts = atoms-of (∪ parts)*

**lemma** *family-intersect-partitions*:

**assumes**  $\bigwedge Ps. Ps \in parts \implies Ps \text{ partitions } A$

**assumes**  $\bigwedge Ps. Ps \in parts \implies \text{finite } Ps$

**assumes** *finite parts*

**assumes**  $parts \neq \{\}$

**shows** *family-intersect parts partitions A*

*<proof>*

**lemma** *family-intersect-memE*:

**assumes**  $\bigwedge Ps. Ps \in parts \implies Ps \text{ partitions } A$

**assumes**  $\bigwedge Ps. Ps \in parts \implies \text{finite } Ps$

**assumes** *finite parts*

**assumes**  $parts \neq \{\}$

**shows**  $\bigwedge Ps a. a \in \text{family-intersect parts} \implies Ps \in parts \implies \exists P \in Ps. a \subseteq P$

*<proof>*

**lemma** *family-intersect-mem-inter*:

**assumes**  $\bigwedge Ps. Ps \in (parts:: 'a \text{ set set set}) \implies Ps \text{ partitions } A$

**assumes**  $\bigwedge Ps. Ps \in parts \implies \text{finite } Ps$

**assumes** *finite parts*

**assumes**  $parts \neq \{\}$

**assumes** *a ∈ family-intersect parts*

**shows**  $\exists f. \forall Ps \in parts. f Ps \in Ps \wedge a = (\bigcap Ps \in parts. f Ps)$

*<proof>*

If we take a finite family of partitions in a particular generated boolean algebra, where each partition itself is finite, then their induced partition is also in the algebra.

**lemma** *family-intersect-in-gen-boolean-algebra*:

**assumes**  $A \in \text{gen-boolean-algebra } S B$

**assumes**  $\bigwedge Ps. Ps \in parts \implies Ps \text{ partitions } A$

**assumes**  $\bigwedge Ps. Ps \in parts \implies \text{finite } Ps$

**assumes**  $\bigwedge Ps P. Ps \in parts \implies P \in Ps \implies P \in \text{gen-boolean-algebra } S B$

**assumes** *finite parts*

**assumes**  $parts \neq \{\}$

**shows**  $\bigwedge P. P \in \text{family-intersect parts} \implies P \in \text{gen-boolean-algebra } S B$

*<proof>*

```

end
theory Padic-Field-Powers
  imports Ring-Powers Padic-Field-Polynomials Generated-Boolean-Algebra
           Padic-Field-Topology

```

```

begin

```

This theory is intended to develop the necessary background on subsets of powers of a  $p$ -adic field to prove Macintyre's quantifier elimination theorem. In particular, we define semi-algebraic subsets of  $\mathbb{Q}_p^n$ , semi-algebraic functions  $\mathbb{Q}_p^n \rightarrow \mathbb{Q}_p$ , and semi-algebraic mappings  $\mathbb{Q}_p^n \rightarrow \mathbb{Q}_p^m$  for arbitrary  $n, m \in \mathbb{N}$ . In addition we prove that many common sets and functions are semi-algebraic. We are closely following the paper [1] by Denef, where an algebraic proof of Macintyre's theorem is developed.

### 13 Cartesian Powers of $p$ -adic Fields

```

lemma list-tl:

```

```

tl (t#x) = x
  <proof>

```

```

lemma list-hd:

```

```

hd (t#x) = t
  <proof>

```

```

sublocale padic-fields < cring-coord-rings  $\mathbb{Q}_p$  UP  $\mathbb{Q}_p$ 

```

```

  <proof>

```

```

sublocale padic-fields <  $\mathbb{Q}_p$ : domain-coord-rings  $\mathbb{Q}_p$  UP  $\mathbb{Q}_p$ 

```

```

  <proof>

```

```

context padic-fields

```

```

begin

```

```

no-notation Zp.to-fun (infixl  $\langle \cdot \rangle$  70)

```

```

no-notation ideal-prod (infixl  $\cdot$  80)

```

```

notation

```

```

evimage (infixr  $^{-1}$  90) and

```

```

euminus-set ( $-$   $^c$  70)

```

```

type-synonym padic-tuple = padic-number list

```

```

type-synonym padic-function = padic-number  $\Rightarrow$  padic-number

```

```

type-synonym padic-nary-function = padic-tuple  $\Rightarrow$  padic-number

```

```

type-synonym padic-function-tuple = padic-nary-function list

```

**type-synonym** *padic-nary-function-poly* = *nat*  $\Rightarrow$  *padic-nary-function*

### 13.1 Polynomials over $\mathbb{Q}_p$ and Polynomial Maps

**lemma** *last-closed'*:

**assumes**  $x@t \in \text{carrier } (Q_p^n)$   
**shows**  $t \in \text{carrier } Q_p$   
*<proof>*

**lemma** *segment-in-car'*:

**assumes**  $x@t \in \text{carrier } (Q_p^{\text{Suc } n})$   
**shows**  $x \in \text{carrier } (Q_p^n)$   
*<proof>*

**lemma** *Qp-zero*:

$Q_p^0 = \text{nil-ring}$   
*<proof>*

**lemma** *Qp-zero-carrier*:

$\text{carrier } (Q_p^0) = \{\}\}$   
*<proof>*

Abbreviation for constant polynomials

**abbreviation**(*input*) *Qp-to-IP* where  
*Qp-to-IP*  $k \equiv \text{Qp.indexed-const } k$

**lemma** *Qp-to-IP-car*:

**assumes**  $k \in \text{carrier } Q_p$   
**shows** *Qp-to-IP*  $k \in \text{carrier } (Q_p[\mathcal{X}_n])$   
*<proof>*

**lemma**(in *cring-coord-rings*) *smult-closed*:

**assumes**  $a \in \text{carrier } R$   
**assumes**  $q \in \text{carrier } (R[\mathcal{X}_n])$   
**shows**  $a \odot_{R[\mathcal{X}_n]} q \in \text{carrier } (R[\mathcal{X}_n])$   
*<proof>*

**lemma** *Qp-poly-smult-cfs*:

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $P \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $(a \odot_{Q_p[\mathcal{X}_n]} P) m = a \otimes (P m)$   
*<proof>*

**lemma** *Qp-smult-r-distr*:

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $P \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $q \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $a \odot_{Q_p[\mathcal{X}_n]} (P \oplus_{Q_p[\mathcal{X}_n]} q) = (a \odot_{Q_p[\mathcal{X}_n]} P) \oplus_{Q_p[\mathcal{X}_n]} (a \odot_{Q_p[\mathcal{X}_n]} q)$

$\langle proof \rangle$

**lemma** *Qp-smult-l-distr*:

**assumes**  $a \in carrier\ Q_p$

**assumes**  $b \in carrier\ Q_p$

**assumes**  $P \in carrier\ (Q_p[\mathcal{X}_n])$

**shows**  $(a \oplus b) \odot_{Q_p[\mathcal{X}_n]} P = (a \odot_{Q_p[\mathcal{X}_n]} P) \oplus_{Q_p[\mathcal{X}_n]} (b \odot_{Q_p[\mathcal{X}_n]} P)$

$\langle proof \rangle$

**abbreviation** *(input) Qp-funs where*

*Qp-funs*  $n \equiv Fun_n\ Q_p$

## 13.2 Evaluation of Polynomials in $\mathbb{Q}_p$

**abbreviation** *(input) Qp-ev where*

*Qp-ev*  $P\ q \equiv (eval-at-point\ Q_p\ q\ P)$

**lemma** *Qp-ev-one*:

**assumes**  $a \in carrier\ (Q_p^n)$

**shows**  $Qp-ev\ \mathbf{1}_{Q_p[\mathcal{X}_n]}\ a = \mathbf{1}\ \langle proof \rangle$

**lemma** *Qp-ev-zero*:

**assumes**  $a \in carrier\ (Q_p^n)$

**shows**  $Qp-ev\ \mathbf{0}_{Q_p[\mathcal{X}_n]}\ a = \mathbf{0}\ \langle proof \rangle$

**lemma** *Qp-eval-pvar-pow*:

**assumes**  $a \in carrier\ (Q_p^n)$

**assumes**  $k < n$

**assumes**  $(m::nat) \neq 0$

**shows**  $Qp-ev\ ((pvar\ Q_p\ k)[\wedge]_{Q_p[\mathcal{X}_n]}\ m)\ a = ((a!k)[\wedge]m)$

$\langle proof \rangle$

composition of polynomials over  $\mathbb{Q}_p$

**definition** *Qp-poly-comp where*

*Qp-poly-comp*  $m\ fs = poly-compose\ (length\ fs)\ m\ fs$

lemmas about polynomial maps

**lemma** *Qp-is-poly-tupleI*:

**assumes**  $\bigwedge i. i < length\ fs \implies fs!i \in carrier\ (Q_p[\mathcal{X}_m])$

**shows** *is-poly-tuple*  $m\ fs$

$\langle proof \rangle$

**lemma** *Qp-is-poly-tuple-append*:

**assumes** *is-poly-tuple*  $m\ fs$

**assumes** *is-poly-tuple*  $m\ gs$

**shows** *is-poly-tuple*  $m\ (fs@gs)$

$\langle proof \rangle$

**lemma** *Qp-poly-mapE*:

**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes**  $length\ fs = m$   
**assumes**  $as \in carrier\ (Q_p^n)$   
**assumes**  $j < m$   
**shows**  $(poly-map\ n\ fs\ as)!j \in carrier\ Q_p$   
 $\langle proof \rangle$

**lemma** *Qp-poly-mapE'*:  
**assumes**  $as \in carrier\ (Q_p^n)$   
**shows**  $length\ (poly-map\ n\ fs\ as) = length\ fs$   
 $\langle proof \rangle$

**lemma** *Qp-poly-mapE''*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes**  $length\ fs = m$   
**assumes**  $n \neq 0$   
**assumes**  $as \in carrier\ (Q_p^n)$   
**assumes**  $j < m$   
**shows**  $(poly-map\ n\ fs\ as)!j = (Qp-ev\ (fs!j)\ as)$   
 $\langle proof \rangle$

**lemma** *poly-map-apply*:  
**assumes**  $as \in carrier\ (Q_p^n)$   
**shows**  $poly-map\ n\ fs\ as = poly-tuple-eval\ fs\ as$   
 $\langle proof \rangle$

**lemma** *poly-map-pullbackI*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes**  $as \in carrier\ (Q_p^n)$   
**assumes**  $poly-map\ n\ fs\ as \in S$   
**shows**  $as \in poly-map\ n\ fs^{-1}_n\ S$   
 $\langle proof \rangle$

**lemma** *poly-map-pullbackI'*:  
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes**  $as \in carrier\ (Q_p^n)$   
**assumes**  $poly-map\ n\ fs\ as \in S$   
**shows**  $as \in ((poly-map\ n\ fs) - ' S)$   
 $\langle proof \rangle$

lemmas about polynomial composition

**lemma** *poly-compose-ring-hom*:  
**assumes** *is-poly-tuple*  $m$   $fs$   
**assumes**  $length\ fs = n$   
**shows**  $(ring-hom-ring\ (Q_p[\mathcal{X}_n])\ (Q_p[\mathcal{X}_m])\ (Qp-poly-comp\ m\ fs))$   
 $\langle proof \rangle$

**lemma** *poly-compose-closed*:  
**assumes** *is-poly-tuple*  $m$   $fs$

**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (Q_p[\mathcal{X}_n])$   
**shows**  $(Qp\text{-poly-comp}\ m\ fs\ f) \in carrier\ (Q_p[\mathcal{X}_m])$   
 $\langle proof \rangle$

**lemma** *poly-compose-add:*

**assumes** *is-poly-tuple*  $m\ fs$   
**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in carrier\ (Q_p[\mathcal{X}_n])$   
**shows**  $Qp\text{-poly-comp}\ m\ fs\ (f \oplus_{Q_p[\mathcal{X}_n]}\ g) = (Qp\text{-poly-comp}\ m\ fs\ f) \oplus_{Q_p[\mathcal{X}_m]}$   
 $(Qp\text{-poly-comp}\ m\ fs\ g)$   
 $\langle proof \rangle$

**lemma** *poly-compose-mult:*

**assumes** *is-poly-tuple*  $m\ fs$   
**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in carrier\ (Q_p[\mathcal{X}_n])$   
**shows**  $Qp\text{-poly-comp}\ m\ fs\ (f \otimes_{Q_p[\mathcal{X}_n]}\ g) = (Qp\text{-poly-comp}\ m\ fs\ f) \otimes_{Q_p[\mathcal{X}_m]}$   
 $(Qp\text{-poly-comp}\ m\ fs\ g)$   
 $\langle proof \rangle$

**lemma** *poly-compose-const:*

**assumes** *is-poly-tuple*  $m\ fs$   
**assumes**  $length\ fs = n$   
**assumes**  $a \in carrier\ Q_p$   
**shows**  $Qp\text{-poly-comp}\ m\ fs\ (Qp\text{-to-IP}\ a) = Qp\text{-to-IP}\ a$   
 $\langle proof \rangle$

**lemma** *Qp-poly-comp-eval:*

**assumes** *is-poly-tuple*  $m\ fs$   
**assumes**  $length\ fs = n$   
**assumes**  $f \in carrier\ (Q_p[\mathcal{X}_n])$   
**assumes**  $as \in carrier\ (Q_p^m)$   
**shows**  $Qp\text{-ev}\ (Qp\text{-poly-comp}\ m\ fs\ f)\ as = Qp\text{-ev}\ f\ (poly\text{-map}\ m\ fs\ as)$   
 $\langle proof \rangle$

### 13.3 Mapping Univariate Polynomials to Multivariable Polynomials in One Variable

**abbreviation**(*input*) *to-Qp-x* **where**

$to\text{-}Qp\text{-}x \equiv (IP\text{-to-UP}\ (0::nat)) :: (nat\ multiset \Rightarrow padic\text{-}number) \Rightarrow nat \Rightarrow padic\text{-}number)$

**abbreviation**(*input*) *from-Qp-x* **where**

$from\text{-}Qp\text{-}x \equiv UP\text{-to-IP}\ Q_p\ (0::nat)$

**lemma** *from-Qp-x-closed:*

**assumes**  $q \in \text{carrier } Q_{p-x}$   
**shows**  $\text{from-}Q_{p-x} q \in \text{carrier } (Q_p[\mathcal{X}_1])$   
 $\langle \text{proof} \rangle$

**lemma** *to-}Q\_{p-x}-closed:*  
**assumes**  $q \in \text{carrier } (Q_p[\mathcal{X}_1])$   
**shows**  $\text{to-}Q_{p-x} q \in \text{carrier } Q_{p-x}$   
 $\langle \text{proof} \rangle$

**lemma** *to-}Q\_{p-x}-from-}Q\_{p-x}:*  
**assumes**  $q \in \text{carrier } (Q_p[\mathcal{X}_1])$   
**shows**  $\text{from-}Q_{p-x} (\text{to-}Q_{p-x} q) = q$   
 $\langle \text{proof} \rangle$

**lemma** *from-}Q\_{p-x}-to-}Q\_{p-x}:*  
**assumes**  $q \in \text{carrier } Q_{p-x}$   
**shows**  $\text{to-}Q_{p-x} (\text{from-}Q_{p-x} q) = q$   
 $\langle \text{proof} \rangle$

ring hom properties of these maps

**lemma** *to-}Q\_{p-x}-ring-hom:*  
 $\text{to-}Q_{p-x} \in \text{ring-hom } (Q_p[\mathcal{X}_1]) Q_{p-x}$   
 $\langle \text{proof} \rangle$

**lemma** *from-}Q\_{p-x}-ring-hom:*  
 $\text{from-}Q_{p-x} \in \text{ring-hom } Q_{p-x} (Q_p[\mathcal{X}_1])$   
 $\langle \text{proof} \rangle$

**lemma** *from-}Q\_{p-x}-add:*  
**assumes**  $a \in \text{carrier } Q_{p-x}$   
**assumes**  $b \in \text{carrier } Q_{p-x}$   
**shows**  $\text{from-}Q_{p-x} (a \oplus_{Q_{p-x}} b) = \text{from-}Q_{p-x} a \oplus_{Q_p[\mathcal{X}_1]} \text{from-}Q_{p-x} b$   
 $\langle \text{proof} \rangle$

**lemma** *from-}Q\_{p-x}-mult:*  
**assumes**  $a \in \text{carrier } Q_{p-x}$   
**assumes**  $b \in \text{carrier } Q_{p-x}$   
**shows**  $\text{from-}Q_{p-x} (a \otimes_{Q_{p-x}} b) = \text{from-}Q_{p-x} a \otimes_{Q_p[\mathcal{X}_1]} \text{from-}Q_{p-x} b$   
 $\langle \text{proof} \rangle$

equivalence of evaluation maps

**lemma** *Q\_{p-poly-}Q\_{p-x}-eval:*  
**assumes**  $P \in \text{carrier } (Q_p[\mathcal{X}_1])$   
**assumes**  $a \in \text{carrier } (Q_p^1)$   
**shows**  $Q_{p\text{-ev}} P a = (\text{to-}Q_{p-x} P) \cdot (Q_{p\text{-to-}R} a)$   
 $\langle \text{proof} \rangle$

**lemma** *Q\_{p-x-}Q\_{p-poly-}eval:*

**assumes**  $P \in \text{carrier } Q_{p-x}$   
**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $P \cdot a = Q_{p\text{-ev}} (\text{from-}Q_{p-x} P) (\text{to-R1 } a)$   
 <proof>

### 13.4 $n^{\text{th}}$ -Power Sets over $Q_p$

**definition** *P-set where*

$P\text{-set } (n::\text{nat}) = \{a \in \text{nonzero } Q_p. (\exists y \in \text{carrier } Q_p. (y[\wedge] n) = a))\}$

**lemma** *P-set-carrier:*

$P\text{-set } n \subseteq \text{carrier } Q_p$   
 <proof>

**lemma** *P-set-memI:*

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $a \neq \mathbf{0}$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $b[\wedge](n::\text{nat}) = a$   
**shows**  $a \in P\text{-set } n$   
 <proof>

**lemma** *P-set-nonzero:*

$P\text{-set } n \subseteq \text{nonzero } Q_p$   
 <proof>

**lemma** *P-set-nonzero':*

**assumes**  $a \in P\text{-set } n$   
**shows**  $a \in \text{nonzero } Q_p$   
 $a \in \text{carrier } Q_p$   
 <proof>

**lemma** *P-set-one:*

**assumes**  $n \neq 0$   
**shows**  $\mathbf{1} \in P\text{-set } (n::\text{nat})$   
 <proof>

**lemma** *zeroth-P-set:*

$P\text{-set } 0 = \{\mathbf{1}\}$   
 <proof>

**lemma** *P-set-mult-closed:*

**assumes**  $n \neq 0$   
**assumes**  $a \in P\text{-set } n$   
**assumes**  $b \in P\text{-set } n$   
**shows**  $a \otimes b \in P\text{-set } n$   
 <proof>

**lemma** *P-set-inv-closed*:

**assumes**  $a \in P\text{-set } n$

**shows**  $\text{inv } a \in P\text{-set } n$

$\langle \text{proof} \rangle$

**lemma** *P-set-val*:

**assumes**  $a \in P\text{-set } (n::\text{nat})$

**shows**  $(\text{ord } a) \bmod n = 0$

$\langle \text{proof} \rangle$

**lemma** *P-set-pow*:

**assumes**  $n > 0$

**assumes**  $s \in P\text{-set } n$

**shows**  $s[\wedge]k \in P\text{-set } (n*k)$

$\langle \text{proof} \rangle$

## 13.5 Semialgebraic Sets

In this section we introduce the notion of a  $p$ -adic semialgebraic set. Intuitively, these are the subsets of  $\mathbb{Q}_p^n$  which are definable by first order quantifier-free formulas in the standard first-order language of rings, with an additional relation symbol included for the relation  $\text{val}(x) \leq \text{val}(y)$ , interpreted according to the definition of the  $p$ -adic valuation on  $\mathbb{Q}_p$ . In fact, by Macintyre’s quantifier elimination theorem for the first-order theory of  $\mathbb{Q}_p$  in this language, one can equivalently remove the “quantifier-free” clause from the latter definition. The definition we give here is also equivalent, and due to Denef in [1]. The given definition here is desirable mainly for its utility in producing a proof of Macintyre’s theorem, which is our overarching goal.

### 13.5.1 Defining Semialgebraic Sets

**definition** *basic-semialg-set where*

*basic-semialg-set*  $(m::\text{nat}) (n::\text{nat}) P = \{q \in \text{carrier } (Q_p^m). \exists y \in \text{carrier } Q_p. Qp\text{-ev } P \ q = (y[\wedge]n)\}$

**lemma** *basic-semialg-set-zero-set*:

**assumes**  $P \in \text{carrier } (Q_p[\mathcal{X}_m])$

**assumes**  $q \in \text{carrier } (Q_p^m)$

**assumes**  $Qp\text{-ev } P \ q = \mathbf{0}$

**assumes**  $n \neq 0$

**shows**  $q \in \text{basic-semialg-set } (m::\text{nat}) (n::\text{nat}) P$

$\langle \text{proof} \rangle$

**lemma** *basic-semialg-set-def'*:

**assumes**  $n \neq 0$

**assumes**  $P \in \text{carrier } (Q_p[\mathcal{X}_m])$

**shows** *basic-semialg-set* ( $m::nat$ ) ( $n::nat$ )  $P = \{q \in carrier (Q_p^m). Qp-ev P q = \mathbf{0} \vee Qp-ev P q \in (P-set n)\}$   
 $\langle proof \rangle$

**lemma** *basic-semialg-set-memI*:  
**assumes**  $q \in carrier (Q_p^m)$   
**assumes**  $y \in carrier Q_p$   
**assumes**  $Qp-ev P q = (y[\uparrow]n)$   
**shows**  $q \in basic-semialg-set m n P$   
 $\langle proof \rangle$

**lemma** *basic-semialg-set-memE*:  
**assumes**  $q \in basic-semialg-set m n P$   
**shows**  $q \in carrier (Q_p^m)$   
 $\exists y \in carrier Q_p. Qp-ev P q = (y[\uparrow]n)$   
 $\langle proof \rangle$

**definition** *is-basic-semialg* ::  $nat \Rightarrow ((nat \Rightarrow int) \times (nat \Rightarrow int)) set list set \Rightarrow bool$  **where**  
 $is-basic-semialg m S \equiv (\exists (n::nat) \neq 0. (\exists P \in carrier (Q_p[\mathcal{X}_m]). S = basic-semialg-set m n P))$

**abbreviation**(*input*) *basic-semialgs* **where**  
 $basic-semialgs m \equiv \{S. (is-basic-semialg m S)\}$

**definition** *semialg-sets* **where**  
 $semialg-sets n = gen-boolean-algebra (carrier (Q_p^n)) (basic-semialgs n)$

**lemma** *carrier-is-semialg*:  
 $(carrier (Q_p^n)) \in semialg-sets n$   
 $\langle proof \rangle$

**lemma** *empty-set-is-semialg*:  
 $\{\} \in semialg-sets n$   
 $\langle proof \rangle$

**lemma** *semialg-intersect*:  
**assumes**  $A \in semialg-sets n$   
**assumes**  $B \in semialg-sets n$   
**shows**  $(A \cap B) \in semialg-sets n$   
 $\langle proof \rangle$

**lemma** *semialg-union*:  
**assumes**  $A \in semialg-sets n$   
**assumes**  $B \in semialg-sets n$   
**shows**  $(A \cup B) \in semialg-sets n$   
 $\langle proof \rangle$

**lemma** *semialg-complement*:

**assumes**  $A \in \text{semialg-sets } n$   
**shows**  $(\text{carrier } (Q_p^n) - A) \in \text{semialg-sets } n$   
 $\langle \text{proof} \rangle$

**lemma** *semialg-zero*:  
**assumes**  $A \in \text{semialg-sets } 0$   
**shows**  $A = \{\square\} \vee A = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *basic-semialg-is-semialg*:  
**assumes** *is-basic-semialg*  $n$   $A$   
**shows**  $A \in \text{semialg-sets } n$   
 $\langle \text{proof} \rangle$

**lemma** *basic-semialg-is-semialg'*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $m \neq 0$   
**assumes**  $A = \text{basic-semialg-set } n$   $m$   $f$   
**shows**  $A \in \text{semialg-sets } n$   
 $\langle \text{proof} \rangle$

**definition** *is-semialgebraic where*  
*is-semialgebraic*  $n$   $S = (S \in \text{semialg-sets } n)$

**lemma** *is-semialgebraicE*:  
**assumes** *is-semialgebraic*  $n$   $S$   
**shows**  $S \in \text{semialg-sets } n$   
 $\langle \text{proof} \rangle$

**lemma** *is-semialgebraic-closed*:  
**assumes** *is-semialgebraic*  $n$   $S$   
**shows**  $S \subseteq \text{carrier } (Q_p^n)$   
 $\langle \text{proof} \rangle$

**lemma** *is-semialgebraicI*:  
**assumes**  $S \in \text{semialg-sets } n$   
**shows** *is-semialgebraic*  $n$   $S$   
 $\langle \text{proof} \rangle$

**lemma** *basic-semialg-is-semialgebraic*:  
**assumes** *is-basic-semialg*  $n$   $A$   
**shows** *is-semialgebraic*  $n$   $A$   
 $\langle \text{proof} \rangle$

**lemma** *basic-semialg-is-semialgebraic'*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $m \neq 0$   
**assumes**  $A = \text{basic-semialg-set } n$   $m$   $f$   
**shows** *is-semialgebraic*  $n$   $A$

*<proof>*

### 13.5.2 Algebraic Sets over $p$ -adic Fields

**lemma** *p-times-square-not-square:*

**assumes**  $a \in \text{nonzero } Q_p$   
**shows**  $\mathfrak{p} \otimes (a \text{ [ } \wedge \text{ ] } (2::\text{nat})) \notin P\text{-set } (2::\text{nat})$   
*<proof>*

**lemma** *p-times-square-not-square':*

**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $\mathfrak{p} \otimes (a \text{ [ } \wedge \text{ ] } (2::\text{nat})) = \mathbf{0} \implies a = \mathbf{0}$   
*<proof>*

**lemma** *zero-set-semialg-set:*

**assumes**  $q \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $Qp\text{-ev } q \ a = \mathbf{0} \iff (\exists y \in \text{carrier } Q_p. \mathfrak{p} \otimes ((Qp\text{-ev } q \ a) \text{ [ } \wedge \text{ ] } (2::\text{nat})) = y \text{ [ } \wedge \text{ ] } (2::\text{nat}))$   
*<proof>*

**lemma** *alg-as-semialg:*

**assumes**  $P \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $q = \mathfrak{p} \odot_{Q_p[\mathcal{X}_n]} (P \text{ [ } \wedge \text{ ] }_{Q_p[\mathcal{X}_n]} (2::\text{nat}))$   
**shows**  $\text{zero-set } Q_p \ n \ P = \text{basic-semialg-set } n \ (2::\text{nat}) \ q$   
*<proof>*

**lemma** *is-zero-set-imp-basic-semialg:*

**assumes**  $P \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $S = \text{zero-set } Q_p \ n \ P$   
**shows**  $\text{is-basic-semialg } n \ S$   
*<proof>*

**lemma** *is-zero-set-imp-semialg:*

**assumes**  $P \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $S = \text{zero-set } Q_p \ n \ P$   
**shows**  $\text{is-semialgebraic } n \ S$   
*<proof>*

Algebraic sets are semialgebraic

**lemma** *is-algebraic-imp-is-semialg:*

**assumes**  $\text{is-algebraic } Q_p \ n \ S$   
**shows**  $\text{is-semialgebraic } n \ S$   
*<proof>*

### 13.5.3 Basic Lemmas about the Semialgebraic Predicate

Finite and cofinite sets are semialgebraic

**lemma** *finite-is-semialg:*

**assumes**  $F \subseteq \text{carrier } (Q_p^n)$   
**assumes**  $\text{finite } F$   
**shows**  $\text{is-semialgebraic } n \ F$   
 $\langle \text{proof} \rangle$

**definition**  $\text{is-cofinite where}$   
 $\text{is-cofinite } n \ F = \text{finite } (\text{ring-pow-comp } Q_p \ n \ F)$

**lemma**  $\text{is-cofiniteE:}$   
**assumes**  $F \subseteq \text{carrier } (Q_p^n)$   
**assumes**  $\text{is-cofinite } n \ F$   
**shows**  $\text{finite } (\text{carrier } (Q_p^n) - F)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{complement-is-semialg:}$   
**assumes**  $\text{is-semialgebraic } n \ F$   
**shows**  $\text{is-semialgebraic } n \ ((\text{carrier } (Q_p^n)) - F)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{cofinite-is-semialgebraic:}$   
**assumes**  $F \subseteq \text{carrier } (Q_p^n)$   
**assumes**  $\text{is-cofinite } n \ F$   
**shows**  $\text{is-semialgebraic } n \ F$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{diff-is-semialgebraic:}$   
**assumes**  $\text{is-semialgebraic } n \ A$   
**assumes**  $\text{is-semialgebraic } n \ B$   
**shows**  $\text{is-semialgebraic } n \ (A - B)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{intersection-is-semialg:}$   
**assumes**  $\text{is-semialgebraic } n \ A$   
**assumes**  $\text{is-semialgebraic } n \ B$   
**shows**  $\text{is-semialgebraic } n \ (A \cap B)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{union-is-semialgebraic:}$   
**assumes**  $\text{is-semialgebraic } n \ A$   
**assumes**  $\text{is-semialgebraic } n \ B$   
**shows**  $\text{is-semialgebraic } n \ (A \cup B)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{carrier-is-semialgebraic:}$   
 $\text{is-semialgebraic } n \ (\text{carrier } (Q_p^n))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{empty-is-semialgebraic:}$   
 $\text{is-semialgebraic } n \ \{\}$

*<proof>*

### 13.5.4 One-Dimensional Semialgebraic Sets

**definition** *one-var-semialg where*

*one-var-semialg*  $S = ((to-R1 \text{ ' } S) \in (semialg-sets \ 1))$

**definition** *univ-basic-semialg-set where*

*univ-basic-semialg-set*  $(m::nat) \ P = \{a \in carrier \ Q_p. (\exists y \in carrier \ Q_p. (P \cdot a = (y[\wedge m])))\}$

Equivalence of *univ\_basic\_semialg\_sets* and semialgebraic subsets of  $\mathbb{Q}^1$

**lemma** *univ-basic-semialg-set-to-semialg-set:*

**assumes**  $P \in carrier \ Q_p$

**assumes**  $m \neq 0$

**shows**  $to-R1 \text{ ' } (univ-basic-semialg-set \ m \ P) = basic-semialg-set \ 1 \ m \ (from-Qp-x \ P)$

*<proof>*

**definition** *is-univ-semialgebraic where*

*is-univ-semialgebraic*  $S = (S \subseteq carrier \ Q_p \wedge is-semialgebraic \ 1 \ (to-R1 \text{ ' } S))$

**lemma** *is-univ-semialgebraicE:*

**assumes** *is-univ-semialgebraic*  $S$

**shows** *is-semialgebraic 1*  $(to-R1 \text{ ' } S)$

*<proof>*

**lemma** *is-univ-semialgebraicI:*

**assumes** *is-semialgebraic 1*  $(to-R1 \text{ ' } S)$

**shows** *is-univ-semialgebraic*  $S$

*<proof>*

**lemma** *univ-basic-semialg-set-is-univ-semialgebraic:*

**assumes**  $P \in carrier \ Q_p$

**assumes**  $m \neq 0$

**shows** *is-univ-semialgebraic*  $(univ-basic-semialg-set \ m \ P)$

*<proof>*

**lemma** *intersection-is-univ-semialgebraic:*

**assumes** *is-univ-semialgebraic*  $A$

**assumes** *is-univ-semialgebraic*  $B$

**shows** *is-univ-semialgebraic*  $(A \cap B)$

*<proof>*

**lemma** *union-is-univ-semialgebraic:*

**assumes** *is-univ-semialgebraic*  $A$

**assumes** *is-univ-semialgebraic*  $B$

**shows** *is-univ-semialgebraic*  $(A \cup B)$

*<proof>*

**lemma** *diff-is-univ-semialgebraic*:  
**assumes** *is-univ-semialgebraic*  $A$   
**assumes** *is-univ-semialgebraic*  $B$   
**shows** *is-univ-semialgebraic*  $(A - B)$   
 $\langle$ *proof* $\rangle$

**lemma** *finite-is-univ-semialgebraic*:  
**assumes**  $A \subseteq \text{carrier } Q_p$   
**assumes** *finite*  $A$   
**shows** *is-univ-semialgebraic*  $A$   
 $\langle$ *proof* $\rangle$

### 13.5.5 Defining the $p$ -adic Valuation Semialgebraically

**lemma** *Qp-square-root-criterion0*:  
**assumes**  $p \neq 2$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $\text{val } a \leq \text{val } b$   
**assumes**  $a \neq \mathbf{0}$   
**assumes**  $b \neq \mathbf{0}$   
**assumes**  $\text{val } a \geq 0$   
**shows**  $\exists y \in \text{carrier } Q_p. a[\uparrow](2::\text{nat}) \oplus_{Q_p} \mathfrak{p} \otimes b[\uparrow](2::\text{nat}) = (y [\uparrow] (2::\text{nat}))$   
 $\langle$ *proof* $\rangle$

**lemma** *eint-minus-ineq'*:  
**assumes**  $(a::\text{eint}) \geq b$   
**shows**  $a - b \geq 0$   
 $\langle$ *proof* $\rangle$

**lemma** *Qp-square-root-criterion*:  
**assumes**  $p \neq 2$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $\text{ord } b \geq \text{ord } a$   
**assumes**  $a \neq \mathbf{0}$   
**assumes**  $b \neq \mathbf{0}$   
**shows**  $\exists y \in \text{carrier } Q_p. a[\uparrow](2::\text{nat}) \oplus_{Q_p} \mathfrak{p} \otimes b[\uparrow](2::\text{nat}) = (y [\uparrow] (2::\text{nat}))$   
 $\langle$ *proof* $\rangle$

**lemma** *Qp-val-ring-alt-def0*:  
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $\text{ord } a \geq 0$   
**shows**  $\exists y \in \text{carrier } Q_p. \mathbf{1} \oplus_{Q_p} (\mathfrak{p}[\uparrow](3::\text{nat})) \otimes (a[\uparrow](4::\text{nat})) = (y[\uparrow](2::\text{nat}))$   
 $\langle$ *proof* $\rangle$

Defining the valuation semialgebraically for odd primes

**lemma** *P-set-ord-semialg-odd-p*:

**assumes**  $p \neq 2$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**shows**  $\text{val } a \leq \text{val } b \iff (\exists y \in \text{carrier } Q_p. (a[\uparrow](2::\text{nat})) \oplus_{Q_p} (\mathfrak{p} \otimes (b[\uparrow](2::\text{nat}))))$   
 $= (y[\uparrow](2::\text{nat}))$   
 $\langle \text{proof} \rangle$

Defining the valuation ring semialgebraically for all primes

**lemma** *Qp-val-ring-alt-def:*

**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $a \in \mathcal{O}_p \iff (\exists y \in \text{carrier } Q_p. \mathbf{1} \oplus_{Q_p} (\mathfrak{p}[\uparrow](3::\text{nat})) \otimes (a[\uparrow](4::\text{nat})) = (y[\uparrow](2::\text{nat})))$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-val-alt-def:*

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**shows**  $\text{val } b \leq \text{val } a \iff (\exists y \in \text{carrier } Q_p. (b[\uparrow](4::\text{nat})) \oplus_{Q_p} (\mathfrak{p}[\uparrow](3::\text{nat})) \otimes (a[\uparrow](4::\text{nat})) = (y[\uparrow](2::\text{nat})))$   
 $\langle \text{proof} \rangle$

The polynomial in two variables which semialgebraically defines the valuation relation

**definition** *Qp-val-poly where*

$Qp\text{-val-poly} = (\text{pvar } Q_p \ 1)[\uparrow]_{Q_p[\mathcal{X}_2]}(4::\text{nat}) \oplus_{Q_p[\mathcal{X}_2]} (\mathfrak{p}[\uparrow](3::\text{nat})) \odot_{Q_p[\mathcal{X}_2]} ((\text{pvar } Q_p \ 0)[\uparrow]_{Q_p[\mathcal{X}_2]}(4::\text{nat}))$

**lemma** *Qp-val-poly-closed:*

$Qp\text{-val-poly} \in \text{carrier } (Q_p[\mathcal{X}_2])$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-val-poly-eval:*

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**shows**  $Qp\text{-ev } Qp\text{-val-poly } [a, b] = (b[\uparrow](4::\text{nat})) \oplus_{Q_p} (\mathfrak{p}[\uparrow](3::\text{nat})) \otimes (a[\uparrow](4::\text{nat}))$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-2I:*

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**shows**  $[a, b] \in \text{carrier } (Q_p^2)$   
 $\langle \text{proof} \rangle$

**lemma** *pair-id:*

**assumes**  $\text{length } as = 2$   
**shows**  $as = [as!0, as!1]$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-val-semialg*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**shows**  $\text{val } b \leq \text{val } a \iff [a,b] \in \text{basic-semialg-set } 2 \text{ (2::nat) } Q_p\text{-val-poly}$   
 $\langle \text{proof} \rangle$

**definition** *val-relation-set where*  
 $\text{val-relation-set} = \{as \in \text{carrier } (Q_p^2). \text{val } (as!1) \leq \text{val } (as!0)\}$

**lemma** *val-relation-setE*:  
**assumes**  $as \in \text{val-relation-set}$   
**shows**  $as!0 \in \text{carrier } Q_p \wedge as!1 \in \text{carrier } Q_p \wedge as = [as!0, as!1] \wedge \text{val } (as!1) \leq \text{val } (as!0)$   
 $\langle \text{proof} \rangle$

**lemma** *val-relation-setI*:  
**assumes**  $as!0 \in \text{carrier } Q_p$   
**assumes**  $as!1 \in \text{carrier } Q_p$   
**assumes**  $\text{length } as = 2$   
**assumes**  $\text{val } (as!1) \leq \text{val } (as!0)$   
**shows**  $as \in \text{val-relation-set}$   
 $\langle \text{proof} \rangle$

**lemma** *val-relation-semialg*:  
 $\text{val-relation-set} = \text{basic-semialg-set } 2 \text{ (2::nat) } Q_p\text{-val-poly}$   
 $\langle \text{proof} \rangle$

**lemma** *val-relation-is-semialgebraic*:  
 $\text{is-semialgebraic } 2 \text{ val-relation-set}$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-val-ring-is-semialg*:  
**obtains**  $P$  **where**  $P \in \text{carrier } Q_p\text{-x} \wedge \mathcal{O}_p = \text{univ-basic-semialg-set } 2 P$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-val-ring-is-univ-semialgebraic*:  
 $\text{is-univ-semialgebraic } \mathcal{O}_p$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-val-ring-is-semialgebraic*:  
 $\text{is-semialgebraic } 1 \text{ (to-R1' } \mathcal{O}_p)$   
 $\langle \text{proof} \rangle$

### 13.5.6 Inverse Images of Semialgebraic Sets by Polynomial Maps

**lemma** *basic-semialg-pullback*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_k])$   
**assumes** *is-poly-tuple*  $n$   $fs$   
**assumes**  $\text{length } fs = k$

**assumes**  $S = \text{basic-semialg-set } k \ m \ f$   
**assumes**  $m \neq 0$   
**shows**  $\text{poly-map } n \ fs \ ^{-1}_n S = \text{basic-semialg-set } n \ m \ (Qp\text{-poly-comp } n \ fs \ f)$   
 <proof>

**lemma** *basic-semialg-pullback'*:  
**assumes** *is-poly-tuple*  $n \ fs$   
**assumes**  $\text{length } fs = k$   
**assumes**  $A \in \text{basic-semialgs } k$   
**shows**  $\text{poly-map } n \ fs \ ^{-1}_n A \in (\text{basic-semialgs } n)$   
 <proof>

**lemma** *semialg-pullback*:  
**assumes** *is-poly-tuple*  $n \ fs$   
**assumes**  $\text{length } fs = k$   
**assumes**  $S \in \text{semialg-sets } k$   
**shows**  $\text{poly-map } n \ fs \ ^{-1}_n S \in \text{semialg-sets } n$   
 <proof>

**lemma** *pullback-is-semialg*:  
**assumes** *is-poly-tuple*  $n \ fs$   
**assumes**  $\text{length } fs = k$   
**assumes**  $S \in \text{semialg-sets } k$   
**shows** *is-semialgebraic*  $n \ (\text{poly-map } n \ fs \ ^{-1}_n S)$   
 <proof>

Equality and inequality sets for a pair of polynomials

**definition** *val-ineq-set* **where**  
 $\text{val-ineq-set } n \ f \ g = \{x \in \text{carrier } (Q_p^n). \text{val } (Qp\text{-ev } f \ x) \leq \text{val } (Qp\text{-ev } g \ x)\}$

**lemma** *poly-map-length* :  
**assumes**  $\text{length } fs = m$   
**assumes**  $as \in \text{carrier } (Q_p^n)$   
**shows**  $\text{length } (\text{poly-map } n \ fs \ as) = m$   
 <proof>

**lemma** *val-ineq-set-pullback*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{val-ineq-set } n \ f \ g = \text{poly-map } n \ [g, f] \ ^{-1}_n \text{val-relation-set}$   
 <proof>

**lemma** *val-ineq-set-is-semialg*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{val-ineq-set } n \ f \ g \in \text{semialg-sets } n$   
 <proof>

**lemma** *val-ineq-set-is-semialgebraic*:

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \text{ (val-ineq-set } n \text{ } f \text{ } g)$   
 $\langle \text{proof} \rangle$

**lemma** *val-ineq-setI*:

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $x \in (\text{val-ineq-set } n \text{ } f \text{ } g)$   
**shows**  $x \in \text{carrier } (Q_p^n)$   
 $\text{val } (Qp\text{-ev } f \text{ } x) \leq \text{val } (Qp\text{-ev } g \text{ } x)$   
 $\langle \text{proof} \rangle$

**lemma** *val-ineq-setE*:

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**assumes**  $\text{val } (Qp\text{-ev } f \text{ } x) \leq \text{val } (Qp\text{-ev } g \text{ } x)$   
**shows**  $x \in (\text{val-ineq-set } n \text{ } f \text{ } g)$   
 $\langle \text{proof} \rangle$

**lemma** *val-ineq-set-is-semialgebraic'*:

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier } (Q_p^n). \text{val } (Qp\text{-ev } f \text{ } x) \leq \text{val } (Qp\text{-ev } g \text{ } x)\}$   
 $\langle \text{proof} \rangle$

**lemma** *val-eq-set-is-semialgebraic*:

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier } (Q_p^n). \text{val } (Qp\text{-ev } f \text{ } x) = \text{val } (Qp\text{-ev } g \text{ } x)\}$   
 $\langle \text{proof} \rangle$

**lemma** *equalityI''*:

**assumes**  $\bigwedge x. A \text{ } x \implies B \text{ } x$   
**assumes**  $\bigwedge x. B \text{ } x \implies A \text{ } x$   
**shows**  $\{x. A \text{ } x\} = \{x. B \text{ } x\}$   
 $\langle \text{proof} \rangle$

**lemma** *val-strict-ineq-set-is-semialgebraic*:

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier } (Q_p^n). \text{val } (Qp\text{-ev } f \text{ } x) < \text{val } (Qp\text{-ev } g \text{ } x)\}$   
 $\langle \text{proof} \rangle$

**lemma** *constant-poly-val-exists*:

**shows**  $\exists g \in \text{carrier } (Q_p[\mathcal{X}_n]). (\forall x \in \text{carrier } (Q_p^n). \text{val } (Qp\text{-ev } g \text{ } x) = c)$   
 $\langle \text{proof} \rangle$

**lemma** *val-ineq-set-is-semialgebraic''*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier } (Q_p^n). \text{val } (Qp\text{-ev } f \ x) \leq c\}$   
 $\langle \text{proof} \rangle$

**lemma** *val-ineq-set-is-semialgebraic'''*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier } (Q_p^n). c \leq \text{val } (Qp\text{-ev } f \ x)\}$   
 $\langle \text{proof} \rangle$

**lemma** *val-eq-set-is-semialgebraic'*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier } (Q_p^n). \text{val } (Qp\text{-ev } f \ x) = c\}$   
 $\langle \text{proof} \rangle$

**lemma** *val-strict-ineq-set-is-semialgebraic'*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier } (Q_p^n). \text{val } (Qp\text{-ev } f \ x) < c\}$   
 $\langle \text{proof} \rangle$

**lemma** *val-strict-ineq-set-is-semialgebraic''*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier } (Q_p^n). c < \text{val } (Qp\text{-ev } f \ x)\}$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *cring*) *R1-memE*:  
**assumes**  $x \in \text{carrier } (R^1)$   
**shows**  $x = [(\text{hd } x)]$   
 $\langle \text{proof} \rangle$

**lemma**(**in** *cring*) *R1-memE'*:  
**assumes**  $x \in \text{carrier } (R^1)$   
**shows**  $\text{hd } x \in \text{carrier } R$   
 $\langle \text{proof} \rangle$

**lemma** *univ-val-ineq-set-is-univ-semialgebraic*:  
*is-univ-semialgebraic*  $\{x \in \text{carrier } Q_p. \text{val } x \leq c\}$   
 $\langle \text{proof} \rangle$

**lemma** *univ-val-strict-ineq-set-is-univ-semialgebraic*:  
*is-univ-semialgebraic*  $\{x \in \text{carrier } Q_p. \text{val } x < c\}$   
 $\langle \text{proof} \rangle$

**lemma** *univ-val-eq-set-is-univ-semialgebraic*:  
*is-univ-semialgebraic*  $\{x \in \text{carrier } Q_p. \text{val } x = c\}$   
 $\langle \text{proof} \rangle$

### 13.5.7 One Dimensional $p$ -adic Balls are Semialgebraic

**lemma** *coord-ring-one-def*:  
*Pring*  $Q_p \{(0::nat)\} = (Q_p[\mathcal{X}_1])$   
 ⟨*proof*⟩

**lemma** *times-p-pow-val*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b = p[\wedge]n \otimes a$   
**shows**  $\text{val } b = \text{val } a + n$   
 ⟨*proof*⟩

**lemma** *times-p-pow-neg-val*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b = p[\wedge]-n \otimes a$   
**shows**  $\text{val } b = \text{val } a - n$   
 ⟨*proof*⟩

**lemma** *eint-minus-int-pos*:  
**assumes**  $a - \text{eint } n \geq 0$   
**shows**  $a \geq n$   
 ⟨*proof*⟩

$p$ -adic balls as pullbacks of polynomial maps

**lemma** *balls-as-pullbacks*:  
**assumes**  $c \in \text{carrier } Q_p$   
**shows**  $\exists P \in \text{carrier } (Q_p[\mathcal{X}_1]). \text{to-}R1' B_n[c] = \text{poly-map } 1 [P]^{-1}_I (\text{to-}R1' \mathcal{O}_p)$   
 ⟨*proof*⟩

**lemma** *ball-is-semialgebraic*:  
**assumes**  $c \in \text{carrier } Q_p$   
**shows** *is-semialgebraic*  $1 (\text{to-}R1' B_n[c])$   
 ⟨*proof*⟩

**lemma** *ball-is-univ-semialgebraic*:  
**assumes**  $c \in \text{carrier } Q_p$   
**shows** *is-univ-semialgebraic*  $(B_n[c])$   
 ⟨*proof*⟩

**abbreviation** *Qp-to-R1-set* **where**  
*Qp-to-R1-set*  $S \equiv \text{to-}R1' S$

### 13.5.8 Finite Unions and Intersections of Semialgebraic Sets

**definition** *are-semialgebraic* **where**  
*are-semialgebraic*  $n Xs = (\forall x. x \in Xs \longrightarrow \text{is-semialgebraic } n x)$

**lemma** *are-semialgebraicI*:  
**assumes**  $\bigwedge x. x \in Xs \implies \text{is-semialgebraic } n x$   
**shows** *are-semialgebraic*  $n Xs$

*<proof>*

**lemma** *are-semialgebraicE*:

**assumes** *are-semialgebraic n Xs*

**assumes**  $x \in Xs$

**shows** *is-semialgebraic n x*

*<proof>*

**definition** *are-univ-semialgebraic where*

*are-univ-semialgebraic Xs = ( $\forall x. x \in Xs \longrightarrow is-univ-semialgebraic x$ )*

**lemma** *are-univ-semialgebraicI*:

**assumes**  $\bigwedge x. x \in Xs \implies is-univ-semialgebraic x$

**shows** *are-univ-semialgebraic Xs*

*<proof>*

**lemma** *are-univ-semialgebraicE*:

**assumes** *are-univ-semialgebraic Xs*

**assumes**  $x \in Xs$

**shows** *is-univ-semialgebraic x*

*<proof>*

**lemma** *are-univ-semialgebraic-semialgebraic*:

**assumes** *are-univ-semialgebraic Xs*

**shows** *are-semialgebraic 1 (Qp-to-R1-set ' Xs)*

*<proof>*

**lemma** *to-R1-set-union*:

*to-R1 ' ( $\bigcup Xs$ ) =  $\bigcup$  (Qp-to-R1-set ' Xs)*

*<proof>*

**lemma** *to-R1-inter*:

**assumes**  $Xs \neq \{\}$

**shows** *to-R1 ' ( $\bigcap Xs$ ) =  $\bigcap$  (Qp-to-R1-set ' Xs)*

*<proof>*

**lemma** *finite-union-is-semialgebraic*:

**assumes** *finite Xs*

**shows**  $Xs \subseteq semialg-sets n \longrightarrow is-semialgebraic n (\bigcup Xs)$

*<proof>*

**lemma** *finite-union-is-semialgebraic'*:

**assumes** *finite Xs*

**assumes**  $Xs \subseteq semialg-sets n$

**shows** *is-semialgebraic n ( $\bigcup Xs$ )*

*<proof>*

**lemma**(in *padic-fields*) *finite-union-is-semialgebraic''*:

**assumes** *finite S*

**assumes**  $\bigwedge x. x \in S \implies \text{is-semialgebraic } m (F x)$   
**shows**  $\text{is-semialgebraic } m (\bigcup x \in S. F x)$   
 ⟨proof⟩

**lemma** *finite-union-is-univ-semialgebraic'*:

**assumes** *finite*  $Xs$   
**assumes** *are-univ-semialgebraic*  $Xs$   
**shows** *is-univ-semialgebraic*  $(\bigcup Xs)$   
 ⟨proof⟩

**lemma** *finite-intersection-is-semialgebraic*:

**assumes** *finite*  $Xs$   
**shows**  $Xs \subseteq \text{semialg-sets } n \wedge Xs \neq \{\}$   $\longrightarrow$  *is-semialgebraic*  $n (\bigcap Xs)$   
 ⟨proof⟩

**lemma** *finite-intersection-is-semialgebraic'*:

**assumes** *finite*  $Xs$   
**assumes**  $Xs \subseteq \text{semialg-sets } n \wedge Xs \neq \{\}$   
**shows** *is-semialgebraic*  $n (\bigcap Xs)$   
 ⟨proof⟩

**lemma** *finite-intersection-is-semialgebraic''*:

**assumes** *finite*  $Xs$   
**assumes** *are-semialgebraic*  $n Xs \wedge Xs \neq \{\}$   
**shows** *is-semialgebraic*  $n (\bigcap Xs)$   
 ⟨proof⟩

**lemma** *finite-intersection-is-univ-semialgebraic*:

**assumes** *finite*  $Xs$   
**assumes** *are-univ-semialgebraic*  $Xs$   
**assumes**  $Xs \neq \{\}$   
**shows** *is-univ-semialgebraic*  $(\bigcap Xs)$   
 ⟨proof⟩

## 13.6 Cartesian Products of Semialgebraic Sets

**lemma** *Qp-times-basic-semialg-right*:

**assumes**  $a \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows** *cartesian-product (basic-semialg-set*  $n k a)$   $(\text{carrier } (Q_p^m)) = \text{basic-semialg-set}$   
 $(n + m) k a$   
 ⟨proof⟩

**lemma** *Qp-times-basic-semialg-right-is-semialgebraic*:

**assumes**  $k > 0$   
**assumes**  $a \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows** *is-semialgebraic*  $(n + m)$   $(\text{cartesian-product (basic-semialg-set } n k a)$   
 $(\text{carrier } (Q_p^m)))$   
 ⟨proof⟩

**lemma** *Qp-times-basic-semialg-right-is-semialgebraic'*:  
**assumes**  $A \in \text{basic-semialgs } n$   
**shows** *is-semialgebraic*  $(n + m)$  (*cartesian-product*  $A$  (*carrier*  $(Q_p^m)$ ))  
 $\langle \text{proof} \rangle$

**lemma** *cartesian-product-memE'*:  
**assumes**  $x \in \text{cartesian-product } A B$   
**obtains**  $a b$  **where**  $a \in A \wedge b \in B \wedge x = a@b$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-times-basic-semialg-left*:  
**assumes**  $a \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows** *cartesian-product* (*carrier*  $(Q_p^m)$ ) (*basic-semialg-set*  $n k a$ ) = *basic-semialg-set*  $(n+m) k$  (*shift-vars*  $n m a$ )  
 $\langle \text{proof} \rangle$

**lemma** *Qp-times-basic-semialg-left-is-semialgebraic*:  
**assumes**  $k > 0$   
**assumes**  $a \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows** *is-semialgebraic*  $(n + m)$  (*cartesian-product* (*carrier*  $(Q_p^m)$ ) (*basic-semialg-set*  $n k a$ ))  
 $\langle \text{proof} \rangle$

**lemma** *Qp-times-basic-semialg-left-is-semialgebraic'*:  
**assumes**  $A \in \text{basic-semialgs } n$   
**shows** *is-semialgebraic*  $(n + m)$  (*cartesian-product* (*carrier*  $(Q_p^m)$ )  $A$ )  
 $\langle \text{proof} \rangle$

**lemma** *product-of-basic-semialgs-is-semialg*:  
**assumes**  $k > 0$   
**assumes**  $l > 0$   
**assumes**  $a \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $b \in \text{carrier } (Q_p[\mathcal{X}_m])$   
**shows** *is-semialgebraic*  $(n + m)$  (*cartesian-product* (*basic-semialg-set*  $n k a$ ) (*basic-semialg-set*  $m l b$ ))  
 $\langle \text{proof} \rangle$

**lemma** *product-of-basic-semialgs-is-semialg'*:  
**assumes**  $A \in (\text{basic-semialgs } n)$   
**assumes**  $B \in (\text{basic-semialgs } m)$   
**shows** *is-semialgebraic*  $(n + m)$  (*cartesian-product*  $A B$ )  
 $\langle \text{proof} \rangle$

**lemma** *car-times-semialg-is-semialg*:  
**assumes** *is-semialgebraic*  $m B$   
**shows** *is-semialgebraic*  $(n + m)$  (*cartesian-product* (*carrier*  $(Q_p^n)$ )  $B$ )  
 $\langle \text{proof} \rangle$

**lemma** *basic-semialg-times-semialg-is-semialg*:

**assumes**  $A \in \text{basic-semialgs } n$   
**assumes**  $\text{is-semialgebraic } m B$   
**shows**  $\text{is-semialgebraic } (n + m)$  (*cartesian-product*  $A B$ )  
 ⟨*proof*⟩

Semialgebraic sets are closed under cartesian products

**lemma** *cartesian-product-is-semialgebraic*:  
**assumes**  $\text{is-semialgebraic } n A$   
**assumes**  $\text{is-semialgebraic } m B$   
**shows**  $\text{is-semialgebraic } (n + m)$  (*cartesian-product*  $A B$ )  
 ⟨*proof*⟩

### 13.7 $N^{\text{th}}$ Power Residues

**definition** *nth-root-poly where*  
 $\text{nth-root-poly } (n::\text{nat}) a = ((X\text{-poly } Q_p) [\uparrow]_{Q_p-x} n) \ominus_{Q_p-x} (\text{to-poly } a)$

**lemma** *nth-root-poly-closed*:  
**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $\text{nth-root-poly } n a \in \text{carrier } Q_p-x$   
 ⟨*proof*⟩

**lemma** *nth-root-poly-eval*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**shows**  $(\text{nth-root-poly } n a) \cdot b = (b[\uparrow]n) \ominus_{Q_p} a$   
 ⟨*proof*⟩

Hensel's lemma gives us this criterion for the existence of  $n$ -th roots

**lemma** *nth-root-poly-root*:  
**assumes**  $(n::\text{nat}) > 1$   
**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $a \neq \mathbf{1}$   
**assumes**  $\text{val } (\mathbf{1} \ominus_{Q_p} a) > 2 * \text{val } ([n] \cdot \mathbf{1})$   
**shows**  $(\exists b \in \mathcal{O}_p. (b[\uparrow]n) = a)$   
 ⟨*proof*⟩

All points sufficiently close to 1 have  $n$ th roots

**lemma** *eint-nat-times-2*:  
 $2*(n::\text{nat}) = 2*\text{eint } n$   
 ⟨*proof*⟩

**lemma** *P-set-of-one*:  
 $P\text{-set } 1 = \text{nonzero } Q_p$   
 ⟨*proof*⟩

**lemma** *nth-power-fact*:  
**assumes**  $(n::\text{nat}) \geq 1$

**shows**  $\exists (m::nat) > 0. \forall u \in \text{carrier } Q_p. \text{ac } m \ u = 1 \wedge \text{val } u = 0 \longrightarrow u \in P\text{-set}$   
 $n$   
 $\langle \text{proof} \rangle$

**definition** *pow-res where*  
 $\text{pow-res } (n::nat) \ x = \{a. a \in \text{carrier } Q_p \wedge (\exists y \in \text{nonzero } Q_p. (a = x \otimes (y[\uparrow n])))\}$

**lemma** *nonzero-pow-res:*  
**assumes**  $x \in \text{nonzero } Q_p$   
**shows**  $\text{pow-res } (n::nat) \ x \subseteq \text{nonzero } Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-of-zero:*  
**shows**  $\text{pow-res } n \ \mathbf{0} = \{\mathbf{0}\}$   
 $\langle \text{proof} \rangle$

**lemma** *equal-pow-resI:*  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $y \in \text{pow-res } n \ x$   
**shows**  $\text{pow-res } n \ x = \text{pow-res } n \ y$   
 $\langle \text{proof} \rangle$

**lemma** *zeroth-pow-res:*  
**assumes**  $x \in \text{carrier } Q_p$   
**shows**  $\text{pow-res } 0 \ x = \{x\}$   
 $\langle \text{proof} \rangle$

**lemma** *Zp-car-zero-res:* **assumes**  $x \in \text{carrier } Z_p$   
**shows**  $x \ 0 = 0$   
 $\langle \text{proof} \rangle$

**lemma** *zeroth-ac:*  
**assumes**  $x \in \text{carrier } Q_p$   
**shows**  $\text{ac } 0 \ x = 0$   
 $\langle \text{proof} \rangle$

**lemma** *nonzero-ac-imp-nonzero:*  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $\text{ac } m \ x \neq 0$   
**shows**  $x \in \text{nonzero } Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *nonzero-ac-val-ord:*  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $\text{ac } m \ x \neq 0$   
**shows**  $\text{val } x = \text{ord } x$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-equal-ord:*

**assumes**  $n > 0$   
**shows**  $\exists m > 0. \forall x y. x \in \text{nonzero } Q_p \wedge y \in \text{nonzero } Q_p \wedge ac\ m\ x = ac\ m\ y$   
 $\wedge \text{ord } x = \text{ord } y \longrightarrow \text{pow-res } n\ x = \text{pow-res } n\ y$   
 <proof>

**lemma** *pow-res-equal*:

**assumes**  $n > 0$   
**shows**  $\exists m > 0. \forall x y. x \in \text{nonzero } Q_p \wedge y \in \text{nonzero } Q_p \wedge ac\ m\ x = ac\ m\ y \wedge$   
 $\text{ord } x = (\text{ord } y \text{ mod } n) \longrightarrow \text{pow-res } n\ x = \text{pow-res } n\ y$   
 <proof>

**definition** *pow-res-classes where*

*pow-res-classes*  $n = \{S. \exists x \in \text{nonzero } Q_p. S = \text{pow-res } n\ x\}$

**lemma** *pow-res-semialg-def*:

**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $n \geq 1$   
**shows**  $\exists P \in \text{carrier } Q_p\text{-}x. \text{pow-res } n\ x = (\text{univ-basic-semialg-set } n\ P) - \{0\}$   
 <proof>

**lemma** *pow-res-is-univ-semialgebraic*:

**assumes**  $x \in \text{carrier } Q_p$   
**shows** *is-univ-semialgebraic* ( $\text{pow-res } n\ x$ )  
 <proof>

**lemma** *pow-res-is-semialg*:

**assumes**  $x \in \text{carrier } Q_p$   
**shows** *is-semialgebraic 1* (*to-R1* ' ( $\text{pow-res } n\ x$ ))  
 <proof>

**lemma** *pow-res-refl*:

**assumes**  $x \in \text{carrier } Q_p$   
**shows**  $x \in \text{pow-res } n\ x$   
 <proof>

**lemma** *equal-pow-resE*:

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $n > 0$   
**assumes**  $\text{pow-res } n\ a = \text{pow-res } n\ b$   
**shows**  $\exists s \in P\text{-set } n. a = b \otimes s$   
 <proof>

**lemma** *pow-res-one*:

**assumes**  $x \in \text{nonzero } Q_p$   
**shows**  $\text{pow-res } 1\ x = \text{nonzero } Q_p$   
 <proof>

**lemma** *pow-res-zero*:  
 assumes  $n > 0$   
 shows  $\text{pow-res } n \ \mathbf{0} = \{\mathbf{0}\}$   
 ⟨proof⟩

**lemma** *equal-pow-resI'*:  
 assumes  $a \in \text{carrier } Q_p$   
 assumes  $b \in \text{carrier } Q_p$   
 assumes  $c \in P\text{-set } n$   
 assumes  $a = b \otimes c$   
 assumes  $n > 0$   
 shows  $\text{pow-res } n \ a = \text{pow-res } n \ b$   
 ⟨proof⟩

**lemma** *equal-pow-resI''*:  
 assumes  $n > 0$   
 assumes  $a \in \text{nonzero } Q_p$   
 assumes  $b \in \text{nonzero } Q_p$   
 assumes  $a \otimes \text{inv } b \in P\text{-set } n$   
 shows  $\text{pow-res } n \ a = \text{pow-res } n \ b$   
 ⟨proof⟩

**lemma** *equal-pow-resI'''*:  
 assumes  $n > 0$   
 assumes  $a \in \text{nonzero } Q_p$   
 assumes  $b \in \text{nonzero } Q_p$   
 assumes  $c \in \text{nonzero } Q_p$   
 assumes  $\text{pow-res } n \ (c \otimes a) = \text{pow-res } n \ (c \otimes b)$   
 shows  $\text{pow-res } n \ a = \text{pow-res } n \ b$   
 ⟨proof⟩

**lemma** *equal-pow-resI''''*:  
 assumes  $n > 0$   
 assumes  $a \in \text{carrier } Q_p$   
 assumes  $b \in \text{carrier } Q_p$   
 assumes  $a = b \otimes u$   
 assumes  $u \in P\text{-set } n$   
 shows  $\text{pow-res } n \ a = \text{pow-res } n \ b$   
 ⟨proof⟩

**lemma** *Zp-Units-ord-zero*:  
 assumes  $a \in \text{Units } Z_p$   
 shows  $\text{ord-}Z_p \ a = 0$   
 ⟨proof⟩

**lemma** *pow-res-nth-pow*:  
 assumes  $a \in \text{nonzero } Q_p$

**assumes**  $n > 0$   
**shows**  $\text{pow-res } n \ (a[\wedge]n) = \text{pow-res } n \ \mathbf{1}$   
 <proof>

**lemma** *pow-res-of-p-pow*:  
**assumes**  $n > 0$   
**shows**  $\text{pow-res } n \ (\mathfrak{p}[\wedge]((l::\text{int})*n)) = \text{pow-res } n \ \mathbf{1}$   
 <proof>

**lemma** *pow-res-nonzero*:  
**assumes**  $n > 0$   
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $\text{pow-res } n \ a = \text{pow-res } n \ b$   
**shows**  $b \in \text{nonzero } Q_p$   
 <proof>

**lemma** *pow-res-mult*:  
**assumes**  $n > 0$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $d \in \text{carrier } Q_p$   
**assumes**  $\text{pow-res } n \ a = \text{pow-res } n \ c$   
**assumes**  $\text{pow-res } n \ b = \text{pow-res } n \ d$   
**shows**  $\text{pow-res } n \ (a \otimes b) = \text{pow-res } n \ (c \otimes d)$   
 <proof>

**lemma** *pow-res-p-pow-factor*:  
**assumes**  $n > 0$   
**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $\text{pow-res } n \ a = \text{pow-res } n \ (\mathfrak{p}[\wedge]((l::\text{int})*n) \otimes a)$   
 <proof>

**lemma** *pow-res-classes-finite*:  
**assumes**  $n \geq 1$   
**shows** *finite* ( $\text{pow-res-classes } n$ )  
 <proof>

**lemma** *pow-res-classes-univ-semialg*:  
**assumes**  $S \in \text{pow-res-classes } n$   
**shows** *is-univ-semialgebraic*  $S$   
 <proof>

**lemma** *pow-res-classes-semialg*:  
**assumes**  $S \in \text{pow-res-classes } n$   
**shows** *is-semialgebraic 1* ( $\text{to-}R1' \ S$ )  
 <proof>

**definition** *nth-pow-wits* **where**

*nth-pow-wits*  $n = (\lambda S. (\text{SOME } x. x \in (S \cap \mathcal{O}_p)))' (\text{pow-res-classes } n)$

**lemma** *nth-pow-wits-finite*:

**assumes**  $n > 0$

**shows** *finite* (*nth-pow-wits*  $n$ )

*<proof>*

**lemma** *nth-pow-wits-covers*:

**assumes**  $n > 0$

**assumes**  $x \in \text{nonzero } \mathcal{Q}_p$

**shows**  $\exists y \in (\text{nth-pow-wits } n). y \in \text{nonzero } \mathcal{Q}_p \wedge y \in \mathcal{O}_p \wedge x \in \text{pow-res } n \ y$

*<proof>*

**lemma** *nth-pow-wits-closed*:

**assumes**  $n > 0$

**assumes**  $x \in \text{nth-pow-wits } n$

**shows**  $x \in \text{carrier } \mathcal{Q}_p \ x \in \mathcal{O}_p \ x \in \text{nonzero } \mathcal{Q}_p \ \exists y \in \text{pow-res-classes } n. y = \text{pow-res } n \ x$

*<proof>*

**lemma** *finite-extensional-funcset*:

**assumes** *finite*  $A$

**assumes** *finite* ( $B::'b \text{ set}$ )

**shows** *finite* ( $A \rightarrow_E B$ )

*<proof>*

**lemma** *nth-pow-wits-exists*:

**assumes**  $m > 0$

**assumes**  $c \in \text{pow-res-classes } m$

**shows**  $\exists x. x \in c \cap \mathcal{O}_p$

*<proof>*

**lemma** *pow-res-classes-mem-eq*:

**assumes**  $m > 0$

**assumes**  $a \in \text{pow-res-classes } m$

**assumes**  $x \in a$

**shows**  $a = \text{pow-res } m \ x$

*<proof>*

**lemma** *nth-pow-wits-neq-pow-res*:

**assumes**  $m > 0$

**assumes**  $x \in \text{nth-pow-wits } m$

**assumes**  $y \in \text{nth-pow-wits } m$

**assumes**  $x \neq y$

**shows**  $\text{pow-res } m \ x \neq \text{pow-res } m \ y$

*<proof>*

**lemma** *nth-pow-wits-disjoint-pow-res*:

**assumes**  $m > 0$   
**assumes**  $x \in \text{nth-pow-wits } m$   
**assumes**  $y \in \text{nth-pow-wits } m$   
**assumes**  $x \neq y$   
**shows**  $\text{pow-res } m \ x \cap \text{pow-res } m \ y = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *nth-power-fact'*:  
**assumes**  $0 < (n::\text{nat})$   
**shows**  $\exists m > 0. \forall u \in \text{carrier } Q_p. \text{ac } m \ u = 1 \wedge \text{val } u = 0 \longrightarrow u \in P\text{-set } n$   
 $\langle \text{proof} \rangle$

**lemma** *equal-pow-res-criterion*:  
**assumes**  $N > 0$   
**assumes**  $n > 0$   
**assumes**  $\forall u \in \text{carrier } Q_p. \text{ac } N \ u = 1 \wedge \text{val } u = 0 \longrightarrow u \in P\text{-set } n$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $a = b \otimes (\mathbf{1} \oplus c)$   
**assumes**  $\text{val } c \geq N$   
**shows**  $\text{pow-res } n \ a = \text{pow-res } n \ b$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-nat-pow*:  
**assumes**  $n > 0$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $\text{pow-res } n \ a = \text{pow-res } n \ b$   
**shows**  $\text{pow-res } n \ (a[\uparrow](k::\text{nat})) = \text{pow-res } n \ (b[\uparrow]k)$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-mult'*:  
**assumes**  $n > 0$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{carrier } Q_p$   
**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $d \in \text{carrier } Q_p$   
**assumes**  $e \in \text{carrier } Q_p$   
**assumes**  $f \in \text{carrier } Q_p$   
**assumes**  $\text{pow-res } n \ a = \text{pow-res } n \ d$   
**assumes**  $\text{pow-res } n \ b = \text{pow-res } n \ e$   
**assumes**  $\text{pow-res } n \ c = \text{pow-res } n \ f$   
**shows**  $\text{pow-res } n \ (a \otimes b \otimes c) = \text{pow-res } n \ (d \otimes e \otimes f)$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-disjoint*:

**assumes**  $n > 0$   
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $a \notin \text{pow-res } n \mathbf{1}$   
**shows**  $\neg (\exists y \in \text{nonzero } Q_p. a = y[\wedge]n)$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-disjoint'*:

**assumes**  $n > 0$   
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $\text{pow-res } n a \neq \text{pow-res } n \mathbf{1}$   
**shows**  $\neg (\exists y \in \text{nonzero } Q_p. a = y[\wedge]n)$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-one-imp-nth-pow*:

**assumes**  $n > 0$   
**assumes**  $a \in \text{pow-res } n \mathbf{1}$   
**shows**  $\exists y \in \text{nonzero } Q_p. a = y[\wedge]n$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-eq*:

**assumes**  $n > 0$   
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $b \in \text{pow-res } n a$   
**shows**  $\text{pow-res } n b = \text{pow-res } n a$   
 $\langle \text{proof} \rangle$

**lemma** *pow-res-classes-n-eq-one*:

*pow-res-classes*  $1 = \{\text{nonzero } Q_p\}$   
 $\langle \text{proof} \rangle$

**lemma** *nth-pow-wits-closed'*:

**assumes**  $n > 0$   
**assumes**  $x \in \text{nth-pow-wits } n$   
**shows**  $x \in \mathcal{O}_p \wedge x \in \text{nonzero } Q_p$   $\langle \text{proof} \rangle$

## 13.8 Semialgebraic Sets Defined by Congruences

### 13.8.1 $p$ -adic ord Congruence Sets

**lemma** *carrier-is-univ-semialgebraic*:

*is-univ-semialgebraic* (*carrier*  $Q_p$ )  
 $\langle \text{proof} \rangle$

**lemma** *nonzero-is-univ-semialgebraic*:

*is-univ-semialgebraic* (*nonzero*  $Q_p$ )  
 $\langle \text{proof} \rangle$

**definition** *ord-congruence-set* **where**

*ord-congruence-set*  $n a = \{x \in \text{nonzero } Q_p. \text{ord } x \bmod n = a\}$

**lemma** *ord-congruence-set-nonzero*:  
*ord-congruence-set*  $n$   $a \subseteq \text{nonzero } Q_p$   
 ⟨proof⟩

**lemma** *ord-congruence-set-closed*:  
*ord-congruence-set*  $n$   $a \subseteq \text{carrier } Q_p$   
 ⟨proof⟩

**lemma** *ord-congruence-set-memE*:  
**assumes**  $x \in \text{ord-congruence-set } n$   $a$   
**shows**  $x \in \text{nonzero } Q_p$   
 $\text{ord } x \text{ mod } n = a$   
 ⟨proof⟩

**lemma** *ord-congruence-set-memI*:  
**assumes**  $x \in \text{nonzero } Q_p$   
**assumes**  $\text{ord } x \text{ mod } n = a$   
**shows**  $x \in \text{ord-congruence-set } n$   $a$   
 ⟨proof⟩

We want to prove that *ord\_congruence\_set* is a finite union of semialgebraic sets, hence is also semialgebraic.

**lemma** *pow-res-ord-cong*:  
**assumes**  $x \in \text{carrier } Q_p$   
**assumes**  $x \in \text{ord-congruence-set } n$   $a$   
**shows**  $\text{pow-res } n$   $x \subseteq \text{ord-congruence-set } n$   $a$   
 ⟨proof⟩

**lemma** *pow-res-classes-are-univ-semialgebraic*:  
**shows** *are-univ-semialgebraic* (*pow-res-classes*  $n$ )  
 ⟨proof⟩

**lemma** *ord-congruence-set-univ-semialg*:  
**assumes**  $n \geq 0$   
**shows** *is-univ-semialgebraic* (*ord-congruence-set*  $n$   $a$ )  
 ⟨proof⟩

**lemma** *ord-congruence-set-is-semialg*:  
**assumes**  $n \geq 0$   
**shows** *is-semialgebraic 1* (*Qp-to-R1-set* (*ord-congruence-set*  $n$   $a$ ))  
 ⟨proof⟩

### 13.8.2 Congruence Sets for the order of the Evaluation of a Polynomial

**lemma** *poly-map-singleton*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows** *poly-map*  $n$   $[f]$   $x = [(Qp-ev f x)]$

*<proof>*

**definition** *poly-cong-set* **where**

*poly-cong-set*  $n f m a = \{x \in \text{carrier } (Q_p^n). (Qp\text{-ev } f x) \neq \mathbf{0} \wedge (\text{ord } (Qp\text{-ev } f x) \text{ mod } m = a)\}$

**lemma** *poly-cong-set-as-pullback*:

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$

**shows** *poly-cong-set*  $n f m a = \text{poly-map } n [f]^{-1}_n(Qp\text{-to-R1-set } (\text{ord-congruence-set } m a))$

*<proof>*

**lemma** *singleton-poly-tuple*:

*is-poly-tuple*  $n [f] \longleftrightarrow f \in \text{carrier } (Q_p[\mathcal{X}_n])$

*<proof>*

**lemma** *poly-cong-set-is-semialgebraic*:

**assumes**  $m \geq 0$

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$

**shows** *is-semialgebraic*  $n$  (*poly-cong-set*  $n f m a$ )

*<proof>*

### 13.8.3 Congruence Sets for Angular Components

If a set is a union of  $n$ -th power residues, then it is semialgebraic.

**lemma** *pow-res-union-imp-semialg*:

**assumes**  $n \geq 1$

**assumes**  $S \subseteq \text{nonzero } Q_p$

**assumes**  $\bigwedge x. x \in S \implies \text{pow-res } n x \subseteq S$

**shows** *is-univ-semialgebraic*  $S$

*<proof>*

**definition** *ac-cong-set1* **where**

*ac-cong-set1*  $n y = \{x \in \text{carrier } Q_p. x \neq \mathbf{0} \wedge \text{ac } n x = \text{ac } n y\}$

**lemma** *ac-cong-set1-is-univ-semialg*:

**assumes**  $n > 0$

**assumes**  $b \in \text{nonzero } Q_p$

**assumes**  $b \in \mathcal{O}_p$

**shows** *is-univ-semialgebraic* (*ac-cong-set1*  $n b$ )

*<proof>*

**definition** *ac-cong-set* **where**

*ac-cong-set*  $n k = \{x \in \text{carrier } Q_p. x \neq \mathbf{0} \wedge \text{ac } n x = k\}$

**lemma** *ac-cong-set-is-univ-semialg*:

**assumes**  $n > 0$

**assumes**  $k \in \text{Units } (Zp\text{-res-ring } n)$

**shows** *is-univ-semialgebraic* (*ac-cong-set*  $n k$ )

*<proof>*

**definition** *val-ring-constant-ac-set* **where**

*val-ring-constant-ac-set*  $n\ k = \{a \in \mathcal{O}_p. \text{val } a = 0 \wedge \text{ac } n\ a = k\}$

**lemma** *val-nonzero'*:

**assumes**  $a \in \text{carrier } Q_p$

**assumes**  $\text{val } a = \text{eint } k$

**shows**  $a \in \text{nonzero } Q_p$

*<proof>*

**lemma** *val-ord'*:

**assumes**  $a \in \text{carrier } Q_p$

**assumes**  $a \neq 0$

**shows**  $\text{val } a = \text{ord } a$

*<proof>*

**lemma** *val-ring-constant-ac-set-is-univ-semialgebraic*:

**assumes**  $n > 0$

**assumes**  $k \neq 0$

**shows** *is-univ-semialgebraic* (*val-ring-constant-ac-set*  $n\ k$ )

*<proof>*

**definition** *val-ring-constant-ac-sets* **where**

*val-ring-constant-ac-sets*  $n = \text{val-ring-constant-ac-set } n\ '(\text{Units } (Zp\text{-res-ring } n))$

**lemma** *val-ring-constant-ac-sets-are-univ-semialgebraic*:

**assumes**  $n > 0$

**shows** *are-univ-semialgebraic* (*val-ring-constant-ac-sets*  $n$ )

*<proof>*

**definition** *ac-cong-set3* **where**

*ac-cong-set3*  $n = \{as. \exists a\ b. a \in \text{nonzero } Q_p \wedge b \in \mathcal{O}_p \wedge \text{val } b = 0 \wedge (\text{ac } n\ a = \text{ac } n\ b) \wedge as = [a, b]\}$

**definition** *ac-cong-set2* **where**

*ac-cong-set2*  $n\ k = \{as. \exists a\ b. a \in \text{nonzero } Q_p \wedge b \in \mathcal{O}_p \wedge \text{val } b = 0 \wedge (\text{ac } n\ a = k) \wedge (\text{ac } n\ b) = k \wedge as = [a, b]\}$

**lemma** *ac-cong-set2-cartesian-product*:

**assumes**  $k \in \text{Units } (Zp\text{-res-ring } n)$

**assumes**  $n > 0$

**shows** *ac-cong-set2*  $n\ k = \text{cartesian-product } (\text{to-}R1'(\text{ac-cong-set } n\ k)) (\text{to-}R1'(\text{val-ring-constant-ac-set } n\ k))$

*<proof>*

**lemma** *ac-cong-set2-is-semialg*:

**assumes**  $k \in \text{Units } (Zp\text{-res-ring } n)$

**assumes**  $n > 0$

**shows** *is-semialgebraic 2* (*ac-cong-set2*  $n$   $k$ )  
 ⟨*proof*⟩

**lemma** *ac-cong-set3-as-union*:

**assumes**  $n > 0$

**shows** *ac-cong-set3*  $n = \bigcup (ac-cong-set2\ n\ \text{' (Units (Zp-res-ring } n))$  )  
 ⟨*proof*⟩

**lemma** *ac-cong-set3-is-semialgebraic*:

**assumes**  $n > 0$

**shows** *is-semialgebraic 2* (*ac-cong-set3*  $n$ )  
 ⟨*proof*⟩

### 13.9 Permutations of indices of semialgebraic sets

**lemma** *fun-inv-permute*:

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**shows** *fun-inv*  $\sigma$  *permutes*  $\{..<n\}$

$\sigma \circ (\text{fun-inv } \sigma) = \text{id}$

$(\text{fun-inv } \sigma) \circ \sigma = \text{id}$

⟨*proof*⟩

**lemma** *poly-tuple-pullback-eq-poly-map-vimage*:

**assumes** *is-poly-tuple*  $n$   $fs$

**assumes** *length*  $fs = m$

**assumes**  $S \subseteq \text{carrier } (Q_p^m)$

**shows** *poly-map*  $n$   $fs$   $^{-1}_n S = \text{poly-tuple-pullback } n$   $S$   $fs$

⟨*proof*⟩

**lemma** *permutation-is-semialgebraic*:

**assumes** *is-semialgebraic*  $n$   $S$

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**shows** *is-semialgebraic*  $n$  (*permute-list*  $\sigma$  '  $S$ )  
 ⟨*proof*⟩

**lemma** *permute-list-closed*:

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**shows** *permute-list*  $\sigma$   $a \in \text{carrier } (Q_p^n)$

⟨*proof*⟩

**lemma** *permute-list-closed'*:

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**assumes** *permute-list*  $\sigma$   $a \in \text{carrier } (Q_p^n)$

**shows**  $a \in \text{carrier } (Q_p^n)$

⟨*proof*⟩

**lemma** *permute-list-compose-inv*:

**assumes**  $\sigma$  *permutes*  $\{..<n\}$

**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $\text{permute-list } \sigma (\text{permute-list } (\text{fun-inv } \sigma) a) = a$   
 $\text{permute-list } (\text{fun-inv } \sigma) (\text{permute-list } \sigma a) = a$   
 ⟨proof⟩

**lemma** *permutation-is-semialgebraic-imp-is-semialgebraic*:

**assumes**  $\text{is-semialgebraic } n (\text{permute-list } \sigma ' S)$   
**assumes**  $\sigma \text{ permutes } \{..<n\}$   
**shows**  $\text{is-semialgebraic } n S$   
 ⟨proof⟩

**lemma** *split-cartesian-product-is-semialgebraic*:

**assumes**  $i \leq n$   
**assumes**  $\text{is-semialgebraic } n A$   
**assumes**  $\text{is-semialgebraic } m B$   
**shows**  $\text{is-semialgebraic } (n + m) (\text{split-cartesian-product } n m i A B)$   
 ⟨proof⟩

**definition** *reverse-val-relation-set* **where**

$\text{reverse-val-relation-set} = \{as \in \text{carrier } (Q_p^2). \text{val } (as ! 0) \leq \text{val } (as ! 1)\}$

**lemma** *Qp-2-car-memE*:

**assumes**  $x \in \text{carrier } (Q_p^2)$   
**shows**  $x = [x!0, x!1]$   
 ⟨proof⟩

**definition** *flip* **where**

$\text{flip} = (\lambda i::\text{nat}. (\text{if } i = 0 \text{ then } 1 \text{ else } (\text{if } i = 1 \text{ then } 0 \text{ else } i)))$

**lemma** *flip-permutes*:

$\text{flip permutes } \{0,1\}$   
 ⟨proof⟩

**lemma** *flip-eval*:

$\text{flip } 0 = 1$   
 $\text{flip } 1 = 0$   
 ⟨proof⟩

**lemma** *flip-x*:

**assumes**  $x \in \text{carrier } (Q_p^2)$   
**shows**  $\text{permute-list } \text{flip } x = [x!1, x!0]$   
 ⟨proof⟩

**lemma** *permute-with-flip-closed*:

**assumes**  $x \in \text{carrier } (Q_p^{2::\text{nat}})$   
**shows**  $\text{permute-list } \text{flip } x \in \text{carrier } (Q_p^{2::\text{nat}})$   
 ⟨proof⟩

**lemma** *reverse-val-relation-set-semialg*:

*is-semialgebraic 2 reverse-val-relation-set*  
 ⟨proof⟩

**definition** *strict-val-relation-set* **where**  
 $strict\text{-}val\text{-}relation\text{-}set = \{as \in carrier (Q_p^2). val (as ! 0) < val (as ! 1)\}$

**definition** *val-diag* **where**  
 $val\text{-}diag = \{as \in carrier (Q_p^2). val (as ! 0) = val (as ! 1)\}$

**lemma** *val-diag-semialg:*  
*is-semialgebraic 2 val-diag*  
 ⟨proof⟩

**lemma** *strict-val-relation-set-is-semialg:*  
*is-semialgebraic 2 strict-val-relation-set*  
 ⟨proof⟩

**lemma** *singleton-length:*  
 $length [a] = 1$   
 ⟨proof⟩

**lemma** *take-closed':*  
**assumes**  $m > 0$   
**assumes**  $x \in carrier (Q_p^{m+1})$   
**shows**  $take\ m\ x \in carrier (Q_p^m)$   
 ⟨proof⟩

**lemma** *triple-val-ineq-set-semialg:*  
**shows** *is-semialgebraic 3*  $\{as \in carrier (Q_p^3). val (as!0) \leq val (as!1) \wedge val (as!1) \leq val (as!2)\}$   
 ⟨proof⟩

**lemma** *triple-val-ineq-set-semialg':*  
**shows** *is-semialgebraic 3*  $\{as \in carrier (Q_p^3). val (as!0) \leq val (as!1) \wedge val (as!1) < val (as!2)\}$   
 ⟨proof⟩

**lemma** *triple-val-ineq-set-semialg'':*  
**shows** *is-semialgebraic 3*  $\{as \in carrier (Q_p^3). val (as!1) < val (as!2)\}$   
 ⟨proof⟩

**lemma** *triple-val-ineq-set-semialg''':*  
**shows** *is-semialgebraic 3*  $\{as \in carrier (Q_p^3). val (as!1) \leq val (as!2)\}$   
 ⟨proof⟩

### 13.10 Semialgebraic Functions

The most natural way to define a semialgebraic function  $f : \mathbb{Q}_p^n \rightarrow \mathbb{Q}_p$  is a function whose graph is a semialgebraic subset of  $\mathbb{Q}_p^{n+1}$ . However,

the definition given here is slightly different, and devised by Denef in [1] in order to prove Macintyre's theorem. As Denef notes, we can use Macintyre's theorem to deduce that the given definition perfectly aligns with the intuitive one.

### 13.10.1 Defining Semialgebraic Functions

Apply a function  $f$  to the tuple consisting of the first  $n$  indices, leaving the remaining indices unchanged

**definition** *partial-image where*

$$\text{partial-image } m \ f \ xs = (f \ (\text{take } m \ xs)) \# (\text{drop } m \ xs)$$

**definition** *partial-pullback where*

$$\text{partial-pullback } m \ f \ l \ S = (\text{partial-image } m \ f) \ ^{-1}_{m+l} S$$

**lemma** *partial-pullback-memE:*

**assumes**  $as \in \text{carrier } (Q_p^{m+l} S)$

**shows**  $as \in \text{carrier } (Q_p^m \text{ partial-image } m \ f \ as) \in S$

*<proof>*

**lemma** *partial-pullback-closed:*

$$\text{partial-pullback } m \ f \ l \ S \subseteq \text{carrier } (Q_p^{m+l})$$

*<proof>*

**lemma** *partial-pullback-memI:*

**assumes**  $as \in \text{carrier } (Q_p^{m+k})$

**assumes**  $(f \ (\text{take } m \ as)) \# (\text{drop } m \ as) \in S$

**shows**  $as \in \text{partial-pullback } m \ f \ k \ S$

*<proof>*

**lemma** *partial-image-eq:*

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $bs \in \text{carrier } (Q_p^k)$

**assumes**  $x = as \ @ \ bs$

**shows**  $\text{partial-image } n \ f \ x = (f \ as) \# \ bs$

*<proof>*

**lemma** *partial-pullback-memE':*

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $bs \in \text{carrier } (Q_p^k)$

**assumes**  $x = as \ @ \ bs$

**assumes**  $x \in \text{partial-pullback } n \ f \ k \ S$

**shows**  $(f \ as) \# \ bs \in S$

*<proof>*

Partial pullbacks have the same algebraic properties as pullbacks

**lemma** *partial-pullback-intersect:*

$partial\text{-pullback } m f l (S1 \cap S2) = (partial\text{-pullback } m f l S1) \cap (partial\text{-pullback } m f l S2)$

$\langle proof \rangle$

**lemma** *partial-pullback-union:*

$partial\text{-pullback } m f l (S1 \cup S2) = (partial\text{-pullback } m f l S1) \cup (partial\text{-pullback } m f l S2)$

$\langle proof \rangle$

**lemma** *cartesian-power-drop:*

**assumes**  $x \in carrier (Q_p^{n+l})$

**shows**  $drop\ n\ x \in carrier (Q_p^l)$

$\langle proof \rangle$

**lemma** *partial-pullback-complement:*

**assumes**  $f \in carrier (Q_p^m) \rightarrow carrier\ Q_p$

**shows**  $partial\text{-pullback } m f l (carrier (Q_p^{Suc\ l}) - S) = carrier (Q_p^{m+l}) - (partial\text{-pullback } m f l S)$

$\langle proof \rangle$

**lemma** *partial-pullback-carrier:*

**assumes**  $f \in carrier (Q_p^m) \rightarrow carrier\ Q_p$

**shows**  $partial\text{-pullback } m f l (carrier (Q_p^{Suc\ l})) = carrier (Q_p^{m+l})$

$\langle proof \rangle$

Definition 1.4 from Denef

**definition** *is-semialg-function where*

$is\text{-semialg}\text{-function } m f = ((f \in carrier (Q_p^m) \rightarrow carrier\ Q_p) \wedge (\forall l \geq 0. \forall S \in semialg\text{-sets } (1+l). is\text{-semialgebraic } (m+l) (partial\text{-pullback } m f l S)))$

**lemma** *is-semialg-function-closed:*

**assumes** *is-semialg-function*  $m f$

**shows**  $f \in carrier (Q_p^m) \rightarrow carrier\ Q_p$

$\langle proof \rangle$

**lemma** *is-semialg-functionE:*

**assumes** *is-semialg-function*  $m f$

**assumes** *is-semialgebraic*  $(1+k) S$

**shows** *is-semialgebraic*  $(m+k) (partial\text{-pullback } m f k S)$

$\langle proof \rangle$

**lemma** *is-semialg-functionI:*

**assumes**  $f \in carrier (Q_p^m) \rightarrow carrier\ Q_p$

**assumes**  $\bigwedge k\ S. S \in semialg\text{-sets } (1+k) \implies is\text{-semialgebraic } (m+k) (partial\text{-pullback } m f k S)$

**shows** *is-semialg-function*  $m f$

$\langle proof \rangle$

Semialgebraicity for functions can be verified on basic semialgebraic sets

**lemma** *is-semialg-functionI'*:

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } Q_p$

**assumes**  $\bigwedge k \ S. S \in \text{basic-semialgs } (1 + k) \implies \text{is-semialgebraic } (m + k)$   
*(partial-pullback m f k S)*

**shows** *is-semialg-function m f*

*<proof>*

Graphs of semialgebraic functions are semialgebraic

**abbreviation** *graph where*

*graph*  $\equiv \text{fun-graph } Q_p$

**lemma** *graph-memE*:

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } Q_p$

**assumes**  $x \in \text{graph } m f$

**shows**  $f (\text{take } m x) = x!m$

$x = (\text{take } m x)@[f (\text{take } m x)]$

$\text{take } m x \in \text{carrier } (Q_p^m)$

*<proof>*

**lemma** *graph-memI*:

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } Q_p$

**assumes**  $f (\text{take } m x) = x!m$

**assumes**  $x \in \text{carrier } (Q_p^{m+1})$

**shows**  $x \in \text{graph } m f$

*<proof>*

**lemma** *graph-mem-closed*:

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } Q_p$

**assumes**  $x \in \text{graph } m f$

**shows**  $x \in \text{carrier } (Q_p^{m+1})$

*<proof>*

**lemma** *graph-closed*:

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } Q_p$

**shows**  $\text{graph } m f \subseteq \text{carrier } (Q_p^{m+1})$

*<proof>*

The  $m$ -dimensional diagonal set is semialgebraic

**notation** *diagonal*  $(\Delta)$

**lemma** *diag-is-algebraic*:

**shows** *is-algebraic*  $Q_p (n + n) (\Delta n)$

*<proof>*

**lemma** *diag-is-semialgebraic*:

**shows** *is-semialgebraic*  $(n + n) (\Delta n)$

*<proof>*

Transposition permutations

**definition** *transpose* **where**

*transpose*  $i\ j = (\text{Fun.swap } i\ j\ \text{id})$

**lemma** *transpose-permutes*:

**assumes**  $i < n$

**assumes**  $j < n$

**shows** *transpose*  $i\ j$  permutes  $\{..<n\}$

*<proof>*

**lemma** *transpose-alt-def*:

*transpose*  $a\ b\ x = (\text{if } x = a \text{ then } b \text{ else if } x = b \text{ then } a \text{ else } x)$

*<proof>*

**definition** *last-to-first* **where**

*last-to-first*  $n = (\lambda i. \text{if } i = (n-1) \text{ then } 0 \text{ else if } i < n-1 \text{ then } i + 1 \text{ else } i)$

**definition** *first-to-last* **where**

*first-to-last*  $n = \text{fun-inv } (\text{last-to-first } n)$

**lemma** *last-to-first-permutes*:

**assumes**  $(n::\text{nat}) > 0$

**shows** *last-to-first*  $n$  permutes  $\{..<n\}$

*<proof>*

**definition** *graph-swap* **where**

*graph-swap*  $n\ f = \text{permute-list } ((\text{first-to-last } (n+1))) \text{ ' } (\text{graph } n\ f)$

**lemma** *last-to-first-eq*:

**assumes**  $\text{length } as = n$

**shows**  $\text{permute-list } (\text{last-to-first } (n+1))\ (a\#\text{as}) = (as@[a])$

*<proof>*

**lemma** *first-to-last-eq*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $a \in \text{carrier } Q_p$

**shows**  $\text{permute-list } (\text{first-to-last } (n+1))\ (as@[a]) = (a\#\text{as})$

*<proof>*

**lemma** *graph-swapI*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$

**shows**  $(f\ as)\#\text{as} \in \text{graph-swap } n\ f$

*<proof>*

**lemma** *graph-swapE*:

**assumes**  $x \in \text{graph-swap } n\ f$

**assumes**  $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$

**shows**  $\text{hd } x = f\ (\text{tl } x)$

*<proof>*

Semialgebraic functions have semialgebraic graphs

**lemma** *graph-as-partial-pullback*:

**assumes**  $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$

**shows**  $\text{partial-pullback } n \ f \ 1 \ (\Delta \ 1) = \text{graph } n \ f$

*<proof>*

**lemma** *semialg-graph*:

**assumes** *is-semialg-function*  $n \ f$

**shows** *is-semialgebraic*  $(n + 1) \ (\text{graph } n \ f)$

*<proof>*

Functions induced by polynomials are semialgebraic

**definition** *var-list-segment* **where**

*var-list-segment*  $i \ j = \text{map } (\lambda i. \ \text{pvar } Q_p \ i) \ [i..< j]$

**lemma** *var-list-segment-length*:

**assumes**  $i \leq j$

**shows**  $\text{length } (\text{var-list-segment } i \ j) = j - i$

*<proof>*

**lemma** *var-list-segment-entry*:

**assumes**  $k < j - i$

**assumes**  $i \leq j$

**shows**  $\text{var-list-segment } i \ j \ ! \ k = \text{pvar } Q_p \ (i + k)$

*<proof>*

**lemma** *var-list-segment-is-poly-tuple*:

**assumes**  $i \leq j$

**assumes**  $j \leq n$

**shows** *is-poly-tuple*  $n \ (\text{var-list-segment } i \ j)$

*<proof>*

**lemma** *map-by-var-list-segment*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $j \leq n$

**assumes**  $i \leq j$

**shows**  $\text{poly-map } n \ (\text{var-list-segment } i \ j) \ as = \text{list-segment } i \ j \ as$

*<proof>*

**lemma** *map-by-var-list-segment-to-length*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $i \leq n$

**shows**  $\text{poly-map } n \ (\text{var-list-segment } i \ n) \ as = \text{drop } i \ as$

*<proof>*

**lemma** *map-tail-by-var-list-segment*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $i < n$   
**shows**  $\text{poly-map } (n+1) (\text{var-list-segment } 1 (n+1)) (a\#as) = as$   
 <proof>

**lemma** *Qp-poly-tuple-Cons*:  
**assumes**  $\text{is-poly-tuple } n fs$   
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_k])$   
**assumes**  $k \leq n$   
**shows**  $\text{is-poly-tuple } n (f\#fs)$   
 <proof>

**lemma** *poly-map-Cons*:  
**assumes**  $\text{is-poly-tuple } n fs$   
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $\text{poly-map } n (f\#fs) a = (Qp\text{-ev } f a)\#\text{poly-map } n fs a$   
 <proof>

**lemma** *poly-map-append'*:  
**assumes**  $\text{is-poly-tuple } n fs$   
**assumes**  $\text{is-poly-tuple } n gs$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $\text{poly-map } n (fs@gs) a = \text{poly-map } n fs a @ \text{poly-map } n gs a$   
 <proof>

**lemma** *partial-pullback-by-poly*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $S \subseteq \text{carrier } (Q_p^{1+k})$   
**shows**  $\text{partial-pullback } n (Qp\text{-ev } f) k S = \text{poly-tuple-pullback } (n+k) S (f\#$   
 $(\text{var-list-segment } n (n+k)))$   
 <proof>

**lemma** *poly-is-semialg*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $\text{is-semialg-function } n (Qp\text{-ev } f)$   
 <proof>

Families of polynomials defined by semialgebraic coefficient functions

**lemma** *semialg-function-on-carrier*:  
**assumes**  $\text{is-semialg-function } n f$   
**assumes**  $\text{restrict } f (\text{carrier } (Q_p^n)) = \text{restrict } g (\text{carrier } (Q_p^n))$   
**shows**  $\text{is-semialg-function } n g$   
 <proof>

**lemma** *semialg-function-on-carrier'*:  
**assumes**  $\text{is-semialg-function } n f$   
**assumes**  $\bigwedge a. a \in \text{carrier } (Q_p^n) \implies f a = g a$   
**shows**  $\text{is-semialg-function } n g$

*<proof>*

**lemma** *constant-function-is-semialg:*

**assumes**  $n > 0$

**assumes**  $x \in \text{carrier } Q_p$

**assumes**  $\bigwedge a. a \in \text{carrier } (Q_p^n) \implies f a = x$

**shows** *is-semialg-function*  $n f$

*<proof>*

**lemma** *cartesian-product-singleton-factor-projection-is-semialg:*

**assumes**  $A \subseteq \text{carrier } (Q_p^m)$

**assumes**  $b \in \text{carrier } (Q_p^n)$

**assumes** *is-semialgebraic*  $(m+n)$  (*cartesian-product*  $A \{b\}$ )

**shows** *is-semialgebraic*  $m A$

*<proof>*

**lemma** *cartesian-product-factor-projection-is-semialg:*

**assumes**  $A \subseteq \text{carrier } (Q_p^m)$

**assumes**  $B \subseteq \text{carrier } (Q_p^n)$

**assumes**  $B \neq \{\}$

**assumes** *is-semialgebraic*  $(m+n)$  (*cartesian-product*  $A B$ )

**shows** *is-semialgebraic*  $m A$

*<proof>*

**lemma** *partial-pullback-cartesian-product:*

**assumes**  $\xi \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } Q_p$

**assumes**  $S \subseteq \text{carrier } (Q_p^1)$

**shows** *cartesian-product* (*partial-pullback*  $m \xi 0 S$ ) (*carrier*  $(Q_p^1)$ ) = *partial-pullback*  $m \xi 1$  (*cartesian-product*  $S$  (*carrier*  $(Q_p^1)$ ))

*<proof>*

**lemma** *cartesian-product-swap:*

**assumes**  $A \subseteq \text{carrier } (Q_p^n)$

**assumes**  $B \subseteq \text{carrier } (Q_p^m)$

**assumes** *is-semialgebraic*  $(m+n)$  (*cartesian-product*  $A B$ )

**shows** *is-semialgebraic*  $(m+n)$  (*cartesian-product*  $B A$ )

*<proof>*

**lemma** *Qp-zero-subset-is-semialg:*

**assumes**  $S \subseteq \text{carrier } (Q_p^0)$

**shows** *is-semialgebraic*  $0 S$

*<proof>*

**lemma** *cartesian-product-empty-list:*

*cartesian-product*  $A \{\}$  =  $A$

*cartesian-product*  $\{\}$   $A$  =  $A$

*<proof>*

**lemma** *cartesian-product-singleton-factor-projection-is-semialg':*

**assumes**  $A \subseteq \text{carrier } (Q_p^m)$   
**assumes**  $b \in \text{carrier } (Q_p^n)$   
**assumes** *is-semialgebraic*  $(m+n)$  (*cartesian-product*  $A \{b\}$ )  
**shows** *is-semialgebraic*  $m A$   
 ⟨*proof*⟩

### 13.11 More on graphs of functions

This section lays the groundwork for showing that semialgebraic functions are closed under various algebraic operations

The take and drop functions on lists are polynomial maps

**lemma** *function-restriction*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow S$   
**assumes**  $n \leq k$   
**shows**  $(g \circ (\text{take } n)) \in \text{carrier } (Q_p^k) \rightarrow S$   
 ⟨*proof*⟩

**lemma** *partial-pullback-restriction*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $n < k$   
**shows** *partial-pullback*  $k (g \circ \text{take } n) m S =$   
 $\text{split-cartesian-product } (n + m) (k - n) n (\text{partial-pullback } n g m S) (\text{carrier } (Q_p^{k - n}))$   
 ⟨*proof*⟩

**lemma** *comp-take-is-semialg*:  
**assumes** *is-semialg-function*  $n g$   
**assumes**  $n < k$   
**assumes**  $0 < n$   
**shows** *is-semialg-function*  $k (g \circ (\text{take } n))$   
 ⟨*proof*⟩

Restriction of a graph to a semialgebraic domain

**lemma** *graph-formula*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**shows** *graph*  $n g = \{as \in \text{carrier } (Q_p^{\text{Suc } n}). g (\text{take } n as) = as!n\}$   
 ⟨*proof*⟩

**definition** *restricted-graph where*  
 $\text{restricted-graph } n g S = \{as \in \text{carrier } (Q_p^{\text{Suc } n}). \text{take } n as \in S \wedge g (\text{take } n as) = as!n \}$

**lemma** *restricted-graph-closed*:  
 $\text{restricted-graph } n g S \subseteq \text{carrier } (Q_p^{\text{Suc } n})$   
 ⟨*proof*⟩

**lemma** *restricted-graph-memE*:

**assumes**  $a \in \text{restricted-graph } n \ g \ S$   
**shows**  $a \in \text{carrier } (Q_p^{\text{Suc } n}) \ \text{take } n \ a \in S \ g \ (\text{take } n \ a) = a!n$   
 $\langle \text{proof} \rangle$

**lemma** *restricted-graph-mem-formula*:  
**assumes**  $a \in \text{restricted-graph } n \ g \ S$   
**shows**  $a = (\text{take } n \ a)@[g \ (\text{take } n \ a)]$   
 $\langle \text{proof} \rangle$

**lemma** *restricted-graph-memI*:  
**assumes**  $a \in \text{carrier } (Q_p^{\text{Suc } n})$   
**assumes**  $\text{take } n \ a \in S$   
**assumes**  $g \ (\text{take } n \ a) = a!n$   
**shows**  $a \in \text{restricted-graph } n \ g \ S$   
 $\langle \text{proof} \rangle$

**lemma** *restricted-graph-memI'*:  
**assumes**  $a \in S$   
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $S \subseteq \text{carrier } (Q_p^n)$   
**shows**  $(a@[g \ a]) \in \text{restricted-graph } n \ g \ S$   
 $\langle \text{proof} \rangle$

**lemma** *restricted-graph-subset*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $S \subseteq \text{carrier } (Q_p^n)$   
**shows**  $\text{restricted-graph } n \ g \ S \subseteq \text{graph } n \ g$   
 $\langle \text{proof} \rangle$

**lemma** *restricted-graph-subset'*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $S \subseteq \text{carrier } (Q_p^n)$   
**shows**  $\text{restricted-graph } n \ g \ S \subseteq \text{cartesian-product } S \ (\text{carrier } (Q_p^1))$   
 $\langle \text{proof} \rangle$

**lemma** *restricted-graph-intersection*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $S \subseteq \text{carrier } (Q_p^n)$   
**shows**  $\text{restricted-graph } n \ g \ S = \text{graph } n \ g \cap (\text{cartesian-product } S \ (\text{carrier } (Q_p^1)))$   
 $\langle \text{proof} \rangle$

**lemma** *restricted-graph-is-semialgebraic*:  
**assumes** *is-semialg-function*  $n \ g$   
**assumes** *is-semialgebraic*  $n \ S$   
**shows** *is-semialgebraic*  $(n+1) \ (\text{restricted-graph } n \ g \ S)$   
 $\langle \text{proof} \rangle$

**lemma** *take-closed*:  
**assumes**  $n \leq k$

**assumes**  $x \in \text{carrier } (Q_p^k)$   
**shows**  $\text{take } n \ x \in \text{carrier } (Q_p^n)$   
 $\langle \text{proof} \rangle$

**lemma** *take-compose-closed*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $n < k$   
**shows**  $g \circ \text{take } n \in \text{carrier } (Q_p^k) \rightarrow \text{carrier } Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *take-graph-formula*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $n < k$   
**assumes**  $0 < n$   
**shows**  $\text{graph } k \ (g \circ (\text{take } n)) = \{as \in \text{carrier } (Q_p^{k+1}). g \ (\text{take } n \ as) = as!k\}$   
 $\langle \text{proof} \rangle$

**lemma** *graph-memI'*:  
**assumes**  $a \in \text{carrier } (Q_p^{\text{Suc } n})$   
**assumes**  $\text{take } n \ a \in \text{carrier } (Q_p^n)$   
**assumes**  $g \ (\text{take } n \ a) = a!n$   
**shows**  $a \in \text{graph } n \ g$   
 $\langle \text{proof} \rangle$

**lemma** *graph-memI''*:  
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**shows**  $(a@[g \ a]) \in \text{graph } n \ g$   
 $\langle \text{proof} \rangle$

**lemma** *graph-as-restricted-graph*:  
**assumes**  $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**shows**  $\text{graph } n \ f = \text{restricted-graph } n \ f \ (\text{carrier } (Q_p^n))$   
 $\langle \text{proof} \rangle$

**definition** *double-graph where*  
 $\text{double-graph } n \ f \ g = \{as \in \text{carrier } (Q_p^{n+2}). f \ (\text{take } n \ as) = as!n \wedge g \ (\text{take } n \ as) = as!(n + 1)\}$

**lemma** *double-graph-rep*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**shows**  $\text{double-graph } n \ f \ g = \text{restricted-graph } (n + 1) \ (g \circ \text{take } n) \ (\text{graph } n \ f)$   
 $\langle \text{proof} \rangle$

**lemma** *double-graph-is-semialg*:  
**assumes**  $n > 0$   
**assumes** *is-semialg-function*  $n \ f$   
**assumes** *is-semialg-function*  $n \ g$

**shows** *is-semialgebraic*  $(n+2)$  (*double-graph*  $n$   $f$   $g$ )  
 ⟨*proof*⟩

**definition** *add-vars* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{padic-tuple} \Rightarrow \text{padic-number}$  **where**  
*add-vars*  $i$   $j$   $as = as!i \oplus_{Q_p} as!j$

**lemma** *add-vars-rep*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $i < n$

**assumes**  $j < n$

**shows**  $\text{add-vars } i$   $j$   $as = Qp\text{-ev } ((\text{pvar } Q_p \ i) \oplus_{Q_p[\mathcal{X}_n]} (\text{pvar } Q_p \ j)) \ as$

⟨*proof*⟩

**lemma** *add-vars-is-semialg*:

**assumes**  $i < n$

**assumes**  $j < n$

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows** *is-semialg-function*  $n$  (*add-vars*  $i$   $j$ )

⟨*proof*⟩

**definition** *mult-vars* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{padic-tuple} \Rightarrow \text{padic-number}$  **where**  
*mult-vars*  $i$   $j$   $as = as!i \otimes as!j$

**lemma** *mult-vars-rep*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $i < n$

**assumes**  $j < n$

**shows**  $\text{mult-vars } i$   $j$   $as = Qp\text{-ev } ((\text{pvar } Q_p \ i) \otimes_{Q_p[\mathcal{X}_n]} (\text{pvar } Q_p \ j)) \ as$

⟨*proof*⟩

**lemma** *mult-vars-is-semialg*:

**assumes**  $i < n$

**assumes**  $j < n$

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows** *is-semialg-function*  $n$  (*mult-vars*  $i$   $j$ )

⟨*proof*⟩

**definition** *minus-vars* ::  $\text{nat} \Rightarrow \text{padic-tuple} \Rightarrow \text{padic-number}$  **where**  
*minus-vars*  $i$   $as = \ominus_{Q_p} as!i$

**lemma** *minus-vars-rep*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $i < n$

**shows**  $\text{minus-vars } i$   $as = Qp\text{-ev } (\ominus_{Q_p[\mathcal{X}_n]} (\text{pvar } Q_p \ i)) \ as$

⟨*proof*⟩

**lemma** *minus-vars-is-semialg*:

**assumes**  $i < n$

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows** *is-semialg-function*  $n$  (*minus-vars*  $i$ )  
 ⟨*proof*⟩

**definition** *extended-graph* **where**

*extended-graph*  $n f g h = \{ as \in \text{carrier } (Q_p^{n+3}).$   
 $f \text{ (take } n \text{ as)} = as!n \wedge g \text{ (take } n \text{ as)} = as! (n + 1) \wedge h [(f \text{ (take } n \text{ as)}), (g \text{ (take } n \text{ as)})] = as! (n + 2) \}$

**lemma** *extended-graph-rep*:

*extended-graph*  $n f g h = \text{restricted-graph } (n + 2) (h \circ (\text{drop } n)) (\text{double-graph } n f g)$   
 ⟨*proof*⟩

**lemma** *function-tuple-eval-closed*:

**assumes** *is-function-tuple*  $Q_p n fs$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows** *function-tuple-eval*  $Q_p n fs x \in \text{carrier } (Q_p^{\text{length } fs})$   
 ⟨*proof*⟩

**definition** *k-graph* **where**

*k-graph*  $n fs = \{ x \in \text{carrier } (Q_p^n + \text{length } fs). x = (\text{take } n x)@ (\text{function-tuple-eval } Q_p n fs (\text{take } n x)) \}$

**lemma** *k-graph-memI*:

**assumes** *is-function-tuple*  $Q_p n fs$   
**assumes**  $x = as@(\text{function-tuple-eval } Q_p n fs as)$   
**assumes**  $as \in \text{carrier } (Q_p^n)$   
**shows**  $x \in \text{k-graph } n fs$   
 ⟨*proof*⟩

composing a function with a function tuple

**lemma** *Qp-function-tuple-comp-closed*:

**assumes**  $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $\text{length } fs = n$   
**assumes** *is-function-tuple*  $Q_p m fs$   
**shows** *function-tuple-comp*  $Q_p fs f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } Q_p$   
 ⟨*proof*⟩

### 13.11.1 Tuples of Semialgebraic Functions

Predicate for a tuple of semialgebraic functions

**definition** *is-semialg-function-tuple* **where**

*is-semialg-function-tuple*  $n fs = (\forall f \in \text{set } fs. \text{is-semialg-function } n f)$

**lemma** *is-semialg-function-tupleI*:

**assumes**  $\bigwedge f. f \in \text{set } fs \implies \text{is-semialg-function } n f$   
**shows** *is-semialg-function-tuple*  $n fs$   
 ⟨*proof*⟩

**lemma** *is-semialg-function-tupleE*:  
**assumes** *is-semialg-function-tuple n fs*  
**assumes**  $i < \text{length } fs$   
**shows** *is-semialg-function n (fs ! i)*  
*<proof>*

**lemma** *is-semialg-function-tupleE'*:  
**assumes** *is-semialg-function-tuple n fs*  
**assumes**  $f \in \text{set } fs$   
**shows** *is-semialg-function n f*  
*<proof>*

**lemma** *semialg-function-tuple-is-function-tuple*:  
**assumes** *is-semialg-function-tuple n fs*  
**shows** *is-function-tuple  $Q_p$  n fs*  
*<proof>*

**lemma** *const-poly-function-tuple-comp-is-semialg*:  
**assumes**  $n > 0$   
**assumes** *is-semialg-function-tuple n fs*  
**assumes**  $a \in \text{carrier } Q_p$   
**shows** *is-semialg-function n (poly-function-tuple-comp  $Q_p$  n fs (Qp-to-IP a))*  
*<proof>*

**lemma** *pvar-poly-function-tuple-comp-is-semialg*:  
**assumes**  $n > 0$   
**assumes** *is-semialg-function-tuple n fs*  
**assumes**  $i < \text{length } fs$   
**shows** *is-semialg-function n (poly-function-tuple-comp  $Q_p$  n fs (pvar  $Q_p$  i))*  
*<proof>*

Polynomial functions with semialgebraic coefficients

**definition** *point-to-univ-poly* ::  $\text{nat} \Rightarrow \text{padic-tuple} \Rightarrow \text{padic-univ-poly}$  **where**  
*point-to-univ-poly n a = ring-cfs-to-univ-poly n a*

**definition** *tuple-partial-image* **where**  
*tuple-partial-image m fs x = (function-tuple-eval  $Q_p$  m fs (take m x))@(drop m x)*

**lemma** *tuple-partial-image-closed*:  
**assumes**  $\text{length } fs > 0$   
**assumes** *is-function-tuple  $Q_p$  n fs*  
**assumes**  $x \in \text{carrier } (Q_p^{n+l})$   
**shows** *tuple-partial-image n fs x  $\in \text{carrier } (Q_p^{\text{length } fs + l})$*   
*<proof>*

**lemma** *tuple-partial-image-indices*:  
**assumes**  $\text{length } fs > 0$   
**assumes** *is-function-tuple  $Q_p$  n fs*

**assumes**  $x \in \text{carrier } (Q_p^{n+l})$   
**assumes**  $i < \text{length } fs$   
**shows**  $(\text{tuple-partial-image } n \text{ } fs \ x) ! i = (fs ! i) \text{ (take } n \ x)$   
 <proof>

**lemma** *tuple-partial-image-indices'*:  
**assumes**  $\text{length } fs > 0$   
**assumes** *is-function-tuple*  $Q_p \ n \ fs$   
**assumes**  $x \in \text{carrier } (Q_p^{n+l})$   
**assumes**  $i < l$   
**shows**  $(\text{tuple-partial-image } n \ fs \ x) ! (\text{length } fs + i) = x!(n + i)$   
 <proof>

**definition** *tuple-partial-pullback where*  
 $\text{tuple-partial-pullback } n \ fs \ l \ S = ((\text{tuple-partial-image } n \ fs) - 'S) \cap \text{carrier } (Q_p^{n+l})$

**lemma** *tuple-partial-pullback-memE*:  
**assumes**  $as \in \text{tuple-partial-pullback } m \ fs \ l \ S$   
**shows**  $as \in \text{carrier } (Q_p^{m+l}) \ \text{tuple-partial-image } m \ fs \ as \in S$   
 <proof>

**lemma** *tuple-partial-pullback-closed*:  
 $\text{tuple-partial-pullback } m \ fs \ l \ S \subseteq \text{carrier } (Q_p^{m+l})$   
 <proof>

**lemma** *tuple-partial-pullback-memI*:  
**assumes**  $as \in \text{carrier } (Q_p^{m+k})$   
**assumes** *is-function-tuple*  $Q_p \ m \ fs$   
**assumes**  $((\text{function-tuple-eval } Q_p \ m \ fs) \text{ (take } m \ as)) @ (\text{drop } m \ as) \in S$   
**shows**  $as \in \text{tuple-partial-pullback } m \ fs \ k \ S$   
 <proof>

**lemma** *tuple-partial-image-eq*:  
**assumes**  $as \in \text{carrier } (Q_p^n)$   
**assumes**  $bs \in \text{carrier } (Q_p^k)$   
**assumes**  $x = as @ bs$   
**shows**  $\text{tuple-partial-image } n \ fs \ x = ((\text{function-tuple-eval } Q_p \ n \ fs) \ as) @ bs$   
 <proof>

**lemma** *tuple-partial-pullback-memE'*:  
**assumes**  $as \in \text{carrier } (Q_p^n)$   
**assumes**  $bs \in \text{carrier } (Q_p^k)$   
**assumes**  $x = as @ bs$   
**assumes**  $x \in \text{tuple-partial-pullback } n \ fs \ k \ S$   
**shows**  $(\text{function-tuple-eval } Q_p \ n \ fs \ as) @ bs \in S$   
 <proof>

tuple partial pullbacks have the same algebraic properties as pullbacks

**lemma** *tuple-partial-pullback-intersect*:

*tuple-partial-pullback m fl* ( $S1 \cap S2$ ) = (*tuple-partial-pullback m fl*  $S1$ )  $\cap$  (*tuple-partial-pullback m fl*  $S2$ )  
 ⟨proof⟩

**lemma** *tuple-partial-pullback-union:*

*tuple-partial-pullback m fl* ( $S1 \cup S2$ ) = (*tuple-partial-pullback m fl*  $S1$ )  $\cup$  (*tuple-partial-pullback m fl*  $S2$ )  
 ⟨proof⟩

**lemma** *tuple-partial-pullback-complement:*

**assumes** *is-function-tuple*  $Q_p$   $m$   $fs$   
**shows** *tuple-partial-pullback m fs l* ((*carrier* ( $Q_p$  <sup>length fs + l</sup>) -  $S$ ) = *carrier* ( $Q_p$   <sup>$m + l$</sup> ) - (*tuple-partial-pullback m fs l*  $S$ ))  
 ⟨proof⟩

**lemma** *tuple-partial-pullback-carrier:*

**assumes** *is-function-tuple*  $Q_p$   $m$   $fs$   
**shows** *tuple-partial-pullback m fs l* (*carrier* ( $Q_p$  <sup>length fs + l</sup>) = *carrier* ( $Q_p$   <sup>$m + l$</sup> ))  
 ⟨proof⟩

**definition** *is-semialg-map-tuple where*

*is-semialg-map-tuple m fs* = (*is-function-tuple*  $Q_p$   $m$   $fs$   $\wedge$   
 ( $\forall l \geq 0. \forall S \in \text{semialg-sets } ((\text{length } fs) + l). \text{is-semialgebraic } (m + l) (\text{tuple-partial-pullback } m \text{ fs } l \ S)$ ))

**lemma** *is-semialg-map-tuple-closed:*

**assumes** *is-semialg-map-tuple m fs*  
**shows** *is-function-tuple*  $Q_p$   $m$   $fs$   
 ⟨proof⟩

**lemma** *is-semialg-map-tupleE:*

**assumes** *is-semialg-map-tuple m fs*  
**assumes** *is-semialgebraic* ((*length fs*) +  $l$ )  $S$   
**shows** *is-semialgebraic* ( $m + l$ ) (*tuple-partial-pullback m fs l*  $S$ )  
 ⟨proof⟩

**lemma** *is-semialg-map-tupleI:*

**assumes** *is-function-tuple*  $Q_p$   $m$   $fs$   
**assumes**  $\bigwedge k. S. S \in \text{semialg-sets } ((\text{length } fs) + k) \implies \text{is-semialgebraic } (m + k)$   
 (*tuple-partial-pullback m fs k*  $S$ )  
**shows** *is-semialg-map-tuple m fs*  
 ⟨proof⟩

Semialgebraicity for maps can be verified on basic semialgebraic sets

**lemma** *is-semialg-map-tupleI':*

**assumes** *is-function-tuple*  $Q_p$   $m$   $fs$   
**assumes**  $\bigwedge k. S. S \in \text{basic-semialgs } ((\text{length } fs) + k) \implies \text{is-semialgebraic } (m + k)$   
 (*tuple-partial-pullback m fs k*  $S$ )  
**shows** *is-semialg-map-tuple m fs*

*<proof>*

The goal of this section is to show that tuples of semialgebraic functions are semialgebraic maps.

The function  $(x_0, x, y) \mapsto (x_0, f(x), y)$

**definition** *twisted-partial-image* **where**

*twisted-partial-image*  $n\ m\ f\ xs = (\text{take } n\ xs) @ \text{partial-image } m\ f\ (\text{drop } n\ xs)$

The set  $(x_0, x, y) \mid (x_0, f(x), y) \in S$

Convention: a function which produces a subset of  $(\mathbb{Q}_p^{i+j+k})$  will receive the 3 arity parameters in sequence, at the very beginning of the function

**definition** *twisted-partial-pullback* **where**

*twisted-partial-pullback*  $n\ m\ l\ f\ S = ((\text{twisted-partial-image } n\ m\ f) - 'S) \cap \text{carrier } (Q_p^{n+m+l})$

**lemma** *twisted-partial-pullback-memE*:

**assumes**  $as \in \text{twisted-partial-pullback } n\ m\ l\ f\ S$

**shows**  $as \in \text{carrier } (Q_p^{n+m+l}) \text{ twisted-partial-image } n\ m\ f\ as \in S$

*<proof>*

**lemma** *twisted-partial-pullback-closed*:

*twisted-partial-pullback*  $n\ m\ l\ f\ S \subseteq \text{carrier } (Q_p^{n+m+l})$

*<proof>*

**lemma** *twisted-partial-pullback-memI*:

**assumes**  $as \in \text{carrier } (Q_p^{n+m+l})$

**assumes**  $(\text{take } n\ as) @ ((f (\text{take } m (\text{drop } n\ as))) \# (\text{drop } (n + m)\ as)) \in S$

**shows**  $as \in \text{twisted-partial-pullback } n\ m\ l\ f\ S$

*<proof>*

**lemma** *twisted-partial-image-eq*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $bs \in \text{carrier } (Q_p^m)$

**assumes**  $cs \in \text{carrier } (Q_p^l)$

**assumes**  $x = as @ bs @ cs$

**shows**  $\text{twisted-partial-image } n\ m\ f\ x = as @ ((f\ bs) \# cs)$

*<proof>*

**lemma** *twisted-partial-pullback-memE'*:

**assumes**  $as \in \text{carrier } (Q_p^n)$

**assumes**  $bs \in \text{carrier } (Q_p^m)$

**assumes**  $cs \in \text{carrier } (Q_p^l)$

**assumes**  $x = as @ bs @ cs$

**assumes**  $x \in \text{twisted-partial-pullback } n\ m\ l\ f\ S$

**shows**  $as @ ((f\ bs) \# cs) \in S$

*<proof>*

partial pullbacks have the same algebraic properties as pullbacks

permutation which moves the entry at index  $i$  to 0

**definition** *twisting-permutation* **where**

*twisting-permutation*  $(i::nat) = (\lambda j. \text{if } j < i \text{ then } j + 1 \text{ else } \\ (\text{if } j = i \text{ then } 0 \text{ else } j))$

**lemma** *twisting-permutation-permutes*:

**assumes**  $i < n$

**shows** *twisting-permutation*  $i$  permutes  $\{..<n\}$

*<proof>*

**lemma** *twisting-permutation-action*:

**assumes**  $\text{length } as = i$

**shows** *permute-list* (*twisting-permutation*  $i$ )  $(b\#(as@bs)) = as@(b\#bs)$

*<proof>*

**lemma** *twisting-permutation-action'*:

**assumes**  $\text{length } as = i$

**shows** *permute-list* (*fun-inv* (*twisting-permutation*  $i$ ))  $(as@(b\#bs)) = (b\#(as@bs))$

*<proof>*

**lemma** *twisting-semialg*:

**assumes** *is-semialgebraic*  $n$   $S$

**assumes**  $n > i$

**shows** *is-semialgebraic*  $n$  (*permute-list* (*twisting-permutation*  $i$ )) ‘ $S$ )

*<proof>*

**lemma** *twisting-semialg'*:

**assumes** *is-semialgebraic*  $n$   $S$

**assumes**  $n > i$

**shows** *is-semialgebraic*  $n$  (*permute-list* (*fun-inv* (*twisting-permutation*  $i$ )) ‘ $S$ )

*<proof>*

Defining a permutation that does:  $(x_0, x_1, y) \mapsto (x_1, x_0, y)$

**definition** *tp-1* **where**

*tp-1*  $i$   $j = (\lambda n. (\text{if } n < i \text{ then } j + n \text{ else } \\ (\text{if } i \leq n \wedge n < i + j \text{ then } n - i \text{ else } \\ n)))$

**lemma** *permutes-I*:

**assumes**  $\bigwedge x. x \notin S \implies f x = x$

**assumes**  $\bigwedge y. y \in S \implies \exists !x \in S. f x = y$

**assumes**  $\bigwedge x. x \in S \implies f x \in S$

**shows**  $f$  permutes  $S$

*<proof>*

**lemma** *tp-1-permutes*:

*(tp-1*  $(i::nat)$   $j$ ) permutes  $\{..< i + j\}$

$\langle \text{proof} \rangle$

**lemma** *tp-1-permutes'*:

$(\text{tp-1 } (i::\text{nat}) j) \text{ permutes } \{..< i + j + k\}$   
 $\langle \text{proof} \rangle$

**lemma** *tp-1-permutation-action*:

**assumes**  $a \in \text{carrier } (Q_p^i)$   
**assumes**  $b \in \text{carrier } (Q_p^j)$   
**assumes**  $c \in \text{carrier } (Q_p^n)$   
**shows**  $\text{permute-list } (\text{tp-1 } i j) (b@a@c) = a@b@c$   
 $\langle \text{proof} \rangle$

**definition** *tw where*

$\text{tw } i j = \text{permute-list } (\text{tp-1 } j i)$

**lemma** *tw-is-semialg*:

**assumes**  $n > 0$   
**assumes** *is-semialgebraic*  $n S$   
**assumes**  $n \geq i + j$   
**shows** *is-semialgebraic*  $n ((\text{tw } i j) 'S)$   
 $\langle \text{proof} \rangle$

**lemma** *twisted-partial-pullback-factored*:

**assumes**  $f \in (\text{carrier } (Q_p^m)) \rightarrow \text{carrier } Q_p$   
**assumes**  $S \subseteq \text{carrier } (Q_p^{n+1+l})$   
**assumes**  $Y = \text{partial-pullback } m f (n + l) (\text{permute-list } (\text{fun-inv } (\text{twisting-permutation } n)) 'S)$   
**shows**  $\text{twisted-partial-pullback } n m l f S = (\text{tw } m n) ' Y$   
 $\langle \text{proof} \rangle$

**lemma** *twisted-partial-pullback-is-semialgebraic*:

**assumes** *is-semialg-function*  $m f$   
**assumes** *is-semialgebraic*  $(n + 1 + l) S$   
**shows** *is-semialgebraic*  $(n + m + l)(\text{twisted-partial-pullback } n m l f S)$   
 $\langle \text{proof} \rangle$

**definition** *augment where*

$\text{augment } n x = \text{take } n x @ \text{take } n x @ \text{drop } n x$

**lemma** *augment-closed*:

**assumes**  $x \in \text{carrier } (Q_p^{n+l})$   
**shows**  $\text{augment } n x \in \text{carrier } (Q_p^{n+n+l})$   
 $\langle \text{proof} \rangle$

**lemma** *tuple-partial-image-factor*:

**assumes** *is-function-tuple*  $Q_p m fs$   
**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } Q_p$   
**assumes**  $\text{length } fs = n$

**assumes**  $x \in \text{carrier } (Q_p^{m+l})$   
**shows**  $\text{tuple-partial-image } m \text{ (fs@[f]) } x = \text{twisted-partial-image } n \text{ m f (tuple-partial-image } m \text{ fs (augment m x))}$   
 <proof>

**definition diagonalize where**  
 $\text{diagonalize } n \text{ m } S = S \cap \text{cartesian-product } (\Delta \ n) \text{ (carrier } (Q_p^m))$

**lemma diagaonlize-is-semialgebraic:**  
**assumes**  $\text{is-semialgebraic } (n+n+m) \ S$   
**shows**  $\text{is-semialgebraic } (n+n+m) \ (\text{diagonalize } n \text{ m } S)$   
 <proof>

**lemma list-segment-take:**  
**assumes**  $\text{length } a \geq n$   
**shows**  $\text{list-segment } 0 \ n \ a = \text{take } n \ a$   
 <proof>

**lemma augment-inverse-is-semialgebraic:**  
**assumes**  $\text{is-semialgebraic } (n+n+l) \ S$   
**shows**  $\text{is-semialgebraic } (n+l) \ ((\text{augment } n \ -' \ S) \cap \text{carrier } (Q_p^{n+l}))$   
 <proof>

**lemma tuple-partial-pullback-is-semialg-map-tuple-induct:**  
**assumes**  $\text{is-semialg-map-tuple } m \ \text{fs}$   
**assumes**  $\text{is-semialg-function } m \ \text{f}$   
**assumes**  $\text{length } \text{fs} = n$   
**shows**  $\text{is-semialg-map-tuple } m \ \text{(fs@[f])}$   
 <proof>

**lemma singleton-tuple-partial-pullback-is-semialg-map-tuple:**  
**assumes**  $\text{is-semialg-function-tuple } m \ \text{fs}$   
**assumes**  $\text{length } \text{fs} = 1$   
**shows**  $\text{is-semialg-map-tuple } m \ \text{fs}$   
 <proof>

**lemma empty-tuple-partial-pullback-is-semialg-map-tuple:**  
**assumes**  $\text{is-semialg-function-tuple } m \ \text{fs}$   
**assumes**  $\text{length } \text{fs} = 0$   
**shows**  $\text{is-semialg-map-tuple } m \ \text{fs}$   
 <proof>

**lemma tuple-partial-pullback-is-semialg-map-tuple:**  
**assumes**  $\text{is-semialg-function-tuple } m \ \text{fs}$   
**shows**  $\text{is-semialg-map-tuple } m \ \text{fs}$   
 <proof>

### 13.11.2 Semialgebraic Functions are Closed under Composition with Semialgebraic Tuples

**lemma** *function-tuple-comp-partial-pullback*:

**assumes** *is-semialg-function-tuple*  $m$   $fs$

**assumes**  $length\ fs = n$

**assumes** *is-semialg-function*  $n$   $f$

**assumes**  $S \subseteq carrier\ (Q_p^{1+k})$

**shows** *partial-pullback*  $m$  (*function-tuple-comp*  $Q_p$   $fs$   $f$ )  $k$   $S =$   
*tuple-partial-pullback*  $m$   $fs$   $k$  (*partial-pullback*  $n$   $f$   $k$   $S$ )

*<proof>*

**lemma** *semialg-function-tuple-comp*:

**assumes** *is-semialg-function-tuple*  $m$   $fs$

**assumes**  $length\ fs = n$

**assumes** *is-semialg-function*  $n$   $f$

**shows** *is-semialg-function*  $m$  (*function-tuple-comp*  $Q_p$   $fs$   $f$ )

*<proof>*

### 13.11.3 Algebraic Operations on Semialgebraic Functions

Defining the set of extensional semialgebraic functions

**definition** *Qp-add-fun* **where**

*Qp-add-fun*  $xs = xs!0 \oplus_{Q_p} xs!1$

**definition** *Qp-mult-fun* **where**

*Qp-mult-fun*  $xs = xs!0 \otimes xs!1$

Inversion function on first coordinates of  $Q_p$  tuples. Arbitrarily redefined at 0 to map to 0

**definition** *Qp-invert* **where**

*Qp-invert*  $xs = (if\ ((xs!0) = \mathbf{0})\ then\ \mathbf{0}\ else\ (inv\ (xs!0)))$

Addition is semialgebraic

**lemma** *addition-is-semialg*:

*is-semialg-function* 2 *Qp-add-fun*

*<proof>*

Multiplication is semialgebraic:

**lemma** *multiplication-is-semialg*:

*is-semialg-function* 2 *Qp-mult-fun*

*<proof>*

Inversion is semialgebraic:

**lemma**(**in** *field*) *field-nat-pow-inv*:

**assumes**  $a \in carrier\ R$

**assumes**  $a \neq \mathbf{0}$

**shows**  $inv\ (a\ [\wedge]\ (n::nat)) = (inv\ a)\ [\wedge]\ (n::nat)$

*<proof>*

**lemma** *Qp-invert-basic-semialg:*

**assumes** *is-basic-semialg* (1 + k) S

**shows** *is-semialgebraic* (1 + k) (*partial-pullback* 1 *Qp-invert* k S)

*<proof>*

**lemma** *Qp-invert-is-semialg:*

*is-semialg-function* 1 *Qp-invert*

*<proof>*

**lemma** *Taylor-deg-1-expansion'':*

**assumes**  $f \in \text{carrier } Q_p\text{-}x$

**assumes**  $\bigwedge n. f\ n \in \mathcal{O}_p$

**assumes**  $a \in \mathcal{O}_p$

**assumes**  $b \in \mathcal{O}_p$

**shows**  $\exists c\ c'\ c''. c = \text{to-fun } f\ a \wedge c' = \text{deriv } f\ a \wedge c \in \mathcal{O}_p \wedge c' \in \mathcal{O}_p \wedge c'' \in \mathcal{O}_p \wedge$   
 $\text{to-fun } f\ (b) = c \oplus c' \otimes (b \ominus a) \oplus (c'' \otimes (b \ominus a))[\uparrow](2::\text{nat})$

*<proof>*

**end**

### 13.12 Sets Defined by Residues of Valuation Ring Elements

**sublocale** *padic-fields* < *Res: cring Zp-res-ring* (*Suc* n)

*<proof>*

**context** *padic-fields*

**begin**

**definition** *Qp-res where*

*Qp-res* x n = *to-Zp* x n

**lemma** *Qp-res-closed:*

**assumes**  $x \in \mathcal{O}_p$

**shows** *Qp-res* x n  $\in \text{carrier } (Zp\text{-res-ring } n)$

*<proof>*

**lemma** *Qp-res-add:*

**assumes**  $x \in \mathcal{O}_p$

**assumes**  $y \in \mathcal{O}_p$

**shows** *Qp-res* (x  $\oplus$  y) n = *Qp-res* x n  $\oplus_{Zp\text{-res-ring } n}$  *Qp-res* y n

*<proof>*

**lemma** *Qp-res-mult:*

**assumes**  $x \in \mathcal{O}_p$

**assumes**  $y \in \mathcal{O}_p$

**shows** *Qp-res* (x  $\otimes$  y) n = *Qp-res* x n  $\otimes_{Zp\text{-res-ring } n}$  *Qp-res* y n

*<proof>*

**lemma** *Qp-res-diff*:  
**assumes**  $x \in \mathcal{O}_p$   
**assumes**  $y \in \mathcal{O}_p$   
**shows**  $Qp\text{-res } (x \ominus y) n = Qp\text{-res } x n \ominus_{Zp\text{-res-ring } n} Qp\text{-res } y n$   
 $\langle proof \rangle$

**lemma** *Qp-res-zero*:  
**shows**  $Qp\text{-res } \mathbf{0} n = 0$   
 $\langle proof \rangle$

**lemma** *Qp-res-one*:  
**assumes**  $n > 0$   
**shows**  $Qp\text{-res } \mathbf{1} n = (1::int)$   
 $\langle proof \rangle$

**lemma** *Qp-res-nat-inc*:  
**shows**  $Qp\text{-res } [(n::nat)].\mathbf{1} n = n \bmod p^{\wedge} n$   
 $\langle proof \rangle$

**lemma** *Qp-res-int-inc*:  
**shows**  $Qp\text{-res } [(k::int)].\mathbf{1} n = k \bmod p^{\wedge} n$   
 $\langle proof \rangle$

**lemma** *Qp-poly-res-monom*:  
**assumes**  $a \in \mathcal{O}_p$   
**assumes**  $x \in \mathcal{O}_p$   
**assumes**  $Qp\text{-res } a n = 0$   
**assumes**  $k > 0$   
**shows**  $Qp\text{-res } (up\text{-ring.monom } (UP Q_p) a k \cdot x) n = 0$   
 $\langle proof \rangle$

**lemma** *Qp-poly-res-zero*:  
**assumes**  $q \in \text{carrier } (UP Q_p)$   
**assumes**  $\bigwedge i. q i \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. Qp\text{-res } (q i) n = 0$   
**assumes**  $x \in \mathcal{O}_p$   
**shows**  $Qp\text{-res } (q \cdot x) n = 0$   
 $\langle proof \rangle$

**lemma** *Qp-poly-res-eval-0*:  
**assumes**  $f \in \text{carrier } (UP Q_p)$   
**assumes**  $g \in \text{carrier } (UP Q_p)$   
**assumes**  $x \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. f i \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. g i \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. Qp\text{-res } (f i) n = Qp\text{-res } (g i) n$   
**shows**  $Qp\text{-res } (f \cdot x) n = Qp\text{-res } (g \cdot x) n$   
 $\langle proof \rangle$

**lemma** *Qp-poly-res-eval-1:*  
**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $x \in \mathcal{O}_p$   
**assumes**  $y \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. f \ i \in \mathcal{O}_p$   
**assumes**  $Qp\text{-res } x \ n = Qp\text{-res } y \ n$   
**shows**  $Qp\text{-res } (f \cdot x) \ n = Qp\text{-res } (f \cdot y) \ n$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-poly-res-eval-2:*  
**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $g \in \text{carrier } (UP \ Q_p)$   
**assumes**  $x \in \mathcal{O}_p$   
**assumes**  $y \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. f \ i \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. g \ i \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. Qp\text{-res } (f \ i) \ n = Qp\text{-res } (g \ i) \ n$   
**assumes**  $Qp\text{-res } x \ n = Qp\text{-res } y \ n$   
**shows**  $Qp\text{-res } (f \cdot x) \ n = Qp\text{-res } (g \cdot y) \ n$   
 $\langle \text{proof} \rangle$

**definition** *poly-res-class where*  
 $\text{poly-res-class } n \ d \ f = \{q \in \text{carrier } (UP \ Q_p). \text{deg } Q_p \ q \leq d \wedge (\forall i. q \ i \in \mathcal{O}_p \wedge Qp\text{-res } (f \ i) \ n = Qp\text{-res } (q \ i) \ n)\}$

**lemma** *poly-res-class-closed:*  
**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $g \in \text{carrier } (UP \ Q_p)$   
**assumes**  $\text{deg } Q_p \ f \leq d$   
**assumes**  $\text{deg } Q_p \ g \leq d$   
**assumes**  $g \in \text{poly-res-class } n \ d \ f$   
**shows**  $\text{poly-res-class } n \ d \ f = \text{poly-res-class } n \ d \ g$   
 $\langle \text{proof} \rangle$

**lemma** *poly-res-class-memE:*  
**assumes**  $f \in \text{poly-res-class } n \ d \ g$   
**shows**  $f \in \text{carrier } (UP \ Q_p)$   
 $\text{deg } Q_p \ f \leq d$   
 $f \ i \in \mathcal{O}_p$   
 $Qp\text{-res } (g \ i) \ n = Qp\text{-res } (f \ i) \ n$   
 $\langle \text{proof} \rangle$

**definition** *val-ring-polys where*  
 $\text{val-ring-polys} = \{f \in \text{carrier } (UP \ Q_p). (\forall i. f \ i \in \mathcal{O}_p)\}$

**lemma** *val-ring-polys-closed:*  
 $\text{val-ring-polys} \subseteq \text{carrier } (UP \ Q_p)$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-polys-memI*:  
**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $\bigwedge i. f \ i \in \mathcal{O}_p$   
**shows**  $f \in \text{val-ring-polys}$   
*<proof>*

**lemma** *val-ring-polys-memE*:  
**assumes**  $f \in \text{val-ring-polys}$   
**shows**  $f \in \text{carrier } (UP \ Q_p)$   
 $f \ i \in \mathcal{O}_p$   
*<proof>*

**definition** *val-ring-polys-grad where*  
 $\text{val-ring-polys-grad } d = \{f \in \text{val-ring-polys}. \text{deg } Q_p \ f \leq d\}$

**lemma** *val-ring-polys-grad-closed*:  
 $\text{val-ring-polys-grad } d \subseteq \text{val-ring-polys}$   
*<proof>*

**lemma** *val-ring-polys-grad-closed'*:  
 $\text{val-ring-polys-grad } d \subseteq \text{carrier } (UP \ Q_p)$   
*<proof>*

**lemma** *val-ring-polys-grad-memI*:  
**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $\bigwedge i. f \ i \in \mathcal{O}_p$   
**assumes**  $\text{deg } Q_p \ f \leq d$   
**shows**  $f \in \text{val-ring-polys-grad } d$   
*<proof>*

**lemma** *val-ring-polys-grad-memE*:  
**assumes**  $f \in \text{val-ring-polys-grad } d$   
**shows**  $f \in \text{carrier } (UP \ Q_p)$   
 $\text{deg } Q_p \ f \leq d$   
 $f \ i \in \mathcal{O}_p$   
*<proof>*

**lemma** *poly-res-classes-in-val-ring-polys-grad*:  
**assumes**  $f \in \text{val-ring-polys-grad } d$   
**shows**  $\text{poly-res-class } n \ d \ f \subseteq \text{val-ring-polys-grad } d$   
*<proof>*

**lemma** *poly-res-class-disjoint*:  
**assumes**  $f \in \text{val-ring-polys-grad } d$   
**assumes**  $f \notin \text{poly-res-class } n \ d \ g$   
**shows**  $\text{poly-res-class } n \ d \ f \cap \text{poly-res-class } n \ d \ g = \{\}$   
*<proof>*

**lemma** *poly-res-class-refl*:  
**assumes**  $f \in \text{val-ring-polys-grad } d$   
**shows**  $f \in \text{poly-res-class } n \ d \ f$   
 $\langle \text{proof} \rangle$

**lemma** *poly-res-class-memI*:  
**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $\text{deg } Q_p \ f \leq d$   
**assumes**  $\bigwedge i. f \ i \in \mathcal{O}_p$   
**assumes**  $\bigwedge i. Qp\text{-res } (f \ i) \ n = Qp\text{-res } (g \ i) \ n$   
**shows**  $f \in \text{poly-res-class } n \ d \ g$   
 $\langle \text{proof} \rangle$

**definition** *poly-res-classes where*  
 $\text{poly-res-classes } n \ d = \text{poly-res-class } n \ d \ \text{'val-ring-polys-grad } d$

**lemma** *poly-res-classes-disjoint*:  
**assumes**  $A \in \text{poly-res-classes } n \ d$   
**assumes**  $B \in \text{poly-res-classes } n \ d$   
**assumes**  $g \in A - B$   
**shows**  $A \cap B = \{\}$   
 $\langle \text{proof} \rangle$

**definition** *int-fun-to-poly where*  
 $\text{int-fun-to-poly } (f::\text{nat} \Rightarrow \text{int}) \ i = [(f \ i)].1$

**lemma** *int-fun-to-poly-closed*:  
**assumes**  $\bigwedge i. i > d \implies f \ i = 0$   
**shows**  $\text{int-fun-to-poly } f \in \text{carrier } (UP \ Q_p)$   
 $\langle \text{proof} \rangle$

**lemma** *int-fun-to-poly-deg*:  
**assumes**  $\bigwedge i. i > d \implies f \ i = 0$   
**shows**  $\text{deg } Q_p \ (\text{int-fun-to-poly } f) \leq d$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-res-mod-triv*:  
**assumes**  $a \in \mathcal{O}_p$   
**shows**  $Qp\text{-res } a \ n \ \text{mod } p \ \hat{\ } n = Qp\text{-res } a \ n$   
 $\langle \text{proof} \rangle$

**lemma** *int-fun-to-poly-is-class-wit*:  
**assumes**  $f \in \text{poly-res-class } n \ d \ g$   
**shows**  $(\text{int-fun-to-poly } (\lambda i::\text{nat}. Qp\text{-res } (f \ i) \ n)) \in \text{poly-res-class } n \ d \ g$   
 $\langle \text{proof} \rangle$

**lemma** *finite-support-funs-finite*:  
 $\text{finite } ((\{..d\} \rightarrow \text{carrier } (Zp\text{-res-ring } n)) \cap \{(f::\text{nat} \Rightarrow \text{int}). \forall i > d. f \ i = 0\})$   
 $\langle \text{proof} \rangle$

**lemma** *poly-res-classes-finite*:  
*finite (poly-res-classes n d)*  
 ⟨*proof*⟩

**lemma** *Qp-res-eq-zeroI*:  
 assumes  $a \in \mathcal{O}_p$   
 assumes *val*  $a \geq n$   
 shows  $Qp\text{-res } a \ n = 0$   
 ⟨*proof*⟩

**lemma** *Qp-res-eqI*:  
 assumes  $a \in \mathcal{O}_p$   
 assumes  $b \in \mathcal{O}_p$   
 assumes  $Qp\text{-res } (a \ominus b) \ n = 0$   
 shows  $Qp\text{-res } a \ n = Qp\text{-res } b \ n$   
 ⟨*proof*⟩

**lemma** *Qp-res-eqI'*:  
 assumes  $a \in \mathcal{O}_p$   
 assumes  $b \in \mathcal{O}_p$   
 assumes *val*  $(a \ominus b) \geq n$   
 shows  $Qp\text{-res } a \ n = Qp\text{-res } b \ n$   
 ⟨*proof*⟩

**lemma** *Qp-res-eqE*:  
 assumes  $a \in \mathcal{O}_p$   
 assumes  $b \in \mathcal{O}_p$   
 assumes  $Qp\text{-res } a \ n = Qp\text{-res } b \ n$   
 shows *val*  $(a \ominus b) \geq n$   
 ⟨*proof*⟩

**lemma** *notin-closed*:  
 $(\neg ((c::eint) \leq x \wedge x \leq d)) = (x < c \vee d < x)$   
 ⟨*proof*⟩

**lemma** *Qp-res-neqI*:  
 assumes  $a \in \mathcal{O}_p$   
 assumes  $b \in \mathcal{O}_p$   
 assumes *val*  $(a \ominus b) < n$   
 shows  $Qp\text{-res } a \ n \neq Qp\text{-res } b \ n$   
 ⟨*proof*⟩

**lemma** *Qp-res-equal*:  
 assumes  $a \in \mathcal{O}_p$   
 assumes  $l = Qp\text{-res } a \ n$   
 shows  $Qp\text{-res } a \ n = Qp\text{-res } ([l]\cdot\mathbf{1}) \ n$   
 ⟨*proof*⟩

**definition** *Qp-res-class where*

$Qp\text{-res-class } n \ b = \{a \in \mathcal{O}_p. Qp\text{-res } a \ n = Qp\text{-res } b \ n\}$

**definition** *Qp-res-classes where*

$Qp\text{-res-classes } n = Qp\text{-res-class } n \ ' \ \mathcal{O}_p$

**lemma** *val-ring-int-inc-closed:*

$[(k::int)].\mathbf{1} \in \mathcal{O}_p$   
 $\langle proof \rangle$

**lemma** *val-ring-nat-inc-closed:*

$[(k::nat)].\mathbf{1} \in \mathcal{O}_p$   
 $\langle proof \rangle$

**lemma** *Qp-res-classes-wits:*

$Qp\text{-res-classes } n = (\lambda l::int. Qp\text{-res-class } n \ ([l].\mathbf{1})) \ ' \ (\text{carrier } (Zp\text{-res-ring } n))$   
 $\langle proof \rangle$

**lemma** *Qp-res-classes-finite:*

$finite \ (Qp\text{-res-classes } n)$   
 $\langle proof \rangle$

**definition** *Qp-cong-set where*

$Qp\text{-cong-set } \alpha \ a = \{x \in \mathcal{O}_p. \text{to-}Zp \ x \ \alpha = a \ \alpha\}$

**lemma** *Qp-cong-set-as-ball:*

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $a \ \alpha = 0$   
**shows**  $Qp\text{-cong-set } \alpha \ a = B_\alpha[0]$   
 $\langle proof \rangle$

**lemma** *Qp-cong-set-as-ball':*

**assumes**  $a \in \text{carrier } Z_p$   
**assumes**  $\text{val-}Zp \ a < \text{eint } (int \ \alpha)$   
**shows**  $Qp\text{-cong-set } \alpha \ a = B_\alpha[(\iota \ a)]$   
 $\langle proof \rangle$

**lemma** *Qp-cong-set-is-univ-semialgebraic:*

**assumes**  $a \in \text{carrier } Z_p$   
**shows**  $\text{is-univ-semialgebraic } (Qp\text{-cong-set } \alpha \ a)$   
 $\langle proof \rangle$

**lemma** *constant-res-set-semialg:*

**assumes**  $l \in \text{carrier } (Zp\text{-res-ring } n)$   
**shows**  $\text{is-univ-semialgebraic } \{x \in \mathcal{O}_p. Qp\text{-res } x \ n = l\}$   
 $\langle proof \rangle$

**end**

```

end
theory Padic-Semialgebraic-Function-Ring
  imports Padic-Field-Powers
begin

```

## 14 Rings of Semialgebraic Functions

In order to efficiently formalize Denef's proof of Macintyre's theorem, it is necessary to be able to reason about semialgebraic functions algebraically. For example, we need to consider polynomials in one variable whose coefficients are semialgebraic functions, and take their Taylor expansions centered at a semialgebraic function. To facilitate this kind of reasoning, it is necessary to construct, for each arity  $m$ , a ring  $\mathbf{SA}(m)$  of semialgebraic functions in  $m$  variables. These functions must be extensional functions which are undefined outside of the carrier set of  $\mathbb{Q}_p^m$ .

The developments in this theory are mainly lemmas and definitions which build the necessary theory to prove the cell decomposition theorems of [1], and finally Macintyre's Theorem, which says that semi-algebraic sets are closed under projections.

### 14.1 Some eint Arithmetic

```

context padic-fields
begin

```

```

lemma eint-minus-ineq':
  assumes  $a \leq \text{eint } N$ 
  assumes  $b - a \leq c$ 
  shows  $b - \text{eint } N \leq c$ 
  <proof>

```

```

lemma eint-minus-plus:
 $a - (\text{eint } b + \text{eint } c) = a - \text{eint } b - \text{eint } c$ 
  <proof>

```

```

lemma eint-minus-plus':
 $a - (\text{eint } b + \text{eint } c) = a - \text{eint } c - \text{eint } b$ 
  <proof>

```

```

lemma eint-minus-plus'':
  assumes  $a - \text{eint } c - \text{eint } b = \text{eint } f$ 
  shows  $a - \text{eint } c - \text{eint } f = \text{eint } b$ 
  <proof>

```

```

lemma uminus-involutive[simp]:
 $-(-x::\text{eint}) = x$ 
  <proof>

```

**lemma** *eint-minus*:

$(a::\text{eint}) - (b::\text{eint}) = a + (-b)$   
*<proof>*

**lemma** *eint-mult-Suc*:

$\text{eint } (\text{Suc } k) * a = \text{eint } k * a + a$   
*<proof>*

**lemma** *eint-mult-Suc-mono*:

**assumes**  $a \leq \text{eint } b \longrightarrow \text{eint } (\text{int } k) * a \leq \text{eint } (\text{int } k) * \text{eint } b$   
**shows**  $a \leq \text{eint } b \longrightarrow \text{eint } (\text{int } (\text{Suc } k)) * a \leq \text{eint } (\text{int } (\text{Suc } k)) * \text{eint } b$   
*<proof>*

**lemma** *eint-nat-mult-mono*:

**assumes**  $(a::\text{eint}) \leq b$   
**shows**  $\text{eint } (k::\text{nat}) * a \leq \text{eint } k * b$   
*<proof>*

**lemma** *eint-Suc-zero*:

$\text{eint } (\text{int } (\text{Suc } 0)) * a = a$   
*<proof>*

**lemma** *eint-add-mono*:

**assumes**  $(a::\text{eint}) \leq b$   
**assumes**  $(c::\text{eint}) \leq d$   
**shows**  $a + c \leq b + d$   
*<proof>*

**lemma** *eint-nat-mult-mono-rev*:

**assumes**  $k > 0$   
**assumes**  $\text{eint } (k::\text{nat}) * a \leq \text{eint } k * b$   
**shows**  $(a::\text{eint}) \leq b$   
*<proof>*

## 14.2 Lemmas on Function Ring Operations

**lemma** *Qp-funs-is-crimg*:

*crimg*  $(\text{Fun}_n \text{ } Q_p)$   
*<proof>*

**lemma** *Qp-funs-is-monoid*:

*monoid*  $(\text{Fun}_n \text{ } Q_p)$   
*<proof>*

**lemma** *Qp-funs-car-memE*:

**assumes**  $f \in \text{carrier } (\text{Fun}_n \text{ } Q_p)$   
**shows**  $f \in (\text{carrier } (Q_p^n)) \rightarrow (\text{carrier } Q_p)$   
*<proof>*

**lemma** *Qp-funs-car-memI*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $\bigwedge x. x \notin (\text{carrier } (Q_p^n)) \implies g x = \text{undefined}$   
**shows**  $g \in \text{carrier } (\text{Fun}_n Q_p)$   
*<proof>*

**lemma** *Qp-funs-car-memI'*:  
**assumes**  $g \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $\text{restrict } g (\text{carrier } (Q_p^n)) = g$   
**shows**  $g \in \text{carrier } (\text{Fun}_n Q_p)$   
*<proof>*

**lemma** *Qp-funs-car-memI''*:  
**assumes**  $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
**assumes**  $g = (\lambda x \in (\text{carrier } (Q_p^n)). f x)$   
**shows**  $g \in \text{carrier } (\text{Fun}_n Q_p)$   
*<proof>*

**lemma** *Qp-funs-one*:  
 $\mathbf{1}_{\text{Fun}_n Q_p} = (\lambda x \in \text{carrier } (Q_p^n). \mathbf{1})$   
*<proof>*

**lemma** *Qp-funs-zero*:  
 $\mathbf{0}_{\text{Fun}_n Q_p} = (\lambda x \in \text{carrier } (Q_p^n). \mathbf{0}_{Q_p})$   
*<proof>*

**lemma** *Qp-funs-add*:  
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in (\text{carrier } (Q_p^n)) \rightarrow \text{carrier } Q_p$   
**assumes**  $g \in (\text{carrier } (Q_p^n)) \rightarrow \text{carrier } Q_p$   
**shows**  $(f \oplus_{\text{Fun}_n Q_p} g) x = f x \oplus_{Q_p} g x$   
*<proof>*

**lemma** *Qp-funs-add'*:  
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in (\text{carrier } (\text{Fun}_n Q_p))$   
**assumes**  $g \in (\text{carrier } (\text{Fun}_n Q_p))$   
**shows**  $(f \oplus_{\text{Fun}_n Q_p} g) x = f x \oplus_{Q_p} g x$   
*<proof>*

**lemma** *Qp-funs-add''*:  
**assumes**  $f \in (\text{carrier } (\text{Fun}_n Q_p))$   
**assumes**  $g \in (\text{carrier } (\text{Fun}_n Q_p))$   
**shows**  $(f \oplus_{\text{Fun}_n Q_p} g) = (\lambda x \in \text{carrier } (Q_p^n). f x \oplus_{Q_p} g x)$   
*<proof>*

**lemma** *Qp-funs-add'''*:  
**assumes**  $x \in \text{carrier } (Q_p^n)$

**shows**  $(f \oplus_{\text{Fun}_n Q_p} g) x = f x \oplus_{Q_p} g x$   
 ⟨proof⟩

**lemma** *Qp-funs-mult*:

**assumes**  $x \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in (\text{carrier } (Q_p^n)) \rightarrow \text{carrier } Q_p$   
**assumes**  $g \in (\text{carrier } (Q_p^n)) \rightarrow \text{carrier } Q_p$   
**shows**  $(f \otimes_{\text{Fun}_n Q_p} g) x = f x \otimes g x$   
 ⟨proof⟩

**lemma** *Qp-funs-mult'*:

**assumes**  $x \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in (\text{carrier } (\text{Fun}_n Q_p))$   
**assumes**  $g \in (\text{carrier } (\text{Fun}_n Q_p))$   
**shows**  $(f \otimes_{\text{Fun}_n Q_p} g) x = f x \otimes g x$   
 ⟨proof⟩

**lemma** *Qp-funs-mult''*:

**assumes**  $f \in (\text{carrier } (\text{Fun}_n Q_p))$   
**assumes**  $g \in (\text{carrier } (\text{Fun}_n Q_p))$   
**shows**  $(f \otimes_{\text{Fun}_n Q_p} g) = (\lambda x \in \text{carrier } (Q_p^n). f x \otimes g x)$   
 ⟨proof⟩

**lemma** *Qp-funs-mult'''*:

**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $(f \otimes_{\text{Fun}_n Q_p} g) x = f x \otimes g x$   
 ⟨proof⟩

**lemma** *Qp-funs-a-inv*:

**assumes**  $x \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in (\text{carrier } (\text{Fun}_n Q_p))$   
**shows**  $(\ominus_{\text{Fun}_n Q_p} f) x = \ominus (f x)$   
 ⟨proof⟩

**lemma** *Qp-funs-a-inv'*:

**assumes**  $f \in (\text{carrier } (\text{Fun}_n Q_p))$   
**shows**  $(\ominus_{\text{Fun}_n Q_p} f) = (\lambda x \in \text{carrier } (Q_p^n). \ominus (f x))$   
 ⟨proof⟩

**abbreviation**(*input*) *Qp-const (c.-)* **where**

*Qp-const n c*  $\equiv$  *constant-function* (*carrier*  $(Q_p^n)$ ) *c*

**lemma** *Qp-constE*:

**assumes**  $c \in \text{carrier } Q_p$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $Qp\text{-const } n \ c \ x = c$   
 ⟨proof⟩

**lemma** *Qp-funs-Units-memI*:  
**assumes**  $f \in (\text{carrier } (\text{Fun}_n Q_p))$   
**assumes**  $\bigwedge x. x \in \text{carrier } (Q_p^n) \implies f x \neq \mathbf{0}_{Q_p}$   
**shows**  $f \in (\text{Units } (\text{Fun}_n Q_p))$   
 $\text{inv}_{\text{Fun}_n Q_p} f = (\lambda x \in \text{carrier } (Q_p^n). \text{inv}_{Q_p} (f x))$   
*<proof>*

**lemma** *Qp-funs-Units-memE*:  
**assumes**  $f \in (\text{Units } (\text{Fun}_n Q_p))$   
**shows**  $f \otimes_{\text{Fun}_n Q_p} \text{inv}_{\text{Fun}_n Q_p} f = \mathbf{1}_{\text{Fun}_n Q_p}$   
 $\text{inv}_{\text{Fun}_n Q_p} f \otimes_{\text{Fun}_n Q_p} f = \mathbf{1}_{\text{Fun}_n Q_p}$   
 $\bigwedge x. x \in \text{carrier } (Q_p^n) \implies f x \neq \mathbf{0}_{Q_p}$   
*<proof>*

**lemma** *Qp-funs-m-inv*:  
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in (\text{Units } (\text{Fun}_n Q_p))$   
**shows**  $(\text{inv}_{\text{Fun}_n Q_p} f) x = \text{inv}_{Q_p} (f x)$   
*<proof>*

### 14.3 Defining the Rings of Semialgebraic Functions

**definition** *semialg-functions where*  
 $\text{semialg-functions } n = \{f \in (\text{carrier } (Q_p^n)) \rightarrow \text{carrier } Q_p. \text{ is-semialg-function } n f$   
 $\wedge f = \text{restrict } f (\text{carrier } (Q_p^n))\}$

**lemma** *semialg-functions-memE*:  
**assumes**  $f \in \text{semialg-functions } n$   
**shows** *is-semialg-function*  $n f$   
 $f \in \text{carrier } (\text{Fun}_n Q_p)$   
 $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
*<proof>*

**lemma** *semialg-functions-in-Qp-funs*:  
 $\text{semialg-functions } n \subseteq \text{carrier } (\text{Fun}_n Q_p)$   
*<proof>*

**lemma** *semialg-functions-memI*:  
**assumes**  $f \in \text{carrier } (\text{Fun}_n Q_p)$   
**assumes** *is-semialg-function*  $n f$   
**shows**  $f \in \text{semialg-functions } n$   
*<proof>*

**lemma** *restrict-is-semialg*:  
**assumes** *is-semialg-function*  $n f$   
**shows** *is-semialg-function*  $n (\text{restrict } f (\text{carrier } (Q_p^n)))$   
*<proof>*

**lemma** *restrict-in-semialg-functions*:

**assumes** *is-semialg-function*  $n$   $f$   
**shows**  $(\text{restrict } f \text{ (carrier } (Q_p^n))) \in \text{semialg-functions } n$   
 $\langle \text{proof} \rangle$

**lemma** *constant-function-is-semialg*:

**assumes**  $a \in \text{carrier } Q_p$   
**shows** *is-semialg-function*  $n$   $(\text{constant-function (carrier } (Q_p^n)) a)$   
 $\langle \text{proof} \rangle$

**lemma** *constant-function-in-semialg-functions*:

**assumes**  $a \in \text{carrier } Q_p$   
**shows**  $Qp\text{-const } n a \in \text{semialg-functions } n$   
 $\langle \text{proof} \rangle$

**lemma** *function-one-as-constant*:

$\mathbf{1}_{\text{Fun}_n Q_p} = Qp\text{-const } n \mathbf{1}$   
 $\langle \text{proof} \rangle$

**lemma** *function-zero-as-constant*:

$\mathbf{0}_{\text{Fun}_n Q_p} = Qp\text{-const } n \mathbf{0}_{Q_p}$   
 $\langle \text{proof} \rangle$

**lemma** *sum-in-semialg-functions*:

**assumes**  $f \in \text{semialg-functions } n$   
**assumes**  $g \in \text{semialg-functions } n$   
**shows**  $f \oplus_{\text{Fun}_n Q_p} g \in \text{semialg-functions } n$   
 $\langle \text{proof} \rangle$

**lemma** *prod-in-semialg-functions*:

**assumes**  $f \in \text{semialg-functions } n$   
**assumes**  $g \in \text{semialg-functions } n$   
**shows**  $f \otimes_{\text{Fun}_n Q_p} g \in \text{semialg-functions } n$   
 $\langle \text{proof} \rangle$

**lemma** *inv-in-semialg-functions*:

**assumes**  $f \in \text{semialg-functions } n$   
**assumes**  $\bigwedge x. x \in \text{carrier } (Q_p^n) \implies f x \neq \mathbf{0}_{Q_p}$   
**shows**  $\text{inv}_{\text{Fun}_n Q_p} f \in \text{semialg-functions } n$   
 $\langle \text{proof} \rangle$

**lemma** *a-inv-in-semialg-functions*:

**assumes**  $f \in \text{semialg-functions } n$   
**shows**  $\ominus_{\text{Fun}_n Q_p} f \in \text{semialg-functions } n$   
 $\langle \text{proof} \rangle$

**lemma** *semialg-functions-subring*:

**shows** *subring (semialg-functions )*  $(\text{Fun}_n Q_p)$   
 $\langle \text{proof} \rangle$

**lemma** *semialg-functions-subcring*:  
**shows** *subcring* (*semialg-functions n*) (*Fun<sub>n</sub> Q<sub>p</sub>*)  
 ⟨*proof*⟩

**definition** *SA where*  
 $SA\ n = (Fun_n\ Q_p)(\text{carrier} := \text{semialg-functions } n)$

**lemma** *SA-is-ring*:  
**shows** *ring* (*SA n*)  
 ⟨*proof*⟩

**lemma** *SA-is-criring*:  
**shows** *cring* (*SA n*)  
 ⟨*proof*⟩

**lemma** *SA-is-monoid*:  
**shows** *monoid* (*SA n*)  
 ⟨*proof*⟩

**lemma** *SA-is-abelian-monoid*:  
**shows** *abelian-monoid* (*SA n*)  
 ⟨*proof*⟩

**lemma** *SA-car*:  
 $\text{carrier} (SA\ n) = \text{semialg-functions } n$   
 ⟨*proof*⟩

**lemma** *SA-car-in-Qp-funs-car*:  
 $\text{carrier} (SA\ n) \subseteq \text{carrier} (Fun_n\ Q_p)$   
 ⟨*proof*⟩

**lemma** *SA-car-memI*:  
**assumes**  $f \in \text{carrier} (Fun_n\ Q_p)$   
**assumes** *is-semialg-function n f*  
**shows**  $f \in \text{carrier} (SA\ n)$   
 ⟨*proof*⟩

**lemma** *SA-car-memE*:  
**assumes**  $f \in \text{carrier} (SA\ n)$   
**shows** *is-semialg-function n f*  
 $f \in \text{carrier} (Fun_n\ Q_p)$   
 $f \in \text{carrier} (Q_p^n) \rightarrow \text{carrier } Q_p$   
 ⟨*proof*⟩

**lemma** *SA-plus*:  
 $(\oplus SA\ n) = (\oplus Fun_n\ Q_p)$   
 ⟨*proof*⟩

**lemma** *SA-times*:

$$(\otimes_{SA} n) = (\otimes_{Fun n} Q_p)$$

*<proof>*

**lemma** *SA-one*:

$$(\mathbf{1}_{SA} n) = (\mathbf{1}_{Fun n} Q_p)$$

*<proof>*

**lemma** *SA-zero*:

$$(\mathbf{0}_{SA} n) = (\mathbf{0}_{Fun n} Q_p)$$

*<proof>*

**lemma** *SA-zero-is-function-ring*:

$$(Fun_0 Q_p) = SA\ 0$$

*<proof>*

**lemma** *constant-fun-closed*:

**assumes**  $c \in carrier\ Q_p$

**shows** *constant-function* ( $carrier\ (Q_p^m)$ )  $c \in carrier\ (SA\ m)$

*<proof>*

**lemma** *SA-0-car-memI*:

**assumes**  $\xi \in carrier\ (Q_p^0) \rightarrow carrier\ Q_p$

**assumes**  $\bigwedge x. x \notin carrier\ (Q_p^0) \implies \xi\ x = undefined$

**shows**  $\xi \in carrier\ (SA\ 0)$

*<proof>*

**lemma** *car-SA-0-mem-imp-const*:

**assumes**  $a \in carrier\ (SA\ 0)$

**shows**  $\exists c \in carrier\ Q_p. a = Qp-const\ 0\ c$

*<proof>*

**lemma** *SA-zeroE*:

**assumes**  $a \in carrier\ (Q_p^n)$

**shows**  $\mathbf{0}_{SA}\ n\ a = \mathbf{0}$

*<proof>*

**lemma** *SA-oneE*:

**assumes**  $a \in carrier\ (Q_p^n)$

**shows**  $\mathbf{1}_{SA}\ n\ a = \mathbf{1}$

*<proof>*

**end**

**sublocale** *padic-fields* < *UPSA?*: *UP-cring*  $SA\ m\ UP\ (SA\ m)$

*<proof>*

**context** *padic-fields*

**begin**

**lemma** *SA-add*:

**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $(f \oplus_{SA\ n} g) x = f x \oplus_{Q_p} g x$   
*<proof>*

**lemma** *SA-add'*:

**assumes**  $x \notin \text{carrier } (Q_p^n)$   
**shows**  $(f \oplus_{SA\ n} g) x = \text{undefined}$   
*<proof>*

**lemma** *SA-mult*:

**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $(f \otimes_{SA\ n} g) x = f x \otimes g x$   
*<proof>*

**lemma** *SA-mult'*:

**assumes**  $x \notin \text{carrier } (Q_p^n)$   
**shows**  $(f \otimes_{SA\ n} g) x = \text{undefined}$   
*<proof>*

**lemma** *SA-u-minus-eval*:

**assumes**  $f \in \text{carrier } (SA\ n)$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $(\ominus_{SA\ n} f) x = \ominus (f x)$   
*<proof>*

**lemma** *SA-a-inv-eval*:

**assumes**  $f \in \text{carrier } (SA\ n)$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $(\ominus_{SA\ n} f) x = \ominus (f x)$   
*<proof>*

**lemma** *SA-nat-pow*:

**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $(f \ [\frown]_{SA\ n} (k::\text{nat})) x = (f x) \ [\frown]_{Q_p} k$   
*<proof>*

**lemma** *SA-nat-pow'*:

**assumes**  $x \notin \text{carrier } (Q_p^n)$   
**shows**  $(f \ [\frown]_{SA\ n} (k::\text{nat})) x = \text{undefined}$   
*<proof>*

**lemma** *SA-add-closed-id*:

**assumes** *is-semialg-function*  $n\ f$   
**assumes** *is-semialg-function*  $n\ g$   
**shows**  $\text{restrict } f \ (\text{carrier } (Q_p^n)) \oplus_{SA\ n} \text{restrict } g \ (\text{carrier } (Q_p^n)) = f \oplus_{SA\ n} g$   
*<proof>*

**lemma** *SA-mult-closed-id*:

**assumes** *is-semialg-function n f*  
**assumes** *is-semialg-function n g*  
**shows**  $\text{restrict } f (\text{carrier } (Q_p^n)) \otimes_{SA\ n} \text{restrict } g (\text{carrier } (Q_p^n)) = f \otimes_{SA\ n} g$   
 ⟨*proof*⟩

**lemma** *SA-add-closed:*

**assumes** *is-semialg-function n f*  
**assumes** *is-semialg-function n g*  
**shows**  $f \oplus_{SA\ n} g \in \text{carrier } (SA\ n)$   
 ⟨*proof*⟩

**lemma** *SA-mult-closed:*

**assumes** *is-semialg-function n f*  
**assumes** *is-semialg-function n g*  
**shows**  $f \otimes_{SA\ n} g \in \text{carrier } (SA\ n)$   
 ⟨*proof*⟩

**lemma** *SA-add-closed-right:*

**assumes** *is-semialg-function n f*  
**assumes**  $g \in \text{carrier } (SA\ n)$   
**shows**  $f \oplus_{SA\ n} g \in \text{carrier } (SA\ n)$   
 ⟨*proof*⟩

**lemma** *SA-mult-closed-right:*

**assumes** *is-semialg-function n f*  
**assumes**  $g \in \text{carrier } (SA\ n)$   
**shows**  $f \otimes_{SA\ n} g \in \text{carrier } (SA\ n)$   
 ⟨*proof*⟩

**lemma** *SA-add-closed-left:*

**assumes**  $f \in \text{carrier } (SA\ n)$   
**assumes** *is-semialg-function n g*  
**shows**  $f \oplus_{SA\ n} g \in \text{carrier } (SA\ n)$   
 ⟨*proof*⟩

**lemma** *SA-mult-closed-left:*

**assumes**  $f \in \text{carrier } (SA\ n)$   
**assumes** *is-semialg-function n g*  
**shows**  $f \otimes_{SA\ n} g \in \text{carrier } (SA\ n)$   
 ⟨*proof*⟩

**lemma** *SA-nat-pow-closed:*

**assumes** *is-semialg-function n f*  
**shows**  $f [\_ ]_{SA\ n} (k::\text{nat}) \in \text{carrier } (SA\ n)$   
 ⟨*proof*⟩

**lemma** *SA-imp-semialg:*

**assumes**  $f \in \text{carrier } (SA\ n)$   
**shows** *is-semialg-function n f*

*<proof>*

**lemma** *SA-minus-closed:*

**assumes**  $f \in \text{carrier } (SA\ n)$

**assumes**  $g \in \text{carrier } (SA\ n)$

**shows**  $(f \ominus_{SA\ n} g) \in \text{carrier } (SA\ n)$

*<proof>*

**lemma**(*in ring*) *add-pow-closed :*

**assumes**  $b \in \text{carrier } R$

**shows**  $[(m::nat)] \cdot_R b \in \text{carrier } R$

*<proof>*

**lemma**(*in ring*) *add-pow-Suc:*

**assumes**  $b \in \text{carrier } R$

**shows**  $[(Suc\ m)] \cdot b = [m] \cdot b \oplus b$

*<proof>*

**lemma**(*in ring*) *add-pow-zero:*

**assumes**  $b \in \text{carrier } R$

**shows**  $[(0::nat)] \cdot b = \mathbf{0}$

*<proof>*

**lemma** *Fun-add-pow-apply:*

**assumes**  $b \in \text{carrier } (Fun_n\ Q_p)$

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows**  $[(m::nat)] \cdot_{Fun_n\ Q_p} b\ a = [m] \cdot (b\ a)$

*<proof>*

**lemma** *SA-add-pow-apply:*

**assumes**  $b \in \text{carrier } (SA\ n)$

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows**  $[(m::nat)] \cdot_{SA\ n} b\ a = [m] \cdot (b\ a)$

*<proof>*

**lemma** *Qp-funs-Units-SA-Units:*

**assumes**  $f \in \text{Units } (Fun_n\ Q_p)$

**assumes** *is-semialg-function*  $n\ f$

**shows**  $f \in \text{Units } (SA\ n)$

*<proof>*

**lemma** *SA-Units-memE:*

**assumes**  $f \in (\text{Units } (SA\ n))$

**shows**  $f \otimes_{SA\ n} \text{inv}_{SA\ n} f = \mathbf{1}_{SA\ n}$

$\text{inv}_{SA\ n} f \otimes_{SA\ n} f = \mathbf{1}_{SA\ n}$

*<proof>*

**lemma** *SA-Units-closed:*

**assumes**  $f \in (\text{Units } (SA\ n))$

**shows**  $f \in \text{carrier } (SA \ n)$   
 $\langle \text{proof} \rangle$

**lemma** *SA-Units-inv-closed*:  
**assumes**  $f \in (\text{Units } (SA \ n))$   
**shows**  $\text{inv}_{SA \ n} f \in \text{carrier } (SA \ n)$   
 $\langle \text{proof} \rangle$

**lemma** *SA-Units-Qp-funs-Units*:  
**assumes**  $f \in (\text{Units } (SA \ n))$   
**shows**  $f \in (\text{Units } (\text{Fun}_n \ Q_p))$   
 $\langle \text{proof} \rangle$

**lemma** *SA-Units-Qp-funs-inv*:  
**assumes**  $f \in (\text{Units } (SA \ n))$   
**shows**  $\text{inv}_{SA \ n} f = \text{inv}_{\text{Fun}_n \ Q_p} f$   
 $\langle \text{proof} \rangle$

**lemma** *SA-Units-memI*:  
**assumes**  $f \in (\text{carrier } (SA \ n))$   
**assumes**  $\bigwedge x. x \in \text{carrier } (Q_p^n) \implies f \ x \neq \mathbf{0}_{Q_p}$   
**shows**  $f \in (\text{Units } (SA \ n))$   
 $\langle \text{proof} \rangle$

**lemma** *SA-Units-memE'*:  
**assumes**  $f \in (\text{Units } (SA \ n))$   
**shows**  $\bigwedge x. x \in \text{carrier } (Q_p^n) \implies f \ x \neq \mathbf{0}_{Q_p}$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-n-nonempty*:  
**shows**  $\text{carrier } (Q_p^n) \neq \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *SA-one-not-zero*:  
**shows**  $\mathbf{1}_{SA \ n} \neq \mathbf{0}_{SA \ n}$   
 $\langle \text{proof} \rangle$

**lemma** *SA-units-not-zero*:  
**assumes**  $f \in \text{Units } (SA \ n)$   
**shows**  $f \neq \mathbf{0}_{SA \ n}$   
 $\langle \text{proof} \rangle$

**lemma** *SA-Units-nonzero*:  
**assumes**  $f \in \text{Units } (SA \ m)$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $f \ x \in \text{nonzero } Q_p$   
 $\langle \text{proof} \rangle$

**lemma** *SA-car-closed*:

**assumes**  $f \in \text{carrier } (SA\ m)$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $f\ x \in \text{carrier } Q_p$   
 ⟨proof⟩

**lemma** *SA-Units-closed-fun*:  
**assumes**  $f \in \text{Units } (SA\ m)$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $f\ x \in \text{carrier } Q_p$   
 ⟨proof⟩

**lemma** *SA-inv-eval*:  
**assumes**  $f \in \text{Units } (SA\ n)$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $(\text{inv}_{SA\ n}\ f)\ x = \text{inv}\ (f\ x)$   
 ⟨proof⟩

**lemma** *SA-div-eval*:  
**assumes**  $f \in \text{Units } (SA\ n)$   
**assumes**  $h \in \text{carrier } (SA\ n)$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $(h \otimes_{SA\ n}\ (\text{inv}_{SA\ n}\ f))\ x = h\ x \otimes \text{inv}\ (f\ x)$   
 ⟨proof⟩

**lemma** *SA-unit-int-pow*:  
**assumes**  $f \in \text{Units } (SA\ m)$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $(f[\hat{\cdot}]_{SA\ m}(i::\text{int}))\ x = (f\ x)[\hat{\cdot}]^i$   
 ⟨proof⟩

**lemma** *restrict-in-SA-car*:  
**assumes** *is-semialg-function*  $n\ f$   
**shows**  $\text{restrict}\ f\ (\text{carrier } (Q_p^n)) \in \text{carrier } (SA\ n)$   
 ⟨proof⟩

**lemma** *SA-smult*:  
 $(\odot_{SA\ n}) = (\odot_{\text{Fun}_n\ Q_p})$   
 ⟨proof⟩

**lemma** *SA-smult-formula*:  
**assumes**  $h \in \text{carrier } (SA\ n)$   
**assumes**  $q \in \text{carrier } Q_p$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $(q \odot_{SA\ n}\ h)\ a = q \otimes (h\ a)$   
 ⟨proof⟩

**lemma** *SA-smult-closed*:  
**assumes**  $h \in \text{carrier } (SA\ n)$   
**assumes**  $q \in \text{carrier } Q_p$

**shows**  $q \odot_{SA\ n} h \in \text{carrier } (SA\ n)$   
 $\langle \text{proof} \rangle$

**lemma** *p-mult-function-val*:  
**assumes**  $f \in \text{carrier } (SA\ m)$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $\text{val } ((\mathbf{p} \odot_{SA\ m} f) x) = \text{val } (f x) + 1$   
 $\langle \text{proof} \rangle$

**lemma** *Qp-char-0''*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $a \neq \mathbf{0}$   
**assumes**  $(k::\text{nat}) > 0$   
**shows**  $[k] \cdot a \neq \mathbf{0}$   
 $\langle \text{proof} \rangle$

**lemma** *SA-char-zero*:  
**assumes**  $f \in \text{carrier } (SA\ m)$   
**assumes**  $f \neq \mathbf{0}_{SA\ m}$   
**assumes**  $n > 0$   
**shows**  $[(n::\text{nat})] \cdot_{SA\ m} f \neq \mathbf{0}_{SA\ m}$   
 $\langle \text{proof} \rangle$

## 14.4 Defining Semialgebraic Maps

We can define a semialgebraic map in essentially the same way that Denef defines semialgebraic functions. As for functions, we can define the partial pullback of a set  $S \subseteq \mathbb{Q}_p^{n+l}$  by a map  $g : \mathbb{Q}_p^m \rightarrow \mathbb{Q}_p^n$  to be the set

$$\{(x, y) \in \mathbb{Q}_p^m \times \mathbb{Q}_p^l \mid (f(x), y) \in S\}$$

and say that  $g$  is a semialgebraic map if for every  $l$ , and every semialgebraic  $S \subseteq \mathbb{Q}_p^{n+l}$ , the partial pullback of  $S$  by  $g$  is also semialgebraic. On this definition, it is immediate that the composition  $f \circ g$  of a semialgebraic function  $f : \mathbb{Q}_p^n \rightarrow \mathbb{Q}$  and a semialgebraic map  $g : \mathbb{Q}_p^m \rightarrow \mathbb{Q}_p^n$  is semialgebraic. It is also not hard to show that a map is semialgebraic if and only if all of its coordinate functions are semialgebraic functions. This allows us to build new semialgebraic functions out of old ones via composition.

Generalizing the notion of partial image partial pullbacks from functions to maps:

**definition** *map-partial-image* **where**  
 $\text{map-partial-image } m\ f\ xs = (f\ (\text{take } m\ xs)) @ (\text{drop } m\ xs)$

**definition** *map-partial-pullback* **where**  
 $\text{map-partial-pullback } m\ f\ l\ S = (\text{map-partial-image } m\ f)^{-1}_{m+l} S$

**lemma** *map-partial-pullback-memE*:

**assumes**  $as \in \text{map-partial-pullback } m \ f \ l \ S$   
**shows**  $as \in \text{carrier } (Q_p^{m+l}) \ \text{map-partial-image } m \ f \ as \in S$   
 ⟨proof⟩

**lemma** *map-partial-pullback-closed*:  
 $\text{map-partial-pullback } m \ f \ l \ S \subseteq \text{carrier } (Q_p^{m+l})$   
 ⟨proof⟩

**lemma** *map-partial-pullback-memI*:  
**assumes**  $as \in \text{carrier } (Q_p^{m+k})$   
**assumes**  $(f \ (\text{take } m \ as)) @ (\text{drop } m \ as) \in S$   
**shows**  $as \in \text{map-partial-pullback } m \ f \ k \ S$   
 ⟨proof⟩

**lemma** *map-partial-image-eq*:  
**assumes**  $as \in \text{carrier } (Q_p^n)$   
**assumes**  $bs \in \text{carrier } (Q_p^k)$   
**assumes**  $x = as @ bs$   
**shows**  $\text{map-partial-image } n \ f \ x = (f \ as) @ bs$   
 ⟨proof⟩

**lemma** *map-partial-pullback-memE'*:  
**assumes**  $as \in \text{carrier } (Q_p^n)$   
**assumes**  $bs \in \text{carrier } (Q_p^k)$   
**assumes**  $x = as @ bs$   
**assumes**  $x \in \text{map-partial-pullback } n \ f \ k \ S$   
**shows**  $(f \ as) @ bs \in S$   
 ⟨proof⟩

Partial pullbacks have the same algebraic properties as pullbacks.

**lemma** *map-partial-pullback-intersect*:  
 $\text{map-partial-pullback } m \ f \ l \ (S1 \cap S2) = (\text{map-partial-pullback } m \ f \ l \ S1) \cap (\text{map-partial-pullback } m \ f \ l \ S2)$   
 ⟨proof⟩

**lemma** *map-partial-pullback-union*:  
 $\text{map-partial-pullback } m \ f \ l \ (S1 \cup S2) = (\text{map-partial-pullback } m \ f \ l \ S1) \cup (\text{map-partial-pullback } m \ f \ l \ S2)$   
 ⟨proof⟩

**lemma** *map-partial-pullback-complement*:  
**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } (Q_p^n)$   
**shows**  $\text{map-partial-pullback } m \ f \ l \ (\text{carrier } (Q_p^{n+l}) - S) = \text{carrier } (Q_p^{m+l}) - (\text{map-partial-pullback } m \ f \ l \ S)$   
 ⟨proof⟩

**lemma** *map-partial-pullback-carrier*:  
**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } (Q_p^n)$   
**shows**  $\text{map-partial-pullback } m \ f \ l \ (\text{carrier } (Q_p^{n+l})) = \text{carrier } (Q_p^{m+l})$

*<proof>*

**definition** *is-semialg-map where*

*is-semialg-map*  $m\ n\ f = (f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } (Q_p^n) \wedge$   
 $(\forall l \geq 0. \forall S \in \text{semialg-sets } (n + l). \text{is-semialgebraic } (m + l)$   
 $(\text{map-partial-pullback } m\ f\ l\ S)))$

**lemma** *is-semialg-map-closed:*

**assumes** *is-semialg-map*  $m\ n\ f$

**shows**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } (Q_p^n)$

*<proof>*

**lemma** *is-semialg-map-closed':*

**assumes** *is-semialg-map*  $m\ n\ f\ x \in \text{carrier } (Q_p^m)$

**shows**  $f\ x \in \text{carrier } (Q_p^n)$

*<proof>*

**lemma** *is-semialg-mapE:*

**assumes** *is-semialg-map*  $m\ n\ f$

**assumes** *is-semialgebraic*  $(n + k)\ S$

**shows** *is-semialgebraic*  $(m + k)\ (\text{map-partial-pullback } m\ f\ k\ S)$

*<proof>*

**lemma** *is-semialg-mapE':*

**assumes** *is-semialg-map*  $m\ n\ f$

**assumes** *is-semialgebraic*  $(n + k)\ S$

**shows** *is-semialgebraic*  $(m + k)\ (\text{map-partial-image } m\ f^{-1}_{m+k}\ S)$

*<proof>*

**lemma** *is-semialg-mapI:*

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } (Q_p^n)$

**assumes**  $\bigwedge k\ S. S \in \text{semialg-sets } (n + k) \implies \text{is-semialgebraic } (m + k)$   
 $(\text{map-partial-pullback } m\ f\ k\ S)$

**shows** *is-semialg-map*  $m\ n\ f$

*<proof>*

**lemma** *is-semialg-mapI':*

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } (Q_p^n)$

**assumes**  $\bigwedge k\ S. S \in \text{semialg-sets } (n + k) \implies \text{is-semialgebraic } (m + k)$   
 $(\text{map-partial-image } m\ f^{-1}_{m+k}\ S)$

**shows** *is-semialg-map*  $m\ n\ f$

*<proof>*

Semialgebraicity for functions can be verified on basic semialgebraic sets.

**lemma** *is-semialg-mapI'':*

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } (Q_p^n)$

**assumes**  $\bigwedge k\ S. S \in \text{basic-semialgs } (n + k) \implies \text{is-semialgebraic } (m + k)$   
 $(\text{map-partial-pullback } m\ f\ k\ S)$

**shows** *is-semialg-map*  $m\ n\ f$

*<proof>*

**lemma** *is-semialg-mapI'''*:

**assumes**  $f \in \text{carrier } (Q_p^m) \rightarrow \text{carrier } (Q_p^n)$

**assumes**  $\bigwedge k. S. S \in \text{basic-semialgs } (n + k) \implies \text{is-semialgebraic } (m + k)$   
*(map-partial-image m f<sup>-1</sup><sub>m+k</sub> S)*

**shows** *is-semialg-map m n f*

*<proof>*

**lemma** *id-is-semialg-map*:

*is-semialg-map n n ( $\lambda x. x$ )*

*<proof>*

**lemma** *map-partial-pullback-comp*:

**assumes** *is-semialg-map m n f*

**assumes** *is-semialg-map k m g*

**shows** *(map-partial-pullback k (f  $\circ$  g) l S) = (map-partial-pullback k g l (map-partial-pullback m f l S))*

*<proof>*

**lemma** *semialg-map-comp-closed*:

**assumes** *is-semialg-map m n f*

**assumes** *is-semialg-map k m g*

**shows** *is-semialg-map k n (f  $\circ$  g)*

*<proof>*

**lemma** *partial-pullback-comp*:

**assumes** *is-semialg-function m f*

**assumes** *is-semialg-map k m g*

**shows** *(partial-pullback k (f  $\circ$  g) l S) = (map-partial-pullback k g l (partial-pullback m f l S))*

*<proof>*

**lemma** *semialg-function-comp-closed*:

**assumes** *is-semialg-function m f*

**assumes** *is-semialg-map k m g*

**shows** *is-semialg-function k (f  $\circ$  g)*

*<proof>*

**lemma** *semialg-map-evimage-is-semialg*:

**assumes** *is-semialg-map k m g*

**assumes** *is-semialgebraic m S*

**shows** *is-semialgebraic k (g<sup>-1</sup><sub>k</sub> S)*

*<proof>*

## 14.5 Examples of Semialgebraic Maps

**lemma** *semialg-map-on-carrier*:

**assumes** *is-semialg-map n m f*

**assumes**  $restrict\ f\ (carrier\ (Q_p^n)) = restrict\ g\ (carrier\ (Q_p^n))$   
**shows**  $is-semialg-map\ n\ m\ g$   
 $\langle proof \rangle$

**lemma** *semialg-function-tuple-is-semialg-map*:  
**assumes**  $is-semialg-function-tuple\ m\ fs$   
**assumes**  $length\ fs = n$   
**shows**  $is-semialg-map\ m\ n\ (function-tuple-eval\ Q_p\ m\ fs)$   
 $\langle proof \rangle$

**lemma** *index-is-semialg-function*:  
**assumes**  $n > k$   
**shows**  $is-semialg-function\ n\ (\lambda as. as!k)$   
 $\langle proof \rangle$

**definition** *Qp-ith where*  
 $Qp-ith\ m\ i = (\lambda x \in carrier\ (Q_p^m). x!i)$

**lemma** *Qp-ith-closed*:  
**assumes**  $i < m$   
**shows**  $Qp-ith\ m\ i \in carrier\ (SA\ m)$   
 $\langle proof \rangle$

**lemma** *take-is-semialg-map*:  
**assumes**  $n \geq k$   
**shows**  $is-semialg-map\ n\ k\ (take\ k)$   
 $\langle proof \rangle$

**lemma** *drop-is-semialg-map*:  
**shows**  $is-semialg-map\ (k + n)\ n\ (drop\ k)$   
 $\langle proof \rangle$

**lemma** *project-at-indices-is-semialg-map*:  
**assumes**  $S \subseteq \{..<n\}$   
**shows**  $is-semialg-map\ n\ (card\ S)\ \pi_S$   
 $\langle proof \rangle$

**lemma** *tl-is-semialg-map*:  
**shows**  $is-semialg-map\ (Suc\ n)\ n\ tl$   
 $\langle proof \rangle$

Coordinate functions are semialgebraic maps.

**lemma** *coord-fun-is-SA*:  
**assumes**  $is-semialg-map\ n\ m\ g$   
**assumes**  $i < m$   
**shows**  $coord-fun\ Q_p\ n\ g\ i \in carrier\ (SA\ n)$   
 $\langle proof \rangle$

**lemma** *coord-fun-map-is-semialg-tuple*:

**assumes** *is-semialg-map*  $n\ m\ g$   
**shows** *is-semialg-function-tuple*  $n$  ( $\text{map } (\text{coord-fun } Q_p\ n\ g) [0..<m]$ )  
 $\langle \text{proof} \rangle$

**lemma** *semialg-map-cons*:

**assumes** *is-semialg-map*  $n\ m\ g$   
**assumes**  $f \in \text{carrier } (SA\ n)$   
**shows** *is-semialg-map*  $n$  ( $\text{Suc } m$ ) ( $\lambda x \in \text{carrier } (Q_p^n). f\ x \# g\ x$ )  
 $\langle \text{proof} \rangle$

Extensional Semialgebraic Maps:

**definition** *semialg-maps where*

*semialg-maps*  $n\ m \equiv \{f. \text{is-semialg-map } n\ m\ f \wedge f \in \text{struct-maps } (Q_p^n) (Q_p^m)\}$

**lemma** *semialg-mapsE*:

**assumes**  $f \in (\text{semialg-maps } n\ m)$   
**shows** *is-semialg-map*  $n\ m\ f$   
 $f \in \text{struct-maps } (Q_p^n) (Q_p^m)$   
 $f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } (Q_p^m)$   
 $\langle \text{proof} \rangle$

**definition** *to-semialg-map where*

*to-semialg-map*  $n\ m\ f = \text{restrict } f (\text{carrier } (Q_p^n))$

**lemma** *to-semialg-map-is-semialg-map*:

**assumes** *is-semialg-map*  $n\ m\ f$   
**shows** *to-semialg-map*  $n\ m\ f \in \text{semialg-maps } n\ m$   
 $\langle \text{proof} \rangle$

## 14.6 Application of Functions to Segments of Tuples

**definition** *take-apply where*

*take-apply*  $m\ n\ f = \text{restrict } (f \circ \text{take } n) (\text{carrier } (Q_p^m))$

**definition** *drop-apply where*

*drop-apply*  $m\ n\ f = \text{restrict } (f \circ \text{drop } n) (\text{carrier } (Q_p^m))$

**lemma** *take-apply-closed*:

**assumes**  $f \in \text{carrier } (\text{Fun}_n\ Q_p)$   
**assumes**  $k \geq n$   
**shows** *take-apply*  $k\ n\ f \in \text{carrier } (\text{Fun}_k\ Q_p)$   
 $\langle \text{proof} \rangle$

**lemma** *take-apply-SA-closed*:

**assumes**  $f \in \text{carrier } (SA\ n)$   
**assumes**  $k \geq n$   
**shows** *take-apply*  $k\ n\ f \in \text{carrier } (SA\ k)$   
 $\langle \text{proof} \rangle$

**lemma** *drop-apply-closed*:  
**assumes**  $f \in \text{carrier } (\text{Fun}_k - n \ Q_p)$   
**assumes**  $k \geq n$   
**shows**  $\text{drop-apply } k \ n \ f \in \text{carrier } (\text{Fun}_k \ Q_p)$   
 $\langle \text{proof} \rangle$

**lemma** *drop-apply-SA-closed*:  
**assumes**  $f \in \text{carrier } (SA \ (k-n))$   
**assumes**  $k \geq n$   
**shows**  $\text{drop-apply } k \ n \ f \in \text{carrier } (SA \ k)$   
 $\langle \text{proof} \rangle$

**lemma** *take-apply-apply*:  
**assumes**  $f \in \text{carrier } (SA \ n)$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**assumes**  $b \in \text{carrier } (Q_p^k)$   
**shows**  $\text{take-apply } (n+k) \ n \ f \ (a@b) = f \ a$   
 $\langle \text{proof} \rangle$

**lemma** *drop-apply-apply*:  
**assumes**  $f \in \text{carrier } (SA \ k)$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**assumes**  $b \in \text{carrier } (Q_p^k)$   
**shows**  $\text{drop-apply } (n+k) \ n \ f \ (a@b) = f \ b$   
 $\langle \text{proof} \rangle$

**lemma** *drop-apply-add*:  
**assumes**  $f \in \text{carrier } (SA \ n)$   
**assumes**  $g \in \text{carrier } (SA \ n)$   
**shows**  $\text{drop-apply } (n+k) \ k \ (f \oplus_{SA \ n} \ g) = \text{drop-apply } (n+k) \ k \ f \oplus_{SA \ (n+k)}$   
 $\text{drop-apply } (n+k) \ k \ g$   
 $\langle \text{proof} \rangle$

**lemma** *drop-apply-mult*:  
**assumes**  $f \in \text{carrier } (SA \ n)$   
**assumes**  $g \in \text{carrier } (SA \ n)$   
**shows**  $\text{drop-apply } (n+k) \ k \ (f \otimes_{SA \ n} \ g) = \text{drop-apply } (n+k) \ k \ f \otimes_{SA \ (n+k)}$   
 $\text{drop-apply } (n+k) \ k \ g$   
 $\langle \text{proof} \rangle$

**lemma** *drop-apply-one*:  
**shows**  $\text{drop-apply } (n+k) \ k \ \mathbf{1}_{SA \ n} = \mathbf{1}_{SA \ (n+k)}$   
 $\langle \text{proof} \rangle$

**lemma** *drop-apply-is-hom*:  
**shows**  $\text{drop-apply } (n+k) \ k \in \text{ring-hom } (SA \ n) \ (SA \ (n+k))$   
 $\langle \text{proof} \rangle$

**lemma** *take-apply-add*:

**assumes**  $f \in \text{carrier } (SA\ k)$

**assumes**  $g \in \text{carrier } (SA\ k)$

**shows**  $\text{take-apply } (n+k)\ k\ (f \oplus_{SA\ k}\ g) = \text{take-apply } (n+k)\ k\ f \oplus_{SA\ (n+k)}$

*take-apply*  $(n+k)\ k\ g$

*<proof>*

**lemma** *take-apply-mult*:

**assumes**  $f \in \text{carrier } (SA\ k)$

**assumes**  $g \in \text{carrier } (SA\ k)$

**shows**  $\text{take-apply } (n+k)\ k\ (f \otimes_{SA\ k}\ g) = \text{take-apply } (n+k)\ k\ f \otimes_{SA\ (n+k)}$

*take-apply*  $(n+k)\ k\ g$

*<proof>*

**lemma** *take-apply-one*:

**shows**  $\text{take-apply } (n+k)\ k\ \mathbf{1}_{SA\ k} = \mathbf{1}_{SA\ (n+k)}$

*<proof>*

**lemma** *take-apply-is-hom*:

**shows**  $\text{take-apply } (n+k)\ k \in \text{ring-hom } (SA\ k)\ (SA\ (n+k))$

*<proof>*

**lemma** *drop-apply-units*:

**assumes**  $f \in \text{Units } (SA\ n)$

**shows**  $\text{drop-apply } (n+k)\ k\ f \in \text{Units } (SA\ (n+k))$

*<proof>*

**lemma** *take-apply-units*:

**assumes**  $f \in \text{Units } (SA\ k)$

**shows**  $\text{take-apply } (n+k)\ k\ f \in \text{Units } (SA\ (n+k))$

*<proof>*

## 14.7 Level Sets of Semialgebraic Functions

**lemma** *evimage-is-semialg*:

**assumes**  $h \in \text{carrier } (SA\ n)$

**assumes** *is-univ-semialgebraic*  $S$

**shows** *is-semialgebraic*  $n\ (h^{-1}_n\ S)$

*<proof>*

**lemma** *semialg-val-ineq-set-is-semialg*:

**assumes**  $g \in \text{carrier } (SA\ k)$

**assumes**  $f \in \text{carrier } (SA\ k)$

**shows** *is-semialgebraic*  $k\ \{x \in \text{carrier } (Q_p^k). \text{val } (g\ x) \leq \text{val } (f\ x)\}$

*<proof>*

**lemma** *semialg-val-ineq-set-is-semialg'*:

**assumes**  $f \in \text{carrier } (SA\ k)$

**shows** *is-semialgebraic*  $k\ \{x \in \text{carrier } (Q_p^k). \text{val } (f\ x) \leq C\}$

*<proof>*

**lemma** *semialg-val-ineq-set-is-semialg''*:

**assumes**  $f \in \text{carrier } (SA \ k)$

**shows** *is-semialgebraic*  $k \ \{x \in \text{carrier } (Q_p^k). \text{val } (f \ x) \geq C\}$

*<proof>*

**lemma** *semialg-level-set-is-semialg*:

**assumes**  $f \in \text{carrier } (SA \ k)$

**assumes**  $c \in \text{carrier } Q_p$

**shows** *is-semialgebraic*  $k \ \{x \in \text{carrier } (Q_p^k). f \ x = c\}$

*<proof>*

**lemma** *semialg-val-eq-set-is-semialg'*:

**assumes**  $f \in \text{carrier } (SA \ k)$

**shows** *is-semialgebraic*  $k \ \{x \in \text{carrier } (Q_p^k). \text{val } (f \ x) = C\}$

*<proof>*

**lemma** *semialg-val-eq-set-is-semialg*:

**assumes**  $g \in \text{carrier } (SA \ k)$

**assumes**  $f \in \text{carrier } (SA \ k)$

**shows** *is-semialgebraic*  $k \ \{x \in \text{carrier } (Q_p^k). \text{val } (g \ x) = \text{val } (f \ x)\}$

*<proof>*

**lemma** *semialg-val-strict-ineq-set-formula*:

$\{x \in \text{carrier } (Q_p^k). \text{val } (g \ x) < \text{val } (f \ x)\} = \{x \in \text{carrier } (Q_p^k). \text{val } (g \ x) \leq \text{val } (f \ x)\} - \{x \in \text{carrier } (Q_p^k). \text{val } (g \ x) = \text{val } (f \ x)\}$

*<proof>*

**lemma** *semialg-val-ineq-set-complement*:

$\text{carrier } (Q_p^k) - \{x \in \text{carrier } (Q_p^k). \text{val } (g \ x) \leq \text{val } (f \ x)\} = \{x \in \text{carrier } (Q_p^k). \text{val } (f \ x) < \text{val } (g \ x)\}$

*<proof>*

**lemma** *semialg-val-strict-ineq-set-complement*:

$\text{carrier } (Q_p^k) - \{x \in \text{carrier } (Q_p^k). \text{val } (g \ x) < \text{val } (f \ x)\} = \{x \in \text{carrier } (Q_p^k). \text{val } (f \ x) \leq \text{val } (g \ x)\}$

*<proof>*

**lemma** *semialg-val-strict-ineq-set-is-semialg*:

**assumes**  $g \in \text{carrier } (SA \ k)$

**assumes**  $f \in \text{carrier } (SA \ k)$

**shows** *is-semialgebraic*  $k \ \{x \in \text{carrier } (Q_p^k). \text{val } (g \ x) < \text{val } (f \ x)\}$

*<proof>*

**lemma** *semialg-val-strict-ineq-set-is-semialg'*:

**assumes**  $f \in \text{carrier } (SA \ k)$

**shows** *is-semialgebraic*  $k \ \{x \in \text{carrier } (Q_p^k). \text{val } (f \ x) < C\}$

*<proof>*

**lemma** *semialg-val-strict-ineq-set-is-semialg''*:

**assumes**  $f \in \text{carrier } (SA\ k)$

**shows** *is-semialgebraic*  $k \{x \in \text{carrier } (Q_p^k). \text{val } (f\ x) > C\}$

*<proof>*

**lemma** *semialg-val-ineq-set-plus*:

**assumes**  $N > 0$

**assumes**  $c \in \text{carrier } (SA\ N)$

**assumes**  $a \in \text{carrier } (SA\ N)$

**shows** *is-semialgebraic*  $N \{x \in \text{carrier } (Q_p^N). \text{val } (c\ x) \leq \text{val } (a\ x) + \text{eint } n\}$

*<proof>*

**lemma** *semialg-val-eq-set-plus*:

**assumes**  $N > 0$

**assumes**  $c \in \text{carrier } (SA\ N)$

**assumes**  $a \in \text{carrier } (SA\ N)$

**shows** *is-semialgebraic*  $N \{x \in \text{carrier } (Q_p^N). \text{val } (c\ x) = \text{val } (a\ x) + \text{eint } n\}$

*<proof>*

**definition** *SA-zero-set where*

*SA-zero-set*  $n\ f = \{x \in \text{carrier } (Q_p^n). f\ x = \mathbf{0}\}$

**lemma** *SA-zero-set-is-semialgebraic*:

**assumes**  $f \in \text{carrier } (SA\ n)$

**shows** *is-semialgebraic*  $n\ (SA\ \text{zero-set } n\ f)$

*<proof>*

**lemma** *SA-zero-set-memE*:

**assumes**  $f \in \text{carrier } (SA\ n)$

**assumes**  $x \in SA\ \text{zero-set } n\ f$

**shows**  $f\ x = \mathbf{0}$

*<proof>*

**lemma** *SA-zero-set-memI*:

**assumes**  $f \in \text{carrier } (SA\ n)$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**assumes**  $f\ x = \mathbf{0}$

**shows**  $x \in SA\ \text{zero-set } n\ f$

*<proof>*

**lemma** *SA-zero-set-of-zero*:

*SA-zero-set*  $m\ (\mathbf{0}_{SA\ m}) = \text{carrier } (Q_p^m)$

*<proof>*

**definition** *SA-nonzero-set where*

*SA-nonzero-set*  $n\ f = \{x \in \text{carrier } (Q_p^n). f\ x \neq \mathbf{0}\}$

**lemma nonzero-evimage-closed:**

**assumes**  $f \in \text{carrier } (SA \ n)$

**shows**  $\text{is-semialgebraic } n \ \{x \in \text{carrier } (Q_p^n). \ f \ x \neq \mathbf{0}\}$

*<proof>*

**lemma SA-nonzero-set-is-semialgebraic:**

**assumes**  $f \in \text{carrier } (SA \ n)$

**shows**  $\text{is-semialgebraic } n \ (SA\text{-nonzero-set } n \ f)$

*<proof>*

**lemma SA-nonzero-set-memE:**

**assumes**  $f \in \text{carrier } (SA \ n)$

**assumes**  $x \in SA\text{-nonzero-set } n \ f$

**shows**  $f \ x \neq \mathbf{0}$

*<proof>*

**lemma SA-nonzero-set-memI:**

**assumes**  $f \in \text{carrier } (SA \ n)$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**assumes**  $f \ x \neq \mathbf{0}$

**shows**  $x \in SA\text{-nonzero-set } n \ f$

*<proof>*

**lemma SA-nonzero-set-of-zero:**

$SA\text{-nonzero-set } m \ (\mathbf{0}_{SA \ m}) = \{\}$

*<proof>*

**lemma SA-car-memI':**

**assumes**  $\bigwedge x. \ x \in \text{carrier } (Q_p^m) \implies f \ x \in \text{carrier } Q_p$

**assumes**  $\bigwedge x. \ x \notin \text{carrier } (Q_p^m) \implies f \ x = \text{undefined}$

**assumes**  $\bigwedge k \ n \ P. \ n > 0 \implies P \in \text{carrier } (Q_p [\mathcal{X}_{1+k}]) \implies \text{is-semialgebraic } (m$

$+ k) \ (\text{partial-pullback } m \ f \ k \ (\text{basic-semialg-set } (1+k) \ n \ P))$

**shows**  $f \in \text{carrier } (SA \ m)$

*<proof>*

**lemma(in padic-fields) SA-zero-set-is-semialg:**

**assumes**  $a \in \text{carrier } (SA \ m)$

**shows**  $\text{is-semialgebraic } m \ \{x \in \text{carrier } (Q_p^m). \ a \ x = \mathbf{0}\}$

*<proof>*

**lemma(in padic-fields) SA-nonzero-set-is-semialg:**

**assumes**  $a \in \text{carrier } (SA \ m)$

**shows**  $\text{is-semialgebraic } m \ \{x \in \text{carrier } (Q_p^m). \ a \ x \neq \mathbf{0}\}$

*<proof>*

**lemma zero-set-nonzero-set-covers:**

$\text{carrier } (Q_p^n) = SA\text{-zero-set } n \ f \cup SA\text{-nonzero-set } n \ f$

*<proof>*

**lemma** *zero-set-nonzero-set-covers'*:  
**assumes**  $S \subseteq \text{carrier } (Q_p^n)$   
**shows**  $S = (S \cap \text{SA-zero-set } n f) \cup (S \cap \text{SA-nonzero-set } n f)$   
 $\langle \text{proof} \rangle$

**lemma** *zero-set-nonzero-set-covers-semialg-set*:  
**assumes** *is-semialgebraic*  $n S$   
**shows**  $S = (S \cap \text{SA-zero-set } n f) \cup (S \cap \text{SA-nonzero-set } n f)$   
 $\langle \text{proof} \rangle$

## 14.8 Partitioning Semialgebraic Sets According to Valuations of Functions

Given a semialgebraic set  $A$  and a finite set of semialgebraic functions  $F$ s, a common construction is to simplify one's understanding of the behaviour of the functions  $F$ s on  $A$  by finitely partitioning  $A$  into subsets where the element  $f \in F$  for which  $\text{val}(fx)$  is minimal is constant as  $x$  ranges over each piece of the partition. The function `Min_set` helps construct this by picking out the subset of a set  $A$  where the valuation of a particular element of  $F$ s is minimal. Such a set will always be semialgebraic.

**lemma** *disjointify-semialg*:  
**assumes** *finite*  $As$   
**assumes**  $As \subseteq \text{semialg-sets } n$   
**shows** *disjointify*  $As \subseteq \text{semialg-sets } n$   
 $\langle \text{proof} \rangle$

**lemma** *semialgebraic-subalgebra*:  
**assumes** *finite*  $Xs$   
**assumes**  $Xs \subseteq \text{semialg-sets } n$   
**shows** *atoms-of*  $Xs \subseteq \text{semialg-sets } n$   
 $\langle \text{proof} \rangle$

**lemma**(in *padic-fields*) *finite-intersection-is-semialg*:  
**assumes** *finite*  $Xs$   
**assumes**  $Xs \neq \{\}$   
**assumes**  $F \text{ ' } Xs \subseteq \text{semialg-sets } m$   
**shows** *is-semialgebraic*  $m (\bigcap i \in Xs. F i)$   
 $\langle \text{proof} \rangle$

**definition** *Min-set where*  
 $\text{Min-set } m As a = \text{carrier } (Q_p^m) \cap (\bigcap f \in As. \{x \in \text{carrier } (Q_p^m). \text{val } (a x) \leq \text{val } (f x)\})$

**lemma** *Min-set-memE*:  
**assumes**  $x \in \text{Min-set } m As a$   
**assumes**  $f \in As$   
**shows**  $\text{val } (a x) \leq \text{val } (f x)$

*<proof>*

**lemma** *Min-set-closed:*

*Min-set*  $m$   $As$   $a \subseteq \text{carrier } (Q_p^m)$

*<proof>*

**lemma** *Min-set-semialg0:*

**assumes**  $As \subseteq \text{carrier } (SA\ m)$

**assumes** *finite*  $As$

**assumes**  $a \in As$

**assumes**  $As \neq \{\}$

**shows** *is-semialgebraic*  $m$  (*Min-set*  $m$   $As$   $a$ )

*<proof>*

**lemma** *Min-set-semialg:*

**assumes**  $As \subseteq \text{carrier } (SA\ m)$

**assumes** *finite*  $As$

**assumes**  $a \in As$

**shows** *is-semialgebraic*  $m$  (*Min-set*  $m$   $As$   $a$ )

*<proof>*

**lemma** *Min-sets-cover:*

**assumes**  $As \neq \{\}$

**assumes** *finite*  $As$

**shows**  $\text{carrier } (Q_p^m) = (\bigcup a \in As. \text{Min-set } m\ As\ a)$

*<proof>*

The sets defined by the function `Min_set` for a fixed set of functions may not all be disjoint, but we can easily refine them to obtain a finite partition via the function "disjointify".

**definition** *Min-set-partition where*

*Min-set-partition*  $m$   $As$   $B = \text{disjointify } ((\cap)B \text{ ` } (Min\text{-set } m\ As \text{ ` } As))$

**lemma** *Min-set-partition-finite:*

**assumes** *finite*  $As$

**shows** *finite* (*Min-set-partition*  $m$   $As$   $B$ )

*<proof>*

**lemma** *Min-set-partition-semialg0:*

**assumes** *finite*  $As$

**assumes**  $As \subseteq \text{carrier } (SA\ m)$

**assumes** *is-semialgebraic*  $m$   $B$

**assumes**  $S \in ((\cap)B \text{ ` } (Min\text{-set } m\ As \text{ ` } As))$

**shows** *is-semialgebraic*  $m$   $S$

*<proof>*

**lemma** *Min-set-partition-semialg:*

**assumes** *finite*  $As$

**assumes**  $As \subseteq \text{carrier } (SA\ m)$

**assumes** *is-semialgebraic*  $m$   $B$   
**assumes**  $S \in (\text{Min-set-partition } m \text{ } As \text{ } B)$   
**shows** *is-semialgebraic*  $m$   $S$   
 <proof>

**lemma** *Min-set-partition-covers0*:  
**assumes** *finite*  $As$   
**assumes**  $As \neq \{\}$   
**assumes**  $As \subseteq \text{carrier } (SA \text{ } m)$   
**assumes** *is-semialgebraic*  $m$   $B$   
**shows**  $\bigcup ((\bigcap) B \text{ } (Min-set \text{ } m \text{ } As \text{ } As)) = B$   
 <proof>

**lemma** *Min-set-partition-covers*:  
**assumes** *finite*  $As$   
**assumes**  $As \subseteq \text{carrier } (SA \text{ } m)$   
**assumes**  $As \neq \{\}$   
**assumes** *is-semialgebraic*  $m$   $B$   
**shows**  $\bigcup (Min-set-partition \text{ } m \text{ } As \text{ } B) = B$   
 <proof>

**lemma** *Min-set-partition-disjoint*:  
**assumes** *finite*  $As$   
**assumes**  $As \subseteq \text{carrier } (SA \text{ } m)$   
**assumes**  $As \neq \{\}$   
**assumes** *is-semialgebraic*  $m$   $B$   
**assumes**  $s \in \text{Min-set-partition } m \text{ } As \text{ } B$   
**assumes**  $s' \in \text{Min-set-partition } m \text{ } As \text{ } B$   
**assumes**  $s \neq s'$   
**shows**  $s \cap s' = \{\}$   
 <proof>

**lemma** *Min-set-partition-memE*:  
**assumes** *finite*  $As$   
**assumes**  $As \subseteq \text{carrier } (SA \text{ } m)$   
**assumes**  $As \neq \{\}$   
**assumes** *is-semialgebraic*  $m$   $B$   
**assumes**  $s \in \text{Min-set-partition } m \text{ } As \text{ } B$   
**shows**  $\exists f \in As. (\forall x \in s. (\forall g \in As. \text{val } (f \text{ } x) \leq \text{val } (g \text{ } x)))$   
 <proof>

## 14.9 Valuative Congruence Sets for Semialgebraic Functions

The set of points  $x$  where the values  $\text{ord } f(x)$  satisfy a congruence are important basic examples of semialgebraic sets, and will be vital in the proof of Macintyre's Theorem. The lemma below is essentially the content of Denef's Lemma 2.1.3 from his cell decomposition paper.

**lemma** *pre-SA-unit-cong-set-is-semialg*:

**assumes**  $k \geq 0$   
**assumes**  $f \in \text{Units}(SA\ n)$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier}(Q_p^n). \text{ord}(f\ x) \bmod k = a\}$   
 <proof>

**lemma** *SA-unit-cong-set-is-semialg:*

**assumes**  $f \in \text{Units}(SA\ n)$   
**shows**  $\text{is-semialgebraic } n \{x \in \text{carrier}(Q_p^n). \text{ord}(f\ x) \bmod k = a\}$   
 <proof>

## 14.10 Gluing Functions Along Semialgebraic Sets

Semialgebraic functions have the useful property that they are closed under piecewise definitions. That is, if  $f, g$  are semialgebraic and  $C \subseteq \mathbb{Q}_p^m$  is a semialgebraic set, then the function:

$$h(x) = \begin{cases} f(x) & \text{if } x \in C \\ g(x) & \text{if } x \in \mathbb{Q}_p^m - C \\ \text{undefined} & \text{otherwise} \end{cases}$$

is again semialgebraic. The function  $h$  can be obtained by the definition

$$\mathbf{h} = \text{fun\_glue } m\ C\ f\ g$$

which is defined below. This closure property means that we can avoid having to define partial semialgebraic functions which are undefined outside of some proper subset of  $\mathbb{Q}_p^m$ , since it usually suffices to just define the function as some arbitrary constant outside of the desired domain. This is useful for defining partial multiplicative inverses of arbitrary functions. If  $f$  is semialgebraic, then its nonzero set  $\{x \in \mathbb{Q}_p^m \mid f\ x \neq 0\}$  is semialgebraic. By gluing  $f$  to the constant function 1 outside of its nonzero set, we obtain an invertible element in the ring  $\text{SA}(m)$  which evaluates to a multiplicative inverse of  $f(x)$  on the largest domain possible.

### 14.10.1 Defining Piecewise Semialgebraic Functions

An important property that will be repeatedly used is that we can define piecewise semialgebraic functions, which will themselves be semialgebraic as long as the pieces are semialgebraic sets. An important application of this principle will be that a function  $f$  which is always nonzero on some semialgebraic set  $A$  can be replaced with a global unit in the ring of semialgebraic functions. This global unit admits a global multiplicative inverse that inverts  $f$  pointwise on  $A$ , and allows us to avoid having to consider localizations of function rings to locally invert such functions.

**definition** *fun-glue where*

$\text{fun-glue } n \ S \ f \ g = (\lambda x \in \text{carrier } (Q_p^n). \text{ if } x \in S \text{ then } f \ x \text{ else } g \ x)$

**lemma** *fun-glueE*:

**assumes**  $f \in \text{carrier } (SA \ n)$   
**assumes**  $g \in \text{carrier } (SA \ n)$   
**assumes**  $S \subseteq \text{carrier } (Q_p^n)$   
**assumes**  $x \in S$   
**shows**  $\text{fun-glue } n \ S \ f \ g \ x = f \ x$   
*<proof>*

**lemma** *fun-glueE'*:

**assumes**  $f \in \text{carrier } (SA \ n)$   
**assumes**  $g \in \text{carrier } (SA \ n)$   
**assumes**  $S \subseteq \text{carrier } (Q_p^n)$   
**assumes**  $x \in \text{carrier } (Q_p^n) - S$   
**shows**  $\text{fun-glue } n \ S \ f \ g \ x = g \ x$   
*<proof>*

**lemma** *fun-glue-evimage*:

**assumes**  $f \in \text{carrier } (SA \ n)$   
**assumes**  $g \in \text{carrier } (SA \ n)$   
**assumes**  $S \subseteq \text{carrier } (Q_p^n)$   
**shows**  $\text{fun-glue } n \ S \ f \ g^{-1} \ n \ T = ((f^{-1} \ n \ T) \cap S) \cup ((g^{-1} \ n \ T) - S)$   
*<proof>*

**lemma** *fun-glue-partial-pullback*:

**assumes**  $f \in \text{carrier } (SA \ k)$   
**assumes**  $g \in \text{carrier } (SA \ k)$   
**assumes**  $S \subseteq \text{carrier } (Q_p^k)$   
**shows**  $\text{partial-pullback } k \ (\text{fun-glue } k \ S \ f \ g) \ n \ T =$   
 $((\text{cartesian-product } S \ (\text{carrier } (Q_p^n))) \cap \text{partial-pullback } k \ f \ n \ T) \cup$   
 $((\text{partial-pullback } k \ g \ n \ T) - (\text{cartesian-product } S \ (\text{carrier } (Q_p^n))))$   
*<proof>*

**lemma** *fun-glue-eval-closed*:

**assumes**  $f \in \text{carrier } (SA \ n)$   
**assumes**  $g \in \text{carrier } (SA \ n)$   
**assumes** *is-semialgebraic*  $n \ S$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**shows**  $\text{fun-glue } n \ S \ f \ g \ x \in \text{carrier } Q_p$   
*<proof>*

**lemma** *fun-glue-closed*:

**assumes**  $f \in \text{carrier } (SA \ n)$   
**assumes**  $g \in \text{carrier } (SA \ n)$   
**assumes** *is-semialgebraic*  $n \ S$   
**shows**  $\text{fun-glue } n \ S \ f \ g \in \text{carrier } (SA \ n)$   
*<proof>*

**lemma** *fun-gluе-unit*:

**assumes**  $f \in \text{carrier } (SA\ n)$

**assumes** *is-semialgebraic*  $n\ S$

**assumes**  $\bigwedge x. x \in S \implies f\ x \neq \mathbf{0}$

**shows** *fun-gluе*  $n\ S\ f\ \mathbf{1}_{SA\ n} \in \text{Units } (SA\ n)$

*<proof>*

**definition** *parametric-fun-gluе where*

*parametric-fun-gluе*  $n\ Xs\ fs = (\lambda x \in \text{carrier } (Q_p^n). \text{let } S = (\text{THE } S. S \in Xs \wedge x \in S) \text{ in } (fs\ S\ x))$

**lemma** *parametric-fun-gluе-formula*:

**assumes** *Xs partitions*  $(\text{carrier } (Q_p^n))$

**assumes**  $x \in S$

**assumes**  $S \in Xs$

**shows** *parametric-fun-gluе*  $n\ Xs\ fs\ x = fs\ S\ x$

*<proof>*

**definition** *char-fun where*

*char-fun*  $n\ S = (\lambda x \in \text{carrier } (Q_p^n). \text{if } x \in S \text{ then } \mathbf{1} \text{ else } \mathbf{0})$

**lemma** *char-fun-is-semialg*:

**assumes** *is-semialgebraic*  $n\ S$

**shows** *char-fun*  $n\ S \in \text{carrier } (SA\ n)$

*<proof>*

**lemma** *SA-finsum-apply*:

**assumes** *finite*  $S$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**shows**  $F \in S \rightarrow \text{carrier } (SA\ n) \longrightarrow \text{finsum } (SA\ n)\ F\ S\ x = (\bigoplus_{s \in S. F\ s\ x})$

*<proof>*

**lemma** *SA-finsum-apply-zero*:

**assumes** *finite*  $S$

**assumes**  $F \in S \rightarrow \text{carrier } (SA\ n)$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**assumes**  $\bigwedge s. s \in S \implies F\ s\ x = \mathbf{0}$

**shows** *finsum*  $(SA\ n)\ F\ S\ x = \mathbf{0}$

*<proof>*

**lemma** *parametric-fun-gluе-is-SA*:

**assumes** *finite*  $Xs$

**assumes** *Xs partitions*  $(\text{carrier } (Q_p^n))$

**assumes**  $fs \in Xs \rightarrow \text{carrier } (SA\ n)$

**assumes**  $\forall S \in Xs. \text{is-semialgebraic } n\ S$

**shows** *parametric-fun-gluе*  $n\ Xs\ fs \in \text{carrier } (SA\ n)$

*<proof>*

### 14.10.2 Turning Functions into Units Via Gluing

By gluing a function to the multiplicative unit on its zero set, we can get a canonical choice of local multiplicative inverse of a function  $f$ . Denef's proof frequently reasons about functions of the form  $\frac{f(x)}{g(x)}$  with the tacit understanding that they are meant to be defined on the largest domain of definition possible. This technical tool allows us to replicate this kind of reasoning in our formal proofs.

**definition** *to-fun-unit where*

*to-fun-unit*  $n$   $f = \text{fun-glue } n \{x \in \text{carrier } (Q_p^n). f x \neq \mathbf{0}\} f \mathbf{1}_{SA\ n}$

**lemma** *to-fun-unit-is-unit:*

**assumes**  $f \in \text{carrier } (SA\ n)$

**shows** *to-fun-unit*  $n$   $f \in \text{Units } (SA\ n)$

*<proof>*

**lemma** *to-fun-unit-closed:*

**assumes**  $f \in \text{carrier } (SA\ n)$

**shows** *to-fun-unit*  $n$   $f \in \text{carrier } (SA\ n)$

*<proof>*

**lemma** *to-fun-unit-eq:*

**assumes**  $f \in \text{carrier } (SA\ n)$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**assumes**  $f x \neq \mathbf{0}$

**shows** *to-fun-unit*  $n$   $f x = f x$

*<proof>*

**lemma** *to-fun-unit-eq':*

**assumes**  $f \in \text{carrier } (SA\ n)$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**assumes**  $f x = \mathbf{0}$

**shows** *to-fun-unit*  $n$   $f x = \mathbf{1}$

*<proof>*

**definition** *one-over-fun where*

*one-over-fun*  $n$   $f = \text{inv}_{SA\ n} (\text{to-fun-unit } n\ f)$

**lemma** *one-over-fun-closed:*

**assumes**  $f \in \text{carrier } (SA\ n)$

**shows** *one-over-fun*  $n$   $f \in \text{carrier } (SA\ n)$

*<proof>*

**lemma** *one-over-fun-eq:*

**assumes**  $f \in \text{carrier } (SA\ n)$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**assumes**  $f x \neq \mathbf{0}$

**shows** *one-over-fun*  $n$   $f x = \text{inv } (f x)$

$\langle proof \rangle$

**lemma** *one-over-fun-smult-eval*:

**assumes**  $f \in carrier (SA\ n)$

**assumes**  $a \neq \mathbf{0}$

**assumes**  $a \in carrier\ Q_p$

**assumes**  $x \in carrier (Q_p^n)$

**assumes**  $f\ x \neq \mathbf{0}$

**shows**  $one-over-fun\ n\ (a \odot_{SA\ n} f)\ x = inv\ (a \otimes (f\ x))$

$\langle proof \rangle$

**lemma** *one-over-fun-smult-eval'*:

**assumes**  $f \in carrier (SA\ n)$

**assumes**  $a \neq \mathbf{0}$

**assumes**  $a \in carrier\ Q_p$

**assumes**  $x \in carrier (Q_p^n)$

**assumes**  $f\ x \neq \mathbf{0}$

**shows**  $one-over-fun\ n\ (a \odot_{SA\ n} f)\ x = inv\ a \otimes inv\ (f\ x)$

$\langle proof \rangle$

**lemma** *SA-add-pow-closed*:

**assumes**  $f \in carrier (SA\ n)$

**shows**  $([k::nat] \cdot_{SA\ n} f) \in carrier (SA\ n)$

$\langle proof \rangle$

**lemma** *one-over-fun-add-pow-eval*:

**assumes**  $f \in carrier (SA\ n)$

**assumes**  $x \in carrier (Q_p^n)$

**assumes**  $f\ x \neq \mathbf{0}$

**assumes**  $(k::nat) > 0$

**shows**  $one-over-fun\ n\ ([k] \cdot_{SA\ n} f)\ x = inv\ ([k] \cdot f\ x)$

$\langle proof \rangle$

**lemma** *one-over-fun-pow-closed*:

**assumes**  $f \in carrier (SA\ n)$

**shows**  $one-over-fun\ n\ (f[\uparrow]_{SA\ n}(k::nat)) \in carrier (SA\ n)$

$\langle proof \rangle$

**lemma** *one-over-fun-pow-eval*:

**assumes**  $f \in carrier (SA\ n)$

**assumes**  $x \in carrier (Q_p^n)$

**assumes**  $f\ x \neq \mathbf{0}$

**shows**  $one-over-fun\ n\ (f[\uparrow]_{SA\ n}(k::nat))\ x = inv\ ((f\ x) [\uparrow] k)$

$\langle proof \rangle$

## 14.11 Inclusions of Lower Dimensional Function Rings

**definition** *fun-inc* where

*fun-inc*  $m\ n\ f = (\lambda\ x \in \text{carrier } (Q_p^m). f\ (\text{take } n\ x))$

**lemma** *fun-inc-closed*:

**assumes**  $f \in \text{carrier } (SA\ n)$

**assumes**  $m \geq n$

**shows**  $\text{fun-inc } m\ n\ f \in \text{carrier } (SA\ m)$

*<proof>*

**lemma** *fun-inc-eval*:

**assumes**  $x \in \text{carrier } (Q_p^m)$

**shows**  $\text{fun-inc } m\ n\ f\ x = f\ (\text{take } n\ x)$

*<proof>*

**lemma** *ord-congruence-set-univ-semialg-fixed*:

**assumes**  $n \geq 0$

**shows** *is-univ-semialgebraic* (*ord-congruence-set*  $n\ a$ )

*<proof>*

**lemma** *ord-congruence-set-SA-function*:

**assumes**  $n \geq 0$

**assumes**  $c \in \text{carrier } (SA\ (m+l))$

**shows** *is-semialgebraic* ( $m+l$ )  $\{x \in \text{carrier } (Q_p^{m+l}). c\ x \in \text{nonzero } Q_p \wedge \text{ord } (c\ x) \bmod n = a\}$

*<proof>*

**lemma** *ac-cong-set-SA*:

**assumes**  $n > 0$

**assumes**  $k \in \text{Units } (Zp\text{-res-ring } n)$

**assumes**  $c \in \text{carrier } (SA\ (m+l))$

**shows** *is-semialgebraic* ( $m+l$ )  $\{x \in \text{carrier } (Q_p^{m+l}). c\ x \in \text{nonzero } Q_p \wedge \text{ac } n\ (c\ x) = k\}$

*<proof>*

**lemma** *ac-cong-set-SA'*:

**assumes**  $n > 0$

**assumes**  $k \in \text{Units } (Zp\text{-res-ring } n)$

**assumes**  $c \in \text{carrier } (SA\ m)$

**shows** *is-semialgebraic*  $m$   $\{x \in \text{carrier } (Q_p^m). c\ x \in \text{nonzero } Q_p \wedge \text{ac } n\ (c\ x) = k\}$

*<proof>*

**lemma** *ac-cong-set-SA''*:

**assumes**  $n > 0$

**assumes**  $m > 0$

**assumes**  $k \in \text{Units } (Zp\text{-res-ring } n)$

**assumes**  $c \in \text{carrier } (SA\ m)$

**assumes**  $\bigwedge x. x \in \text{carrier } (Q_p^m) \implies c\ x \neq \mathbf{0}$   
**shows** *is-semialgebraic*  $m \{x \in \text{carrier } (Q_p^m). \text{ac } n\ (c\ x) = k\}$   
 <proof>

## 14.12 Miscellaneous

**lemma** *nth-pow-wits-SA-fun-prep*:  
**assumes**  $n > 0$   
**assumes**  $h \in \text{carrier } (SA\ m)$   
**assumes**  $\varrho \in \text{nth-pow-wits } n$   
**shows** *is-semialgebraic*  $m\ (h^{-1}\ m\ \text{pow-res } n\ \varrho)$   
 <proof>

**definition** *kth-rt where*  
*kth-rt*  $m\ (k::\text{nat})\ f\ x = (\text{if } x \in \text{carrier } (Q_p^m) \text{ then } (THE\ b. b \in \text{carrier } Q_p \wedge b[\wedge]k = (f\ x) \wedge \text{ac } (\text{nat } (\text{ord } ([k]\cdot\mathbf{1})) + 1)\ b = 1)$   
else undefined )

Normalizing a semialgebraic function to have a constant angular component

**lemma** *ac-res-Unit-inc*:  
**assumes**  $n > 0$   
**assumes**  $t \in \text{Units } (Zp\text{-res-ring } n)$   
**shows**  $\text{ac } n\ ([t]\cdot\mathbf{1}) = t$   
 <proof>

**lemma** *val-of-res-Unit*:  
**assumes**  $n > 0$   
**assumes**  $t \in \text{Units } (Zp\text{-res-ring } n)$   
**shows**  $\text{val } ([t]\cdot\mathbf{1}) = 0$   
 <proof>

**lemma**(in *padic-integers*) *res-map-is-hom*:  
**assumes**  $N > 0$   
**shows** *ring-hom-ring*  $Zp\ (Zp\text{-res-ring } N)\ (\lambda\ x. x\ N)$   
 <proof>

**lemma** *ac-of-fraction*:  
**assumes**  $N > 0$   
**assumes**  $a \in \text{nonzero } Q_p$   
**assumes**  $b \in \text{nonzero } Q_p$   
**shows**  $\text{ac } N\ (a \div b) = \text{ac } N\ a \otimes_{Zp\text{-res-ring } N} \text{inv } Zp\text{-res-ring } N\ \text{ac } N\ b$   
 <proof>

**lemma** *pow-res-eq-rel*:  
**assumes**  $n > 0$   
**assumes**  $b \in \text{carrier } Q_p$   
**shows**  $\{x \in \text{carrier } Q_p. \text{pow-res } n\ x = \text{pow-res } n\ b\} = \text{pow-res } n\ b$   
 <proof>

**lemma** *pow-res-is-univ-semialgebraic'*:

**assumes**  $n > 0$

**assumes**  $b \in \text{carrier } Q_p$

**shows** *is-univ-semialgebraic*  $\{x \in \text{carrier } Q_p. \text{pow-res } n \ x = \text{pow-res } n \ b\}$

*<proof>*

**lemma** *evimage-eqI*:

**assumes**  $c \in \text{carrier } (SA \ n)$

**shows**  $c^{-1}_n \{x \in \text{carrier } Q_p. P \ x\} = \{x \in \text{carrier } (Q_p^n). P \ (c \ x)\}$

*<proof>*

**lemma** *SA-pow-res-is-semialgebraic*:

**assumes**  $n > 0$

**assumes**  $b \in \text{carrier } Q_p$

**assumes**  $c \in \text{carrier } (SA \ N)$

**shows** *is-semialgebraic*  $N \ \{x \in \text{carrier } (Q_p^N). \text{pow-res } n \ (c \ x) = \text{pow-res } n \ b\}$

*<proof>*

**lemma** *eint-diff-imp-eint*:

**assumes**  $a \in \text{nonzero } Q_p$

**assumes**  $b \in \text{carrier } Q_p$

**assumes**  $\text{val } a = \text{val } b + \text{eint } i$

**shows**  $b \in \text{nonzero } Q_p$

*<proof>*

**lemma** *SA-minus-eval*:

**assumes**  $f \in \text{carrier } (SA \ n)$

**assumes**  $g \in \text{carrier } (SA \ n)$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**shows**  $(f \ominus_{SA \ n} g) \ x = f \ x \ominus g \ x$

*<proof>*

**lemma** *Qp-cong-set-evimage*:

**assumes**  $f \in \text{carrier } (SA \ n)$

**assumes**  $a \in \text{carrier } Z_p$

**shows** *is-semialgebraic*  $n \ (f^{-1}_n \ (Qp\text{-cong-set } \alpha \ a))$

*<proof>*

**lemma** *SA-constant-res-set-semialg*:

**assumes**  $l \in \text{carrier } (Zp\text{-res-ring } n)$

**assumes**  $f \in \text{carrier } (SA \ m)$

**shows** *is-semialgebraic*  $m \ \{x \in \text{carrier } (Q_p^m). f \ x \in \mathcal{O}_p \wedge Qp\text{-res } (f \ x) \ n = l\}$

*<proof>*

**lemma** *val-ring-cong-set*:

**assumes**  $f \in \text{carrier } (SA \ k)$

**assumes**  $\bigwedge x. x \in \text{carrier } (Q_p^k) \implies f \ x \in \mathcal{O}_p$

**assumes**  $t \in \text{carrier } (Zp\text{-res-ring } n)$

**shows** *is-semialgebraic*  $k \{x \in \text{carrier } (Q_p^k). \text{to-Zp } (f \ x) \ n = t\}$   
 ⟨*proof*⟩

**lemma** *val-ring-pullback-SA*:

**assumes**  $N > 0$

**assumes**  $c \in \text{carrier } (SA \ N)$

**shows** *is-semialgebraic*  $N \{x \in \text{carrier } (Q_p^N). \ c \ x \in \mathcal{O}_p\}$

⟨*proof*⟩

**lemma**(in *padic-fields*) *res-eq-set-is-semialg*:

**assumes**  $k > 0$

**assumes**  $c \in \text{carrier } (SA \ k)$

**assumes**  $s \in \text{carrier } (Zp\text{-res-ring } n)$

**shows** *is-semialgebraic*  $k \{x \in \text{carrier } (Q_p^k). \ c \ x \in \mathcal{O}_p \wedge \text{to-Zp } (c \ x) \ n = s\}$

⟨*proof*⟩

**lemma** *SA-constant-res-set-semialg'*:

**assumes**  $f \in \text{carrier } (SA \ m)$

**assumes**  $C \in Qp\text{-res-classes } n$

**shows** *is-semialgebraic*  $m \ (f \ ^{-1}_m \ C)$

⟨*proof*⟩

## 14.13 Semialgebraic Polynomials

**lemma** *UP-SA-n-is-ring*:

**shows** *ring*  $(UP \ (SA \ n))$

⟨*proof*⟩

**lemma** *UP-SA-n-is-cring*:

**shows** *cring*  $(UP \ (SA \ n))$

⟨*proof*⟩

The evaluation homomorphism from  $\mathbb{Q}_p\text{-funs}$  to  $\mathbb{Q}_p$

**definition** *eval-hom* **where**

*eval-hom*  $a = (\lambda f. \ f \ a)$

**lemma** *eval-hom-is-hom*:

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows** *ring-hom-ring*  $(Fun_n \ Q_p) \ Q_p \ (eval\text{-hom } a)$

⟨*proof*⟩

the homomorphism from  $Fun \ n \ \mathbb{Q}_p \ [x]$  to  $\mathbb{Q}_p \ [x]$  induced by evaluation of coefficients

**definition** *Qp-fpoly-to-Qp-poly* **where**

*Qp-fpoly-to-Qp-poly*  $n \ a = \text{poly-lift-hom } (Fun_n \ Q_p) \ Q_p \ (eval\text{-hom } a)$

**lemma** *Qp-fpoly-to-Qp-poly-is-hom*:

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows**  $(Qp\text{-fpoly-to-Qp-poly } n \ a) \in \text{ring-hom } (UP \ (Fun_n \ Q_p)) \ (Q_p\text{-}x)$

*<proof>*

**lemma** *Qp-fpoly-to-Qp-poly-extends-apply:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $f \in \text{carrier } (\text{Fun}_n Q_p)$

**shows**  $Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ (\text{to-polynomial } (\text{Fun}_n Q_p) \ f) = \text{to-polynomial } Q_p \ (f \ a)$

*<proof>*

**lemma** *Qp-fpoly-to-Qp-poly-X-var:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows**  $Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ (X\text{-poly } (\text{Fun}_n Q_p)) = X\text{-poly } Q_p$

*<proof>*

**lemma** *Qp-fpoly-to-Qp-poly-monom:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $f \in \text{carrier } (\text{Fun}_n Q_p)$

**shows**  $Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ (\text{up-ring.monom } (UP \ (\text{Fun}_n Q_p)) \ f \ m) = \text{up-ring.monom } Q_p\text{-x } (f \ a) \ m$

*<proof>*

**lemma** *Qp-fpoly-to-Qp-poly-coeff:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $f \in \text{carrier } (UP \ (\text{Fun}_n Q_p))$

**shows**  $Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ f \ k = (f \ k) \ a$

*<proof>*

**lemma** *Qp-fpoly-to-Qp-poly-eval:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $P \in \text{carrier } (UP \ (\text{Fun}_n Q_p))$

**assumes**  $f \in \text{carrier } (\text{Fun}_n Q_p)$

**shows**  $(UP\text{-cring.to-fun } (\text{Fun}_n Q_p) \ P \ f) \ a = UP\text{-cring.to-fun } Q_p \ (Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ P) \ (f \ a)$

*<proof>*

**lemma** *Qp-fpoly-to-Qp-poly-sub:*

**assumes**  $f \in \text{carrier } (UP \ (\text{Fun}_n Q_p))$

**assumes**  $g \in \text{carrier } (UP \ (\text{Fun}_n Q_p))$

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows**  $Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ (\text{compose } (\text{Fun}_n Q_p) \ f \ g) = \text{compose } Q_p \ (Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ f) \ (Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ g)$

*<proof>*

**lemma** *Qp-fpoly-to-Qp-poly-taylor-poly:*

**assumes**  $F \in \text{carrier } (UP \ (\text{Fun}_n Q_p))$

**assumes**  $c \in \text{carrier } (\text{Fun}_n Q_p)$

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows**  $Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ (\text{taylor-expansion } (\text{Fun}_n Q_p) \ c \ F) = \text{taylor-expansion } Q_p \ (c \ a) \ (Qp\text{-fpoly-to-}Qp\text{-poly } n \ a \ F)$

*<proof>*

**lemma** *SA-is-UP-cring:*  
**shows** *UP-cring (SA n)*  
*<proof>*

**lemma** *eval-hom-is-SA-hom:*  
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows** *ring-hom-ring (SA n)  $Q_p$  (eval-hom a)*  
*<proof>*

the homomorphism from  $(SA\ n)[x]$  to  $Q_p[x]$  induced by evaluation of coefficients

**definition** *SA-poly-to-Qp-poly where*  
*SA-poly-to-Qp-poly n a = poly-lift-hom (SA n)  $Q_p$  (eval-hom a)*

**lemma** *SA-poly-to-Qp-poly-is-hom:*  
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $(SA\text{-poly-to-}Q_p\text{-poly } n\ a) \in \text{ring-hom } (UP\ (SA\ n))\ (Q_p\text{-}x)$   
*<proof>*

**lemma** *SA-poly-to-Qp-poly-closed:*  
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**assumes**  $P \in \text{carrier } (UP\ (SA\ n))$   
**shows**  $SA\text{-poly-to-}Q_p\text{-poly } n\ a\ P \in \text{carrier } Q_p\text{-}x$   
*<proof>*

**lemma** *SA-poly-to-Qp-poly-add:*  
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in \text{carrier } (UP\ (SA\ n))$   
**assumes**  $g \in \text{carrier } (UP\ (SA\ n))$   
**shows**  $SA\text{-poly-to-}Q_p\text{-poly } n\ a\ (f \oplus_{UP\ (SA\ n)} g) = SA\text{-poly-to-}Q_p\text{-poly } n\ a\ f$   
 $\oplus_{Q_p\text{-}x} SA\text{-poly-to-}Q_p\text{-poly } n\ a\ g$   
*<proof>*

**lemma** *SA-poly-to-Qp-poly-minus:*  
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in \text{carrier } (UP\ (SA\ n))$   
**assumes**  $g \in \text{carrier } (UP\ (SA\ n))$   
**shows**  $SA\text{-poly-to-}Q_p\text{-poly } n\ a\ (f \ominus_{UP\ (SA\ n)} g) = SA\text{-poly-to-}Q_p\text{-poly } n\ a\ f$   
 $\ominus_{Q_p\text{-}x} SA\text{-poly-to-}Q_p\text{-poly } n\ a\ g$   
*<proof>*

**lemma** *SA-poly-to-Qp-poly-mult:*  
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**assumes**  $f \in \text{carrier } (UP\ (SA\ n))$   
**assumes**  $g \in \text{carrier } (UP\ (SA\ n))$   
**shows**  $SA\text{-poly-to-}Q_p\text{-poly } n\ a\ (f \otimes_{UP\ (SA\ n)} g) = SA\text{-poly-to-}Q_p\text{-poly } n\ a\ f$

$\otimes_{Q_p-x}$  *SA-poly-to-Qp-poly*  $n$   $a$   $g$   
*<proof>*

**lemma** *SA-poly-to-Qp-poly-extends-apply:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $f \in \text{carrier } (SA\ n)$

**shows** *SA-poly-to-Qp-poly*  $n$   $a$  (*to-polynomial*  $(SA\ n)$   $f$ ) = *to-polynomial*  $Q_p$  ( $f$   $a$ )

*<proof>*

**lemma** *SA-poly-to-Qp-poly-X-var:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**shows** *SA-poly-to-Qp-poly*  $n$   $a$  (*X-poly*  $(SA\ n)$ ) = *X-poly*  $Q_p$

*<proof>*

**lemma** *SA-poly-to-Qp-poly-X-plus:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $c \in \text{carrier } (SA\ n)$

**shows** *SA-poly-to-Qp-poly*  $n$   $a$  (*X-poly-plus*  $(SA\ n)$   $c$ ) = *UPQ.X-plus*  $(c\ a)$

*<proof>*

**lemma** *SA-poly-to-Qp-poly-X-minus:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $c \in \text{carrier } (SA\ n)$

**shows** *SA-poly-to-Qp-poly*  $n$   $a$  (*X-poly-minus*  $(SA\ n)$   $c$ ) = *UPQ.X-minus*  $(c\ a)$

*<proof>*

**lemma** *SA-poly-to-Qp-poly-monom:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $f \in \text{carrier } (SA\ n)$

**shows** *SA-poly-to-Qp-poly*  $n$   $a$  (*up-ring.monom*  $(UP\ (SA\ n))$   $f$   $m$ ) = *up-ring.monom*  $Q_p$ - $x$  ( $f$   $a$ )  $m$

*<proof>*

**lemma** *SA-poly-to-Qp-poly-coeff:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $f \in \text{carrier } (UP\ (SA\ n))$

**shows** *SA-poly-to-Qp-poly*  $n$   $a$   $f$   $k$  =  $(f\ k)$   $a$

*<proof>*

**lemma** *SA-poly-to-Qp-poly-eval:*

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $P \in \text{carrier } (UP\ (SA\ n))$

**assumes**  $f \in \text{carrier } (SA\ n)$

**shows** (*UP-crimg.to-fun*  $(SA\ n)$   $P$   $f$ )  $a$  = *UP-crimg.to-fun*  $Q_p$  (*SA-poly-to-Qp-poly*  $n$   $a$   $P$ ) ( $f$   $a$ )

*<proof>*

**lemma** *SA-poly-to-Qp-poly-sub:*

**assumes**  $f \in \text{carrier } (UP \ (SA \ n))$   
**assumes**  $g \in \text{carrier } (UP \ (SA \ n))$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $SA\text{-poly-to-}Q_p\text{-poly } n \ a \ (compose \ (SA \ n) \ f \ g) = compose \ Q_p \ (SA\text{-poly-to-}Q_p\text{-poly } n \ a \ f) \ (SA\text{-poly-to-}Q_p\text{-poly } n \ a \ g)$   
 <proof>

**lemma** *SA-poly-to- $Q_p$ -poly-deg-bound:*  
**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $deg \ Q_p \ (SA\text{-poly-to-}Q_p\text{-poly } m \ x \ g) \leq deg \ (SA \ m) \ g$   
 <proof>

**lemma** *SA-poly-to- $Q_p$ -poly-taylor-poly:*  
**assumes**  $F \in \text{carrier } (UP \ (SA \ n))$   
**assumes**  $c \in \text{carrier } (SA \ n)$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $SA\text{-poly-to-}Q_p\text{-poly } n \ a \ (taylor\text{-expansion } (SA \ n) \ c \ F) = taylor\text{-expansion } Q_p \ (c \ a) \ (SA\text{-poly-to-}Q_p\text{-poly } n \ a \ F)$   
 <proof>

**lemma** *SA-poly-to- $Q_p$ -poly-comm-taylor-term:*  
**assumes**  $F \in \text{carrier } (UP \ (SA \ n))$   
**assumes**  $c \in \text{carrier } (SA \ n)$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $SA\text{-poly-to-}Q_p\text{-poly } n \ a \ (UP\text{-cring.taylor-term } (SA \ n) \ c \ F \ i) = UP\text{-cring.taylor-term } Q_p \ (c \ a) \ (SA\text{-poly-to-}Q_p\text{-poly } n \ a \ F) \ i$   
 <proof>

**lemma** *SA-poly-to- $Q_p$ -poly-comm-pderiv:*  
**assumes**  $F \in \text{carrier } (UP \ (SA \ n))$   
**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $SA\text{-poly-to-}Q_p\text{-poly } n \ a \ (UP\text{-cring.pderiv } (SA \ n) \ F) = UP\text{-cring.pderiv } Q_p \ (SA\text{-poly-to-}Q_p\text{-poly } n \ a \ F)$   
 <proof>

**lemma** *SA-poly-to- $Q_p$ -poly-pderiv:*  
**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $UPQ.pderiv \ (SA\text{-poly-to-}Q_p\text{-poly } m \ x \ g) = (SA\text{-poly-to-}Q_p\text{-poly } m \ x \ (pderiv \ m \ g))$   
 <proof>

**lemma**(in *UP-cring*) *pderiv-deg-lt:*  
**assumes**  $f \in \text{carrier } (UP \ R)$   
**assumes**  $deg \ R \ f > 0$   
**shows**  $deg \ R \ (pderiv \ f) < deg \ R \ f$   
 <proof>

**lemma** *deg-pderiv*:

**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $\text{deg } (SA \ m) \ f > 0$

**shows**  $\text{deg } (SA \ m) \ (\text{pderiv } m \ f) = \text{deg } (SA \ m) \ f - 1$

*<proof>*

**lemma** *SA-poly-to-Qp-poly-smult*:

**assumes**  $a \in \text{carrier } (SA \ m)$

**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $x \in \text{carrier } (Q_p^m)$

**shows**  $SA\text{-poly-to-}Qp\text{-poly } m \ x \ (a \odot_{UP} \ (SA \ m) \ f) = a \ x \odot_{UP \ Q_p} \ SA\text{-poly-to-}Qp\text{-poly } m \ x \ f$

*<proof>*

**lemma** *SA-poly-constant-res-class-semialg*:

**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $\bigwedge i \ x. \ x \in \text{carrier } (Q_p^m) \implies f \ i \ x \in \mathcal{O}_p$

**assumes**  $\text{deg } (SA \ m) \ f \leq d$

**assumes**  $C \in \text{poly-res-classes } n \ d$

**shows**  $\text{is-semialgebraic } m \ \{x \in \text{carrier } (Q_p^m). \ SA\text{-poly-to-}Qp\text{-poly } m \ x \ f \in C\}$

*<proof>*

Maps a polynomial  $F(t) \in UP(SA_n)$  to a function sending  $(t, a) \in (Q_p(n+1)) \mapsto F(a)(t) \in Q_p$

**definition** *SA-poly-to-SA-fun where*

$SA\text{-poly-to-}SA\text{-fun } n \ P = (\lambda a \in \text{carrier } (Q_p^{Suc \ n}). \ UP\text{-cring.to-fun } Q_p \ (SA\text{-poly-to-}Qp\text{-poly } n \ (tl \ a) \ P) \ (hd \ a))$

**lemma** *SA-poly-to-SA-fun-is-fun*:

**assumes**  $P \in \text{carrier } (UP \ (SA \ n))$

**shows**  $SA\text{-poly-to-}SA\text{-fun } n \ P \in (\text{carrier } (Q_p^{Suc \ n}) \rightarrow \text{carrier } Q_p)$

*<proof>*

**lemma** *SA-poly-to-SA-fun-formula*:

**assumes**  $P \in \text{carrier } (UP \ (SA \ n))$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**assumes**  $t \in \text{carrier } Q_p$

**shows**  $SA\text{-poly-to-}SA\text{-fun } n \ P \ (t\#x) = (SA\text{-poly-to-}Qp\text{-poly } n \ x \ P) \cdot t$

*<proof>*

**lemma** *semialg-map-comp-in-SA*:

**assumes**  $f \in \text{carrier } (SA \ n)$

**assumes**  $\text{is-semialg-map } m \ n \ g$

**shows**  $(\lambda a \in \text{carrier } (Q_p^m). \ f \ (g \ a)) \in \text{carrier } (SA \ m)$

*<proof>*

**lemma** *tl-comp-in-SA*:

**assumes**  $f \in \text{carrier } (SA \ n)$

**shows**  $(\lambda a \in \text{carrier } (Q_p^{Suc \ n}). \ f \ (tl \ a)) \in \text{carrier } (SA \ (Suc \ n))$

*<proof>*

**lemma** *SA-poly-to-SA-fun-add-eval:*

**assumes**  $f \in \text{carrier } (UP \ (SA \ n))$

**assumes**  $g \in \text{carrier } (UP \ (SA \ n))$

**assumes**  $a \in \text{carrier } (Q_p^{Suc \ n})$

**shows**  $SA\text{-poly-to-SA-fun } n \ (f \oplus_{UP \ (SA \ n)} \ g) \ a = SA\text{-poly-to-SA-fun } n \ f \ a \oplus_{Q_p}$   
 $SA\text{-poly-to-SA-fun } n \ g \ a$

*<proof>*

**lemma** *SA-poly-to-SA-fun-add:*

**assumes**  $f \in \text{carrier } (UP \ (SA \ n))$

**assumes**  $g \in \text{carrier } (UP \ (SA \ n))$

**shows**  $SA\text{-poly-to-SA-fun } n \ (f \oplus_{UP \ (SA \ n)} \ g) = SA\text{-poly-to-SA-fun } n \ f \oplus_{SA \ (Suc \ n)}$   
 $SA\text{-poly-to-SA-fun } n \ g$

*<proof>*

**lemma** *SA-poly-to-SA-fun-monom:*

**assumes**  $f \in \text{carrier } (SA \ n)$

**assumes**  $a \in \text{carrier } (Q_p^{Suc \ n})$

**shows**  $SA\text{-poly-to-SA-fun } n \ (up\text{-ring.monom } (UP \ (SA \ n)) \ f \ k) \ a = (f \ (tl \ a)) \otimes (hd$   
 $a) [\uparrow]_{Q_p} \ k$

*<proof>*

**lemma** *SA-poly-to-SA-fun-monom':*

**assumes**  $f \in \text{carrier } (SA \ n)$

**assumes**  $x \in \text{carrier } (Q_p^n)$

**assumes**  $t \in \text{carrier } Q_p$

**shows**  $SA\text{-poly-to-SA-fun } n \ (up\text{-ring.monom } (UP \ (SA \ n)) \ f \ k) \ (t\#x) = (f$   
 $x) \otimes t [\uparrow]_{Q_p} \ k$

*<proof>*

**lemma** *hd-is-semialg-function:*

**assumes**  $n > 0$

**shows** *is-semialg-function*  $n \ hd$

*<proof>*

**lemma** *SA-poly-to-SA-fun-monom-closed:*

**assumes**  $f \in \text{carrier } (SA \ n)$

**shows**  $SA\text{-poly-to-SA-fun } n \ (up\text{-ring.monom } (UP \ (SA \ n)) \ f \ k) \in \text{carrier } (SA$   
 $(Suc \ n))$

*<proof>*

**lemma** *SA-poly-to-SA-fun-is-SA:*

**assumes**  $P \in \text{carrier } (UP \ (SA \ n))$

**shows**  $SA\text{-poly-to-SA-fun } n \ P \in \text{carrier } (SA \ (Suc \ n))$

*<proof>*

**lemma** *SA-poly-to-SA-fun-mult:*  
**assumes**  $f \in \text{carrier } (UP \ (SA \ n))$   
**assumes**  $g \in \text{carrier } (UP \ (SA \ n))$   
**shows**  $SA\text{-poly-to-SA-fun } n \ (f \otimes_{UP \ (SA \ n)} g) = SA\text{-poly-to-SA-fun } n \ f \otimes_{SA \ (Suc \ n)} SA\text{-poly-to-SA-fun } n \ g$   
*<proof>*

**lemma** *SA-poly-to-SA-fun-one:*  
**shows**  $SA\text{-poly-to-SA-fun } n \ (\mathbf{1}_{UP \ (SA \ n)}) = \mathbf{1}_{SA \ (Suc \ n)}$   
*<proof>*

**lemma** *SA-poly-to-SA-fun-ring-hom:*  
**shows**  $SA\text{-poly-to-SA-fun } n \in \text{ring-hom } (UP \ (SA \ n)) \ (SA \ (Suc \ n))$   
*<proof>*

**lemma** *SA-poly-to-SA-fun-taylor-term:*  
**assumes**  $F \in \text{carrier } (UP \ (SA \ n))$   
**assumes**  $c \in \text{carrier } (SA \ n)$   
**assumes**  $x \in \text{carrier } (Q_p^n)$   
**assumes**  $t \in \text{carrier } Q_p$   
**assumes**  $f = SA\text{-poly-to-Qp-poly } n \ x \ F$   
**shows**  $SA\text{-poly-to-SA-fun } n \ (UP\text{-cring.taylor-term } (SA \ n) \ c \ F \ k) \ (t\#x) = (taylor\text{-expansion } Q_p \ (c \ x) \ f \ k) \otimes (t \ominus c \ x)[\checkmark]_{Q_p} \ k$   
*<proof>*

**lemma** *SA-finsum-eval:*  
**assumes** *finite*  $I$   
**assumes**  $F \in I \rightarrow \text{carrier } (SA \ m)$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $(\bigoplus_{SA \ m}^{i \in I} F \ i) \ x = (\bigoplus_{i \in I} F \ i) \ x$   
*<proof>*

**lemma**(*in ring*) *finsum-ring-hom:*  
**assumes** *ring*  $S$   
**assumes**  $h \in \text{ring-hom } R \ S$   
**assumes**  $F \in I \rightarrow \text{carrier } R$   
**assumes** *finite*  $I$   
**shows**  $h \ (\bigoplus_{i \in I} F \ i) = (\bigoplus_{i \in I} h \ (F \ i))$   
*<proof>*

**lemma** *SA-poly-to-SA-fun-finsum:*  
**assumes** *finite*  $I$   
**assumes**  $F \in I \rightarrow \text{carrier } (UP \ (SA \ m))$   
**assumes**  $f = (\bigoplus_{UP \ (SA \ m)}^{i \in I} F \ i)$   
**assumes**  $x \in \text{carrier } (Q_p^{Suc \ m})$   
**shows**  $SA\text{-poly-to-SA-fun } m \ f \ x = (\bigoplus_{i \in I} SA\text{-poly-to-SA-fun } m \ (F \ i) \ x)$   
*<proof>*

**lemma** *SA-poly-to-SA-fun-taylor-expansion:*

**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $c \in \text{carrier } (SA \ m)$

**assumes**  $x \in \text{carrier } (Q_p^{Suc \ m})$

**shows**  $SA\text{-poly-to-SA-fun } m \ f \ x = (\bigoplus_{i \in \{..deg \ (SA \ m) \ f\}} \text{taylor-expansion } (SA \ m) \ c \ f \ i \ (tl \ x) \otimes (hd \ x \ominus c \ (tl \ x)) \ [\uparrow] \ i)$

*<proof>*

**lemma** *SA-deg-one-eval:*

**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $deg \ (SA \ m) \ g = 1$

**assumes**  $\xi \in \text{carrier } (Fun_m \ Q_p)$

**assumes**  $UP\text{-ring.lcf } (SA \ m) \ g \in \text{Units } (SA \ m)$

**assumes**  $\forall x \in \text{carrier } (Q_p^m). \ (SA\text{-poly-to-SA-fun } m \ g) \ (\xi \ x \# x) = \mathbf{0}$

**shows**  $\xi = \ominus_{SA \ m} (g \ 0) \otimes_{SA \ m} (inv_{SA \ m} \ (g \ 1))$

*<proof>*

**lemma** *SA-deg-one-eval':*

**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $deg \ (SA \ m) \ g = 1$

**assumes**  $\xi \in \text{carrier } (Fun_m \ Q_p)$

**assumes**  $UP\text{-ring.lcf } (SA \ m) \ g \in \text{Units } (SA \ m)$

**assumes**  $\forall x \in \text{carrier } (Q_p^m). \ (SA\text{-poly-to-SA-fun } m \ g) \ (\xi \ x \# x) = \mathbf{0}$

**shows**  $\xi \in \text{carrier } (SA \ m)$

*<proof>*

**lemma** *Qp-pow-ConsI:*

**assumes**  $t \in \text{carrier } Q_p$

**assumes**  $x \in \text{carrier } (Q_p^m)$

**shows**  $t \# x \in \text{carrier } (Q_p^{Suc \ m})$

*<proof>*

**lemma** *Qp-pow-ConsE:*

**assumes**  $x \in \text{carrier } (Q_p^{Suc \ m})$

**shows**  $tl \ x \in \text{carrier } (Q_p^m)$

$hd \ x \in \text{carrier } Q_p$

*<proof>*

**lemma**(**in ring**) *add-monoid-one:*

$\mathbf{1}_{add\text{-monoid } R} = \mathbf{0}$

*<proof>*

**lemma**(**in ring**) *add-monoid-carrier:*

$\text{carrier } (add\text{-monoid } R) = \text{carrier } R$

*<proof>*

**lemma**(**in ring**) *finsum-mono-neutral-cong:*

**assumes**  $F \in I \rightarrow \text{carrier } R$

**assumes** *finite*  $I$

**assumes**  $\bigwedge i. i \notin J \implies F i = \mathbf{0}$   
**assumes**  $J \subseteq I$   
**shows**  $\text{finsum } R F I = \text{finsum } R F J$   
 ⟨proof⟩

This lemma helps to formalize statements like "by passing to a partition, we can assume the Taylor coefficients are either always zero or never zero"

**lemma** *SA-poly-to-SA-fun-taylor-on-refined-set:*

**assumes**  $f \in \text{carrier } (UP (SA m))$   
**assumes**  $c \in \text{carrier } (SA m)$   
**assumes** *is-semialgebraic*  $m A$   
**assumes**  $\bigwedge i. A \subseteq SA\text{-zero-set } m (\text{taylor-expansion } (SA m) c f i) \vee A \subseteq SA\text{-nonzero-set } m (\text{taylor-expansion } (SA m) c f i)$   
**assumes**  $a = \text{to-fun-unit } m \circ \text{taylor-expansion } (SA m) c f$   
**assumes**  $\text{inds} = \{i. i \leq \text{deg } (SA m) f \wedge A \subseteq SA\text{-nonzero-set } m (\text{taylor-expansion } (SA m) c f i)\}$   
**assumes**  $x \in A$   
**assumes**  $t \in \text{carrier } Q_p$   
**shows**  $SA\text{-poly-to-SA-fun } m f (t \# x) = (\bigoplus_{i \in \text{inds.}} (a i x) \otimes (t \ominus c x) [\uparrow] i)$   
 ⟨proof⟩

**lemma** *SA-poly-to-Qp-poly-taylor-cfs:*

**assumes**  $f \in \text{carrier } (UP (SA m))$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**assumes**  $c \in \text{carrier } (SA m)$   
**shows**  $\text{taylor-expansion } (SA m) c f i x = \text{taylor-expansion } Q_p (c x) (SA\text{-poly-to-Qp-poly } m x f) i$   
 ⟨proof⟩

### 14.13.1 Common Morphisms on Polynomial Rings

Evaluation homomorphism from multivariable polynomials to semialgebraic functions

**definition** *Qp-ev-hom where*

$Qp\text{-ev-hom } n P = \text{restrict } (Qp\text{-ev } P) (\text{carrier } (Q_p^n))$

**lemma** *Qp-ev-hom-ev:*

**assumes**  $a \in \text{carrier } (Q_p^n)$   
**shows**  $Qp\text{-ev-hom } n P a = Qp\text{-ev } P a$   
 ⟨proof⟩

**lemma** *Qp-ev-hom-closed:*

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $Qp\text{-ev-hom } n f \in \text{carrier } (Q_p^n) \rightarrow \text{carrier } Q_p$   
 ⟨proof⟩

**lemma** *Qp-ev-hom-is-semialg-function:*

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$

**shows** *is-semialg-function*  $n$  ( $Qp\text{-ev-hom } n f$ )  
*<proof>*

**lemma** *Qp-ev-hom-closed'*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $Qp\text{-ev-hom } n f \in \text{carrier } (Fun_n Q_p)$   
*<proof>*

**lemma** *Qp-ev-hom-in-SA*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $Qp\text{-ev-hom } n f \in \text{carrier } (SA\ n)$   
*<proof>*

**lemma** *Qp-ev-hom-add*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $Qp\text{-ev-hom } n (f \oplus_{Q_p[\mathcal{X}_n]} g) = (Qp\text{-ev-hom } n f) \oplus_{SA\ n} (Qp\text{-ev-hom } n g)$   
*<proof>*

**lemma** *Qp-ev-hom-mult*:  
**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**assumes**  $g \in \text{carrier } (Q_p[\mathcal{X}_n])$   
**shows**  $Qp\text{-ev-hom } n (f \otimes_{Q_p[\mathcal{X}_n]} g) = (Qp\text{-ev-hom } n f) \otimes_{SA\ n} (Qp\text{-ev-hom } n g)$   
*<proof>*

**lemma** *Qp-ev-hom-one*:  
**shows**  $Qp\text{-ev-hom } n \mathbf{1}_{Q_p[\mathcal{X}_n]} = \mathbf{1}_{SA\ n}$   
*<proof>*

**lemma** *Qp-ev-hom-is-hom*:  
**shows**  $Qp\text{-ev-hom } n \in \text{ring-hom } (Q_p[\mathcal{X}_n]) (SA\ n)$   
*<proof>*

**lemma** *Qp-ev-hom-constant*:  
**assumes**  $c \in \text{carrier } Q_p$   
**shows**  $Qp\text{-ev-hom } n (Qp.\text{indexed-const } c) = \mathbf{c}_n\ c$   
*<proof>*

**notation** *Qp.variable* ( $\mathbf{v}_-, -$ )

**lemma** *Qp-ev-hom-pvar*:  
**assumes**  $i < n$   
**shows**  $Qp\text{-ev-hom } n (pvar\ Q_p\ i) = \mathbf{v}_{n, i}$   
*<proof>*

**definition** *ext-hd where*  
 $ext\text{-hd } m = (\lambda x \in \text{carrier } (Q_p^m). hd\ x)$

**lemma** *hd-zeroth*:

$length\ x > 0 \implies x!0 = hd\ x$   
 ⟨proof⟩

**lemma** *ext-hd-pvar*:

**assumes**  $m > 0$

**shows**  $ext-hd\ m = (\lambda x \in carrier\ (Q_p^m).\ eval-at-point\ Q_p\ x\ (pvar\ Q_p\ 0))$

⟨proof⟩

**lemma** *ext-hd-closed*:

**assumes**  $m > 0$

**shows**  $ext-hd\ m \in carrier\ (SA\ m)$

⟨proof⟩

**lemma** *UP-Qp-poly-to-UP-SA-is-hom*:

**shows**  $poly-lift-hom\ (Q_p[\mathcal{X}_n])\ (SA\ n)\ (Qp-ev-hom\ n) \in ring-hom\ (UP\ (Q_p[\mathcal{X}_n]))\ (UP\ (SA\ n))$

⟨proof⟩

**definition** *coord-ring-to-UP-SA where*

$coord-ring-to-UP-SA\ n = poly-lift-hom\ (Q_p[\mathcal{X}_n])\ (SA\ n)\ (Qp-ev-hom\ n) \circ to-univ-poly\ (Suc\ n)\ 0$

**lemma** *coord-ring-to-UP-SA-is-hom*:

**shows**  $coord-ring-to-UP-SA\ n \in ring-hom\ (Q_p[\mathcal{X}_{Suc\ n}])\ (UP\ (SA\ n))$

⟨proof⟩

**lemma** *coord-ring-to-UP-SA-add*:

**assumes**  $f \in carrier\ (Q_p[\mathcal{X}_{Suc\ n}])$

**assumes**  $g \in carrier\ (Q_p[\mathcal{X}_{Suc\ n}])$

**shows**  $coord-ring-to-UP-SA\ n\ (f \oplus_{Q_p[\mathcal{X}_{Suc\ n}]} g) = coord-ring-to-UP-SA\ n\ f$

$\oplus_{UP\ (SA\ n)}\ coord-ring-to-UP-SA\ n\ g$

⟨proof⟩

**lemma** *coord-ring-to-UP-SA-mult*:

**assumes**  $f \in carrier\ (Q_p[\mathcal{X}_{Suc\ n}])$

**assumes**  $g \in carrier\ (Q_p[\mathcal{X}_{Suc\ n}])$

**shows**  $coord-ring-to-UP-SA\ n\ (f \otimes_{Q_p[\mathcal{X}_{Suc\ n}]} g) = coord-ring-to-UP-SA\ n\ f$

$\otimes_{UP\ (SA\ n)}\ coord-ring-to-UP-SA\ n\ g$

⟨proof⟩

**lemma** *coord-ring-to-UP-SA-one*:

**shows**  $coord-ring-to-UP-SA\ n\ \mathbf{1}_{Q_p[\mathcal{X}_{Suc\ n}]} = \mathbf{1}_{UP\ (SA\ n)}$

⟨proof⟩

**lemma** *coord-ring-to-UP-SA-closed*:

**assumes**  $f \in carrier\ (Q_p[\mathcal{X}_{Suc\ n}])$

**shows**  $coord-ring-to-UP-SA\ n\ f \in carrier\ (UP\ (SA\ n))$

⟨proof⟩

**lemma** *coord-ring-to-UP-SA-constant:*

**assumes**  $c \in \text{carrier } Q_p$

**shows**  $\text{coord-ring-to-UP-SA } n \text{ (} Q_p.\text{indexed-const } c) = \text{to-polynomial } (SA \ n) \ (\mathbf{c}_n \ c)$

$\langle \text{proof} \rangle$

**lemma** *coord-ring-to-UP-SA-pvar-0:*

**shows**  $\text{coord-ring-to-UP-SA } n \text{ (pvar } Q_p \ 0) = \text{up-ring.monom } (UP \ (SA \ n)) \ \mathbf{1}_{SA \ n} \ 1$

$\langle \text{proof} \rangle$

**lemma** *coord-ring-to-UP-SA-pvar-Suc:*

**assumes**  $i > 0$

**assumes**  $i < \text{Suc } n$

**shows**  $\text{coord-ring-to-UP-SA } n \text{ (pvar } Q_p \ i) = \text{to-polynomial } (SA \ n) \ (\mathbf{v}_{n, i-1})$

$\langle \text{proof} \rangle$

**lemma** *coord-ring-to-UP-SA-eval:*

**assumes**  $f \in \text{carrier } (Q_p[\mathcal{X}_{\text{Suc } n}])$

**assumes**  $a \in \text{carrier } (Q_p^n)$

**assumes**  $t \in \text{carrier } Q_p$

**shows**  $Q_p\text{-ev } f \ (t\#a) = ((SA\text{-poly-to-}Q_p\text{-poly } n \ a \ (\text{coord-ring-to-UP-SA } n \ f))) \cdot t$

$\langle \text{proof} \rangle$

### 14.13.2 Gluing Semialgebraic Polynomials

**definition** *SA-poly-glu* **where**

$SA\text{-poly-glu } m \ S \ f \ g = (\lambda \ n. \text{fun-glu } m \ S \ (f \ n) \ (g \ n))$

**lemma** *SA-poly-glu-closed:*

**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$

**assumes** *is-semialgebraic*  $m \ S$

**shows**  $SA\text{-poly-glu } m \ S \ f \ g \in \text{carrier } (UP \ (SA \ m))$

$\langle \text{proof} \rangle$

**lemma** *SA-poly-glu-deg:*

**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$

**assumes** *is-semialgebraic*  $m \ S$

**assumes**  $\text{deg } (SA \ m) \ f \leq d$

**assumes**  $\text{deg } (SA \ m) \ g \leq d$

**shows**  $\text{deg } (SA \ m) \ (SA\text{-poly-glu } m \ S \ f \ g) \leq d$

$\langle \text{proof} \rangle$

**lemma** *UP-SA-cfs-closed:*

**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$

**shows**  $g k \in \text{carrier } (SA \ m)$   
 ⟨proof⟩

**lemma** *SA-poly-glue-cfs1*:  
**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$   
**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$   
**assumes** *is-semialgebraic*  $m \ S$   
**assumes**  $x \in S$   
**shows**  $(SA\text{-poly-glue } m \ S \ f \ g) \ n \ x = f \ n \ x$   
 ⟨proof⟩

**lemma** *SA-poly-glue-cfs2*:  
**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$   
**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$   
**assumes** *is-semialgebraic*  $m \ S$   
**assumes**  $x \notin S$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $(SA\text{-poly-glue } m \ S \ f \ g) \ n \ x = g \ n \ x$   
 ⟨proof⟩

**lemma** *SA-poly-glue-to-Qp-poly1*:  
**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$   
**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$   
**assumes** *is-semialgebraic*  $m \ S$   
**assumes**  $x \in S$   
**shows**  $SA\text{-poly-to-Qp-poly } m \ x \ (SA\text{-poly-glue } m \ S \ f \ g) = SA\text{-poly-to-Qp-poly } m \ x \ f$   
 ⟨proof⟩

**lemma** *SA-poly-glue-to-Qp-poly2*:  
**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$   
**assumes**  $g \in \text{carrier } (UP \ (SA \ m))$   
**assumes** *is-semialgebraic*  $m \ S$   
**assumes**  $x \notin S$   
**assumes**  $x \in \text{carrier } (Q_p^m)$   
**shows**  $SA\text{-poly-to-Qp-poly } m \ x \ (SA\text{-poly-glue } m \ S \ f \ g) = SA\text{-poly-to-Qp-poly } m \ x \ g$   
 ⟨proof⟩

### 14.13.3 Polynomials over the Valuation Ring

**definition** *integral-on where*  
*integral-on*  $m \ B = \{f \in \text{carrier } (UP \ (SA \ m)). (\forall x \in B. \forall i. SA\text{-poly-to-Qp-poly } m \ x \ f \ i \in \mathcal{O}_p)\}$

**lemma** *integral-on-memI*:  
**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$   
**assumes**  $\bigwedge x \ i. x \in B \implies SA\text{-poly-to-Qp-poly } m \ x \ f \ i \in \mathcal{O}_p$

**shows**  $f \in \text{integral-on } m \ B$   
 $\langle \text{proof} \rangle$

**lemma** *integral-on-memE*:  
**assumes**  $f \in \text{integral-on } m \ B$   
**shows**  $f \in \text{carrier } (UP \ (SA \ m))$   
 $\bigwedge x. x \in B \implies SA\text{-poly-to-}Qp\text{-poly } m \ x \ f \ i \in \mathcal{O}_p$   
 $\langle \text{proof} \rangle$

**lemma** *one-integral-on*:  
**assumes**  $B \subseteq \text{carrier } (Q_p^m)$   
**shows**  $1 \ UP \ (SA \ m) \in \text{integral-on } m \ B$   
 $\langle \text{proof} \rangle$

**lemma** *integral-on-plus*:  
**assumes**  $B \subseteq \text{carrier } (Q_p^m)$   
**assumes**  $f \in \text{integral-on } m \ B$   
**assumes**  $g \in \text{integral-on } m \ B$   
**shows**  $f \oplus_{UP \ (SA \ m)} g \in \text{integral-on } m \ B$   
 $\langle \text{proof} \rangle$

**lemma** *integral-on-times*:  
**assumes**  $B \subseteq \text{carrier } (Q_p^m)$   
**assumes**  $f \in \text{integral-on } m \ B$   
**assumes**  $g \in \text{integral-on } m \ B$   
**shows**  $f \otimes_{UP \ (SA \ m)} g \in \text{integral-on } m \ B$   
 $\langle \text{proof} \rangle$

**lemma** *integral-on-a-minus*:  
**assumes**  $B \subseteq \text{carrier } (Q_p^m)$   
**assumes**  $f \in \text{integral-on } m \ B$   
**shows**  $\ominus_{UP \ (SA \ m)} f \in \text{integral-on } m \ B$   
 $\langle \text{proof} \rangle$

**lemma** *integral-on-subring*:  
**assumes**  $B \subseteq \text{carrier } (Q_p^m)$   
**shows** *subring* (*integral-on*  $m \ B$ ) (*UP*  $(SA \ m)$ )  
 $\langle \text{proof} \rangle$

**lemma** *val-ring-add-pow*:  
**assumes**  $a \in \text{carrier } Q_p$   
**assumes**  $\text{val } a \geq 0$   
**shows**  $\text{val } ([n::\text{nat}] \cdot a) \geq 0$   
 $\langle \text{proof} \rangle$

**lemma** *val-ring-poly-eval*:  
**assumes**  $f \in \text{carrier } (UP \ Q_p)$   
**assumes**  $\bigwedge i. f \ i \in \mathcal{O}_p$   
**shows**  $\bigwedge x. x \in \mathcal{O}_p \implies f \cdot x \in \mathcal{O}_p$

*<proof>*

**lemma** *SA-poly-constant-res-class-semialg'*:

**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $\bigwedge i \ x. \ x \in B \implies f \ i \ x \in \mathcal{O}_p$

**assumes**  $\text{deg } (SA \ m) \ f \leq d$

**assumes**  $C \in \text{poly-res-classes } n \ d$

**assumes** *is-semialgebraic*  $m \ B$

**shows** *is-semialgebraic*  $m \ \{x \in B. \ SA\text{-poly-to-}\mathcal{O}_p\text{-poly } m \ x \ f \in C\}$

*<proof>*

**lemma** *SA-poly-constant-res-class-decomp*:

**assumes**  $f \in \text{carrier } (UP \ (SA \ m))$

**assumes**  $\bigwedge i \ x. \ x \in B \implies f \ i \ x \in \mathcal{O}_p$

**assumes**  $\text{deg } (SA \ m) \ f \leq d$

**assumes** *is-semialgebraic*  $m \ B$

**shows**  $B = \left( \bigcup C \in \text{poly-res-classes } n \ d. \ \{x \in B. \ SA\text{-poly-to-}\mathcal{O}_p\text{-poly } m \ x \ f \in C\} \right)$

*<proof>*

**end**

**context** *UP-cring*

**begin**

**lemma** *pderiv-deg-bound*:

**assumes**  $p \in \text{carrier } P$

**assumes**  $\text{deg } R \ p \leq (\text{Suc } d)$

**shows**  $\text{deg } R \ (pderiv \ p) \leq d$

*<proof>*

**lemma**(**in** *cring*) *minus-zero*:

$a \in \text{carrier } R \implies a \ominus \mathbf{0} = a$

*<proof>*

**lemma** (**in** *UP-cring*) *taylor-expansion-at-zero*:

**assumes**  $g \in \text{carrier } (UP \ R)$

**shows** *taylor-expansion*  $R \ \mathbf{0} \ g = g$

*<proof>*

**end**

## 14.14 Partitioning Semialgebraic Sets By Zero Sets of Function

**context** *padic-fields*

**begin**

**definition** *SA-funs-to-SA-decomp* **where**

*SA-funs-to-SA-decomp*  $n \ Fs \ S = \text{atoms-of } ((\cap) \ S \ ' ((SA\text{-zero-set } n \ ' Fs) \cup (SA\text{-nonzero-set$

$n \text{ ' } Fs)))$

**lemma** *SA-funs-to-SA-decomp-closed-0:*

**assumes**  $Fs \subseteq \text{carrier } (SA \ n)$

**assumes** *is-semialgebraic*  $n \ S$

**shows**  $(\cap) \ S \text{ ' } ((SA\text{-zero-set } n \text{ ' } Fs) \cup (SA\text{-nonzero-set } n \text{ ' } Fs)) \subseteq \text{semialg-sets } n$   
*<proof>*

**lemma** *SA-funs-to-SA-decomp-closed:*

**assumes** *finite*  $Fs$

**assumes**  $Fs \subseteq \text{carrier } (SA \ n)$

**assumes** *is-semialgebraic*  $n \ S$

**shows** *SA-funs-to-SA-decomp*  $n \ Fs \ S \subseteq \text{semialg-sets } n$

*<proof>*

**lemma** *SA-funs-to-SA-decomp-finite:*

**assumes** *finite*  $Fs$

**assumes**  $Fs \subseteq \text{carrier } (SA \ n)$

**assumes** *is-semialgebraic*  $n \ S$

**shows** *finite* (*SA-funs-to-SA-decomp*  $n \ Fs \ S$ )

*<proof>*

**lemma** *SA-funs-to-SA-decomp-disjoint:*

**assumes** *finite*  $Fs$

**assumes**  $Fs \subseteq \text{carrier } (SA \ n)$

**assumes** *is-semialgebraic*  $n \ S$

**shows** *disjoint* (*SA-funs-to-SA-decomp*  $n \ Fs \ S$ )

*<proof>*

**lemma** *pre-SA-funs-to-SA-decomp-in-algebra:*

**shows**  $((\cap) \ S \text{ ' } (SA\text{-zero-set } n \text{ ' } Fs \cup SA\text{-nonzero-set } n \text{ ' } Fs)) \subseteq \text{gen-boolean-algebra}$   
 $S \ (SA\text{-zero-set } n \text{ ' } Fs \cup SA\text{-nonzero-set } n \text{ ' } Fs)$

*<proof>*

**lemma** *SA-funs-to-SA-decomp-in-algebra:*

**assumes** *finite*  $Fs$

**shows** *SA-funs-to-SA-decomp*  $n \ Fs \ S \subseteq \text{gen-boolean-algebra } S \ (SA\text{-zero-set } n \text{ ' } Fs \cup SA\text{-nonzero-set } n \text{ ' } Fs)$

*<proof>*

**lemma** *SA-funs-to-SA-decomp-subset:*

**assumes** *finite*  $Fs$

**assumes**  $Fs \subseteq \text{carrier } (SA \ n)$

**assumes** *is-semialgebraic*  $n \ S$

**assumes**  $A \in \text{SA-funs-to-SA-decomp } n \ Fs \ S$

**shows**  $A \subseteq S$

*<proof>*

**lemma** *SA-funs-to-SA-decomp-memE:*

**assumes**  $finite\ Fs$   
**assumes**  $Fs \subseteq carrier\ (SA\ n)$   
**assumes**  $is-semialgebraic\ n\ S$   
**assumes**  $A \in (SA-funs-to-SA-decomp\ n\ Fs\ S)$   
**assumes**  $f \in Fs$   
**shows**  $A \subseteq SA-zero-set\ n\ f \vee A \subseteq SA-nonzero-set\ n\ f$   
 ⟨*proof*⟩

**lemma**  $SA-funs-to-SA-decomp-covers$ :  
**assumes**  $finite\ Fs$   
**assumes**  $Fs \neq \{\}$   
**assumes**  $Fs \subseteq carrier\ (SA\ n)$   
**assumes**  $is-semialgebraic\ n\ S$   
**shows**  $S = \bigcup (SA-funs-to-SA-decomp\ n\ Fs\ S)$   
 ⟨*proof*⟩

**end**  
**end**

## References

- [1] J. Denef.  $p$ -adic semi-algebraic sets and cell decomposition. *Journal für die reine und angewandte Mathematik*, 369:154–166, 1986.
- [2] A. Engler. *Valued fields*. Springer, Berlin New York, 2005.